MSc Program
Engineering Management

Modernization of the Software-Engineering-Process within a large Project-Environment

A Master's Thesis submitted for the degree of
"Master of Science"

supervised by
Univ.Prof. Dr.techn. Dr.h.c.mult. Peter Kopacek

Dipl.-Ing. (FH) Simon Wartanian
1026893

January 2016, Vienna

Bibliographic notice

Dipl.-Ing. (FH) Wartanian, Simon: “Modernization of the Software-Engineering-Process within a large Project-Environment”, 99 Pages, 33 Figures, TU Wien, Engineering Management

Master Thesis 2016

Abstract

New ways of optimizing large software engineering processes are investigated in this thesis, based on the findings out of the two international standardized frameworks CMMI and SPICE. The goal was to find new approaches, inspired by those two standards.

Two approaches – the modular approach and the workflow-step classification – were introduced and analyzed in this thesis.

The first approach – the modular approach – is based on the idea to keep the workflows inside the processes as dynamic as possible by creating modules which can be used in various ways to fulfill the unique need for the implementation of a certain requirement.

The second approach – the workflow-step classification – describes a new way of how to classify and measure workflows inside processes. This approach was created as a measurement strategy for the modular approach, but can also be used in conservative set ups.

In the following chapters these two approaches are introduced in detail, leading to an experimental use in an example environment.

Affidavit

I, **SIMON WARTANIAN**, hereby declare

1. that I am the sole author of the present Master's Thesis, "MODERNIZATION OF THE SOFTWARE-ENGINEERING-PROCESS WITHIN A LARGE PROJECT-ENVIRONMENT", 99 pages, bound, and that I have not used any source or tool other than those referenced or any other illicit aid or tool, and
2. that I have not prior to this date submitted this Master's Thesis as an examination paper in any form in Austria or abroad.

Vienna, 27.03.2016

_____

Signature

Contents

List of Figures

List of Tables

Preface

Based on the analysis of the standardized process frameworks CMMI and SPICE new optimization approaches are created and challenged in this thesis.

The first approach is the “Modular Approach”. It challenges the static layout of a process and provides a new process layout approach on how to increase the way of dynamic work. Meanwhile it is not harming the creativity and innovative way of work of the employees, but still has a fixed workflow-layout as a basis for measurement to satisfy the needs of every management team.

The second approach is the “Workflow Step Classification”. It concentrates on the workflow-layout inside a process and challenges the way, these layouts are created. It provides different classifications for each workflow-step and a measurement portfolio showing which information can be gathered and how to measure correctly after implementing the “Modular Approach”.

In the second half of this thesis those two approaches are implemented in the processes of a real example project in a theoretical way, explaining the ideas on a practical example. Afterwards, parts of those optimizations are introduced in the production environment of the example project, leading to first results of the quality and success of the approaches.

Abbreviations

BP	**B**ase **P**ractice	Page 17
CL	**C**apability **L**evel	Page 12
CMMI	**C**apability **M**aturity **M**odel **I**ntegration	Page 2
ENG	**Eng**ineering Process Group	Page 24
GP	**G**eneric **P**ractice	Page 18
GR	**G**eneric **R**esource	Page 18
ISO	**I**nternational **O**rganization of **S**tandardization	Page 14
KPI	**K**ey **P**erformance **I**ndicator	Page 19
MAN	**Man**agement Process Group	Page 25
ML	**M**aturity **L**evel	Page 11
PA	**P**rocess **A**ttribute	Page 16
PAM	**P**rocess **A**ssessment **M**odel	Page 15
PIM	**P**rocess **Im**provement Process Group	Page 25
PPQA	**P**rocess and **P**roduct **Q**uality **A**ssurance	Page 23
PRM	**P**rocess **R**eference **M**odel	Page 15
QA	**Q**uality **A**ssurance	Page 1
RD	**R**equirements **D**evelopment	Page 22
REQM	**Req**uirements **M**anagement	Page 22
SG	**S**pecific **G**oal	Page 22
SP	**S**pecific **P**ractice	Page 22
SPICE	**S**oftware **P**rocess **I**mprovement and **C**apability D**e**termination	Page 2
SQA	**S**oftware **Q**uality **A**ssurance	Page 23
SQC	**S**oftware **Q**uality **C**ontrol	Page 23
VAL	**Val**idation	Page 23
VER	**Ver**ification	Page 23
WP	**W**ork **P**roduct	Page 17

1. Introduction

Many successful software projects started as small innovative ideas, developed by just few people. Those which were really great, have developed to huge projects with dozens of employees working on it.

Many of the processes in such projects are based on a historical foundation which was set up for a small project. As the project grew from year to year, it was forgotten to adjust the structures and processes in many cases. There are a lot of opportunities in such projects to enhance the efficiency, quality and productivity.

A recent Swiss study shows that the biggest challenge in large software-development-processes is the efficiency, followed by the permanent need of lowering the costs. [cf., (Ziegler, 2015)]

The “World Quality Report” of the international consulting-company Capgemini states that the average investment in Quality Assurance (QA) is 35% of the budget. In 2018 it will even rise up to 40%. [cf., (Capgemini, Sogeti and HP, 2015)]

This thesis should accompany a research for methods which improve historically grown processes and raise the performance and the quality, without harming the dynamic working spirit.

All findings will be tested in the historically grown environment of a real example project, described in chapter 1.3.

1.1 Goals and Delimitation

The goal of this thesis is to find ways to improve processes in terms of quality and performance. Another goal is to investigate a form of optimization, in which the costs will be reduced on a long-term scale, by ensuring a high level of quality and lowering the maintenance costs in the future. Therefore, the research concentrates on ways to improve the quality and the dynamic work, without harming the performance.

As a basis for the optimization strategy the two process frameworks Capability Maturity Model Integration (CMMI) and Software Process Improvement and Capability Determination (SPICE) are analyzed. Based on the findings the processes of the example project should be measured and classified, to identify the areas of improvement.

The goal is to find a new way of improving processes, using the gathered knowledge of the analyzed frameworks as basis. It is not the goal to modify the existing processes strictly based on the standards, leading to an official audit and certification. This thesis is a research for new possibilities of improvement, inspired by the established international standards.

1.2 Structure

In chapter 2 the basis for improvements is investigated, by analyzing the international standards CMMI and SPICE. The most relevant parts of those two big frameworks are described to reach the expected improvements based on the framework findings. Afterwards, the example project's processes are analyzed due to those findings. Due to the discovered optimization opportunities, the most important areas of the frameworks for the optimization are analyzed in detail.

In chapter 3 an optimization approach is created, based on the previous findings. This chapter is the core of the thesis, creating a new optimization approach, based on the identified problems and inspired by the best-practices of the process frameworks. This approach, containing the analyzed best-practices and the new ideas is called aligned approach.

In chapter 4 the aligned approach from chapter 3 is implemented into the environment of the example project. The theoretical idea is implemented within a practical example as a basis for the experimental use in the example project.

In chapter 5 parts of the described processes from chapter 4 are implemented in the environment of the example project. A measurement strategy describes the structure of the Key Performance Indicators (KPI) and their meaning. The following figures in this chapter show how parts of the approach from chapter 3.3 work in a real environment.

Chapter 6 summarizes the idea and the findings. As not every part of the approach could be tested in the example environment it also contains an outlook for missing results and opportunities for the theory.

1.3 Example Project

The example project is a part of a big program containing multiple continuous projects, which customize and deliver an Enterprise resource planning (ERP**)** software product to a certain customer.

1.3.1 Software Development Process

Each of those projects run for approximately one year, containing five main phases.

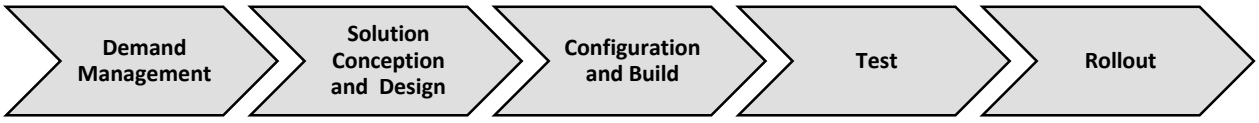

Figure 1.1: Main Phases of Software-Development-Process

Each of these main-processes contains individual sub-processes leading to a process-setup in two hierarchies.

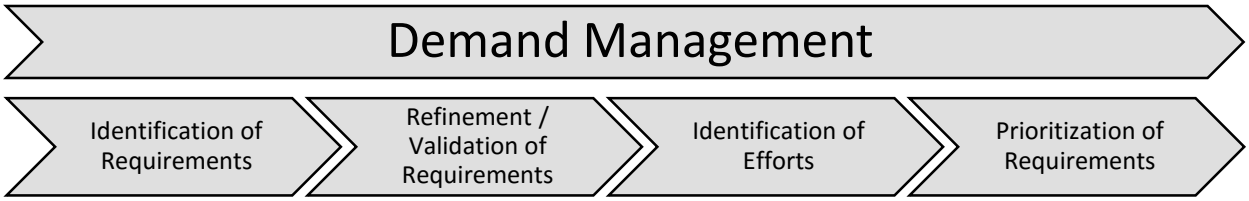

Figure 1.2: Main Phase - Demand Management

At the beginning of each project the needed requirements are identified. All gathered requirements are afterwards refined and validated, leading to a final scope of requirements which will be nominated for implementation within the project. This scope of requirements is analyzed in detail with first effort estimations and second analysis results. Based on this results the requirements are parsed, leading to a final prioritized scope of requirements, which will be implemented during the project.

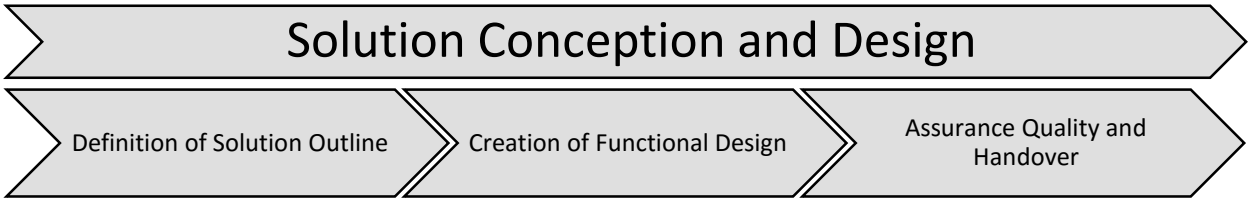

Figure 1.3: Main Phase - Solution Conception and Design

For each requirement of the project-scope a business concept is written, explaining the customization of the existing core-software, to fulfill the need of the requested requirements. After this business concept is aligned with all involved stakeholders a functional design is created as basis for the implementation of the customization.

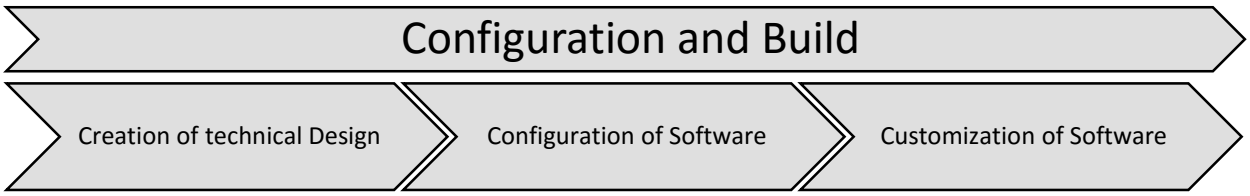

Figure 1.4: Main Phase - Configuration and Build

Based on the functional design a technical design document is created, explaining the way of the customization from a technical point of view. This document is the guideline and

documentation of the following implementation. Based on the functional design the core-software is extended by either a configuration or a customization.

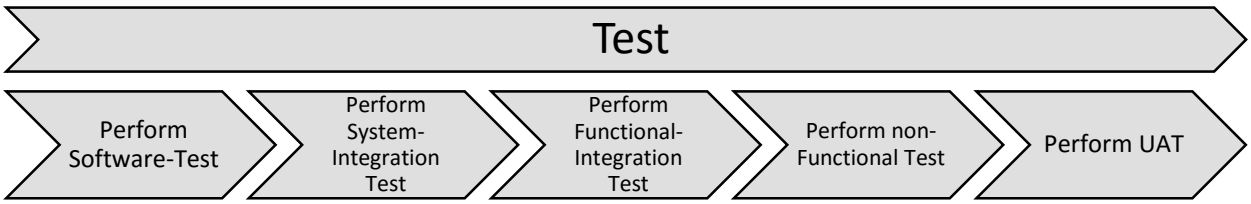

Figure 1.5: Main Phase - Test

Every implementation of a requirement needs to be tested properly afterwards. This includes smoke- and sanity-testing as well as the extension of an existing regression-test-portfolio. After a package of requirements is ready to be delivered to the test-environment of the customer, a system-integration-test also needs to take place. Following this test, a functional-test needs to be proceeded, which not only tests the basic functionality, but also if the implementation fits the requested Requirements by logic. After all project-internal-tests are completed the customer has to perform an user-acceptance-test to ensure the requirements were implemented according to his needs.

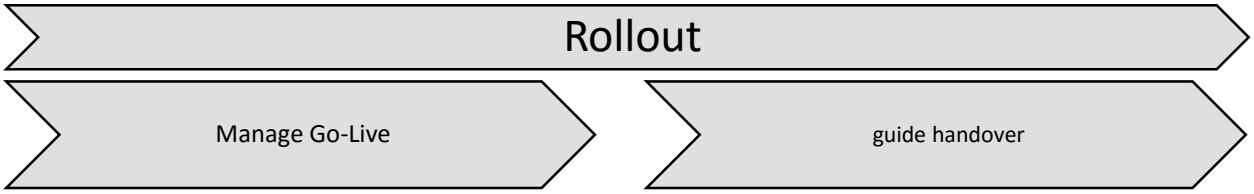

Figure 1.6: Main Phase - Rollout

If all requirements are implemented successfully and the project comes to an end, the finished product needs to be installed as a production system on customer side, leading to an use in production. After this step is done the Build-Project is finished with a handover for further customer support to the Run-organization within the company. The ongoing maintenance of the production-system is handled by a different organizational unit within the company (Run-Organization).

After a project is finished and successfully handed over, the program-team starts with the next customer and the same project-setup all over again, using the same process-landscape.

Currently there are almost thirty of such customers in the queue, waiting for their Build-project to start.

This historically grown process setup will be modified in chapter 4. due to the findings in chapter 2. and chapter 3.

In this large scale example process multiple different roles are working across the described processes. The interaction between these different roles is also a root causing bottlenecks.

1.3.2 Project Roles and Stakeholders

During each project, every person has a pre-defined role with distinguished duties. There are two types of roles. Internal roles inside the project-team and different types of stakeholders who provide services, handle the communication or receive the product.

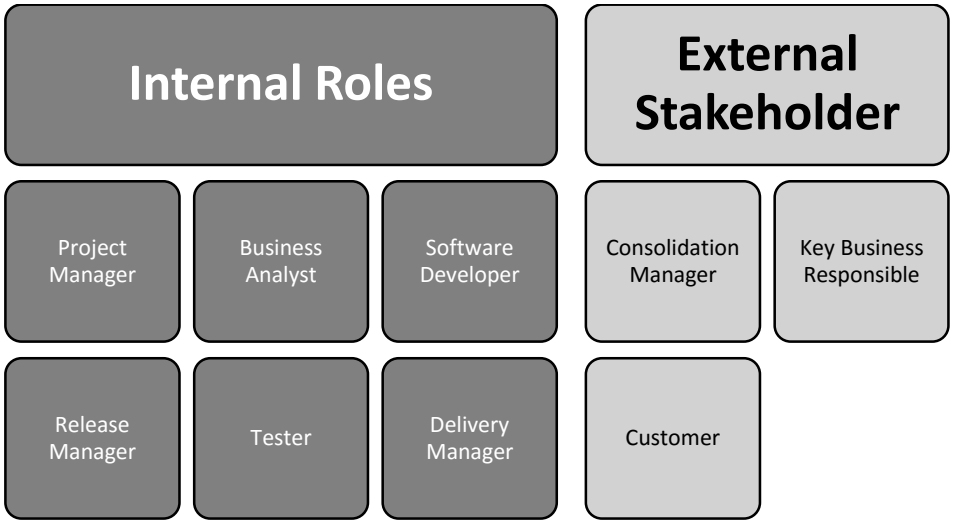

Figure 1.7: Different Project Roles and Stakeholders

1.3.2.1 Project Internal Roles

At the beginning of each project the needed requirements are analyzed and measured by the business analysts, leading to an effort-estimation as an analysis-result. In the next step the project-management together with the involved stakeholders (e.g. customer) decides the priorities and time-schedule for the implementation on this basis, approving those efforts. After

this prioritization, according to its priority every requirement has a deadline, until which it should be implemented.

According to the priority a Business Analyst creates a User Story document for each Requirement, describing a way to solve the requested requirements with a customization of the existing basis-software. This user story needs to be aligned with various stakeholders, leading to the creation of the functional design document. This document is the basis for the technical design and the implementation into the core software, made by the software-developer. After this implementation the Test-Team handles every kind of testing, involving all required stakeholders. At the end, after the successful testing of the new functionality the delivery manager is in charge for handing over the new software-version to the client, including the key business responsible for the specific customer.

Business Analyst: The Business Analyst is the main responsible person for a requirement. He writes the functional design document as basis for the following implementation. Additional to that he is in charge of the communication between all stakeholders and the guidance between the different departments. After the implementation as the testing is finished, it is his responsibility to communicate the completion of each requirement to the stakeholders.

Software Developer: The software developer creates the technical design document as basis for the source-code implementation, based on the functional design provided by the designer. After creating the solution, it is also in his responsibility to test the implementation on a gross level.

Tester: The tester receives a new implemented requirement and tests the implementation in a functional and non-functional way, using common testing methods. Additional to this requirement-testing the tester is responsible for maintaining the permanently growing regression-test-portfolio which is running after every implementation cycle.

Release Manager: After an implementation iteration the release manager is packing together the new version of the application and is in charge of deploying it to the various environments. This contains internal test-environment, as customer test-environment but also the deployment into the production system.

Delivery Manager: The delivery manager is the single point of contact for the customer-side in the project. He is the interface handling the communication with the customer, making it easier to keep the overview over all requested changes. It is also the duty of the Delivery Manager to assist the Customer with functional advices and requirement-creation. Also in the user acceptance tests he is at the customer-side to assist in case of problems and know-how issues.

1.3.3 Different External Stakeholder

Consolidation Manager: The consolidation manager never changes from one project to another. This role is a gatekeeper, observing all requirements so they stay in line with each other and work according to the principles of the software, not changing it, just customizing it.

Customer: The customer is an individual Business Unit which is heading for the software-product created by the project-team. These business units are always working on historical legacy systems which will be migrated to the new solution.

Key Business Responsible: The key business responsible is the single point of contact on customer side (normally well trained on the new software before the project starts). He is the counterpart to the delivery manager on project-side. He knows the business and the involved persons and can judge the situation at the customer side best. He is deeply involved in the phase of requirements gathering, supporting with his know-how and experience.

2. Process-Optimization Frameworks

In this chapter relevant process frameworks will be analyzed as basis for further optimization of the processes in the example project. In chapter 4. the example process will be finally optimized due to the findings in this chapter, according to the strategy developed in chapter 3.

2.1 CMMI – Capability Maturity Model Integration

The Capability Maturity Model Integration – short CMMI – was developed by the Software Engineering Institute of the university of Carnegie Mellon, Pennsylvania. CMMI is a framework to measure the maturity of a process and proposing best-practice solutions for system- and software-engineering. CMMI defines three types of components:

Needed components: As described in the name, these components are mandatory for a certification process (e.g. specific goals of the processes).

Expected components: This type of components are e.g. special practices which lead the activity during a process to a higher level of ability.

Informal components: These components are from informative characteristics. They are used to ease the understanding by describing, making notes or providing examples in natural language to support the understanding of the other components.

Additional to these components, the second pillar of CMMI is the classification of the process-areas. For this CMMI defines standardized core-areas as basis for all application areas. Every process-area contains at least information from the following list:

- Name and abbreviation of the process-area
- Category and grade of maturity of a process-area
- Definition and purpose
- Specific goals and practices

Due to the methodology the process-areas are the basis for all further steps. Based on these areas each process is classified into a category which is pre-defined by the standard. The following figures shows all process-areas from the development-specific framework CMMI-DEV.

(Carnegie Mellon University, 2016)

Category	**Process-Area**	**Abbreviation**
Development	Product Integration	PI
	Requirement Development	RD
	Technical Solution	TS
	Validation	VAL
	Verification	VER
Project Management	Project Monitoring and Control	PMC
	Project Planning	PP
	Requirements Management	REQM
	Supplier Agreement Management	SAM
	Integrated Project Management	IPM
	Risk Management	RSKM
	Quantitative Project Management	QPM
Process Management	Organized Process Definition	OPD
	Organized Process Focus	OPF
	Organized Training	OT
	Organized Process Performance	OPP
	Organized Performance Management	OPM

Support	Configuration Management	CM
	Measurement and Analysis	MA
	Process and Product Quality Assurance	PPQA
	Decision Analyzing and Resolution	DAR
	Causal Analysis and Resolution	CAR

Table 2.1: CMMI - Process-Areas of CMMI-DEV [cf., (CMMI Product Team, 2010)]

2.1.1 CMMI – Maturity Levels

In CMMI all processes can be classified in five different Maturity-Levels (ML), describing the current maturity of the process.

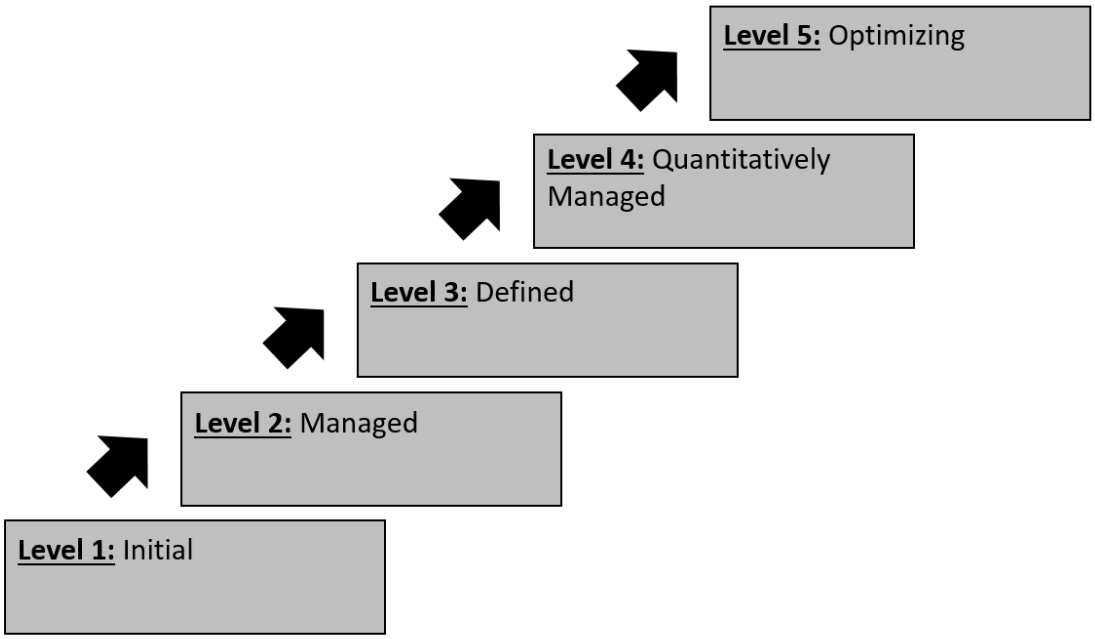

Figure 2.1: CMMI - Maturity Levels

Each of the Levels in Figure 2.1 represents a state in which a process can be, and in which it has a particular definition (CMMI Product Team, 2010):

ML 1 – Initial: Process is unpredictable and poorly to control.

ML 2 – Managed: Process is characterized for projects and can be reactive.

ML 3 – Defined: Process is characterized for an organization and is proactively set up.

ML 4 – Quantitatively Managed: Process is controlled and measured.

ML 5 – Optimizing: Process is running and continuously improving.

2.1.2 CMMI – Capability Levels

Additional to the Maturity-Levels in chapter 2.1.1, each process can be classified in a Capability-Level (CL). These levels concentrate on the quality of the outcome of the process. On the Capability-side four different classifications exist:

CL 0 – Incomplete: The work is done in a way, so the specific goals cannot be reached.

CL 1 – Performed: The work is done in a way, so only the functional goals can be reached.

CL 2 – Managed: The work is done in a properly steered way.

CL 3 – Defined: The work is done in a customized way, based on a standardized process. The way of work is continuously improving.

The capability level describes the level of the output. The measurement is based on the specific goals of a process.

[cf., (CMMI Product Team, 2010)]

2.1.3 CMMI – Interaction of Maturity- and Capability- Levels

Figure 2.2 shows the interaction between the two grading-systems which exist in CMMI: Maturity-Levels and Capability-Levels.

Name	*Abbr.*	*ML*	*CL1*	*CL2*	*CL3*
Configuration Management	CM	2			
Measurement and Analysis	MA	2			
Project Monitoring and Control	PMC	2	**Target Profile 2**		
Project Planning	PP	2			
Process and Product Quality Assurance	PPQA	2			
Requirements Management	REQM	2			
Supplier Agreement Management	SAM	2			
Decision Analysis and Resolution	DAR	3			
Integrated Project Management	IPM	3			
Organizational Process Definition	OPD	3		**Target Profile 3**	
Organizational Process Focus	OPF	3			
Organizational Training	OT	3			
Product Integration	PI	3			
Requirements Development	RD	3			
Risk Management	RSKM	3			
Technical Solution	TS	3			
Validation	VAL	3			
Verification	VER	3			
Organizational Process Performance	OPP	4			
Quantitative Project Management	QPM	4		**Target Profile 4**	
Causal Analysis and Resolution	CAR	5			
Organizational Performance Management	OPM	5		**Target Profile 5**	

Figure 2.2: CMMI – Level Promotion Criteria (CMMI Product Team, 2010)

To reach the next ML each process-area from the previous ML has to be implemented with an output CL, which is described in the last column of Figure 2.2. For example, to reach ML 4, every process-area from ML 2 and ML 3 has to be implemented, with a CL of 3.

[cf., (CMMI Product Team, 2010)]

2.2 SPICE – Software Process Improvement and Capability Determination

SPICE is an international standard, provided by the International Organization for Standardization (ISO). It is delivered as ISO 15504. The standard is structured based on multiple various standard-parts concentrating on two main pillars. The first one, "Capacity Determination", is concentrating on rating an already existing process. For this a process is measured due to its strengths, weaknesses and risks according the requirements for the process-setup. The second main pillar is "Software Process Improvement" which concentrates on the possibilities to improve a particular process. Those two pillars combined are providing a proper basis for improving and controlling of already existing processes.

The main goal of the provided measures is to rise the performance without harming the quality, leading to a higher efficiency. Therefore, three aspects of business-processes are combined:

Assessment: An existing process is analyzed and rated by the pre-defined criteria of the standard by a certified assessor based on collected data.

Improvement: While performing improvements there is always a risk of failure. To minimize that risk, SPICE provides a structured standard how to perform an optimization. An existing process is built on the assessment result, representing the basis for improvements.

Determination: In case an organization considers to outsource a part of its project, SPICE provides a structured way of determining the capacity of the deliveries, including practical process profiles, which also include risk calculations in the result.

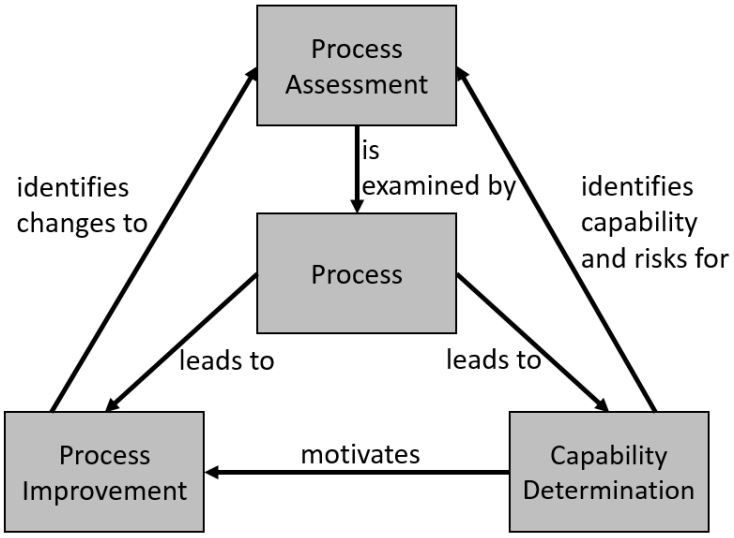

Figure 2.3: SPICE Structure (ISO 15504)

This structure results not only in quality-assurance of the outcome of a process, but also in improvement of the quality during the whole process.

2.2.1 SPICE – Methodology

For the classification of a productive environment the two components, Process-Reference-Model (PRM) and Process-Assessment-Model (PAM), are in charge. PRM divides every process according to its area, taking the output of the process into account as reference for the classification. It is used to describe what the purpose of the process is and why it is used. On the other side, PAM contains a rating-schema for determining the maturity of a process. The rating-schema is build up in six steps, starting from "0 – Incomplete" to "5 – Optimizing".

[cf., (Automotive SIG, 2015)]

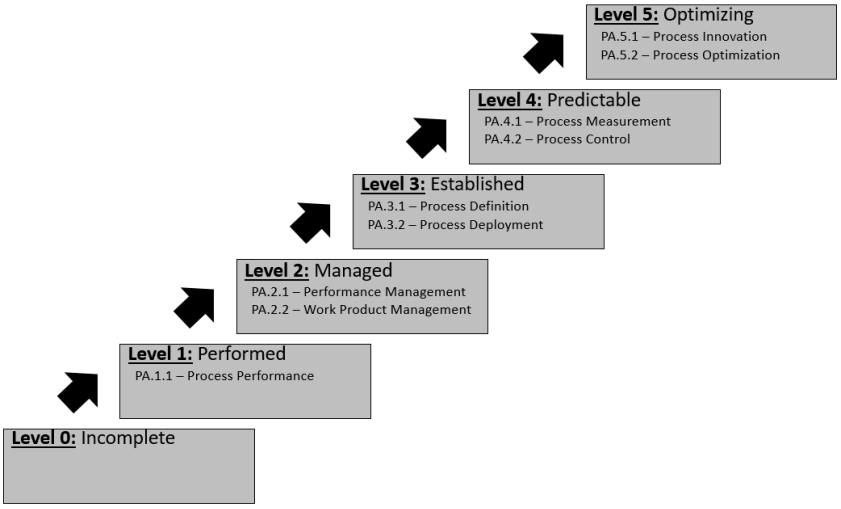

Figure 2.4: SPICE (ISO 15504) - PAM Maturity Levels

Figure 2.4 shows the standardized levels and their attributes of PAM. Every existing process can be classified to one of those levels according to the following definitions. Every level starting from level 1 contains unique Process Attributes (PA) which stand as key-headlines for the level, being used by the process-indicators described in chapter 2.2.4.

Level 0 – Incomplete: Chaotic processes

Level 1 – Performed: Processes are set up intuitively; inputs and outputs exist

Level 2 – Managed: Processes and output are monitored; responsibilities are defined

Level 3 – Established: Predefined standardized processes are customized due to individual situations; resources are monitored

Level 4 – Predictable: Defined measures make monitoring of process and outcome possible

Level 5 – Optimizing: Quantitative measures are used to ensure continuous improvements

[cf., (Automotive SIG, 2010)]

2.2.2 SPICE – Rating Structure

For the rating of an existing process, to classify it into the right classification of the PAM, two different types of rating-indicators exist. The first one, the "Execution-Indicator", is exclusively in use in Level 1 and describes the progress during a process. The second indicator, the **"**Capability-Indicator", describes the maturity of a process from level 0 to 5.

[cf., (Automotive SIG, 2010)]

2.2.3 SPICE – Execution-Indicator

This indicator is the basis for further steps. It is represented on the lowest level of the PAM and therefore the connection between the initial maturity level and the process. Two sub-indicators exist as basis for measurement.

Base Practice (BP): Defines the individual activities which are needed for a process to fulfill its duty. Every process has predefined BPs in the PAM. The default-layout for a BP looks like described in Table 2.2. Find a filled in example in Appendix A.

Process ID	PRO.1
Process Name	<A MEANINGFUL TITLE>
Process Purpose	<A MEANINGFUL DECRIPTION>
Process Outcome	<ALL REQUIRED RESULTS AS AN ENUMERATION>
Base Practices	<AN ENUMERATION WITH ALL BASE PRACTICES>

Table 2.2: PAM BP - Table Layout

[cf., (Automotive SIG, 2010)]

Work Product (WP): Defines the elements which are created, progressed or executed during a process. Also elements which are required for the progress of a process are included in this scope. It is also possible that an element is used by multiple different processes. Every process has predefined WPs in the PAM as well. The default-layout for a WP looks like described in Table 2.3. For an example see Appendix B.

Output Work Products
<WORKING STEP in Format yy-zz> - <TITLE OF STEP> - <INFECTED RESULTS FROM BP>

Table 2.3: PAM WP - Table Layout

[cf., (Automotive SIG, 2010)]

2.2.4 SPICE – Capability-Indicator

For every further level of the PAM the Capability-Indicator is in charge. This indicator is built on sub-indicators which rely on the PA of each PAM level.

GP – Generic Practice: Activities which guide the implementation of a PA in the PAM. They are built on management practices.

GR – Generic Resource: This indicator stands for resources which are used to fulfill a PA in the PAM (e.g. tools, infrastructure but also employees).

[cf., (Automotive SIG, 2007)]

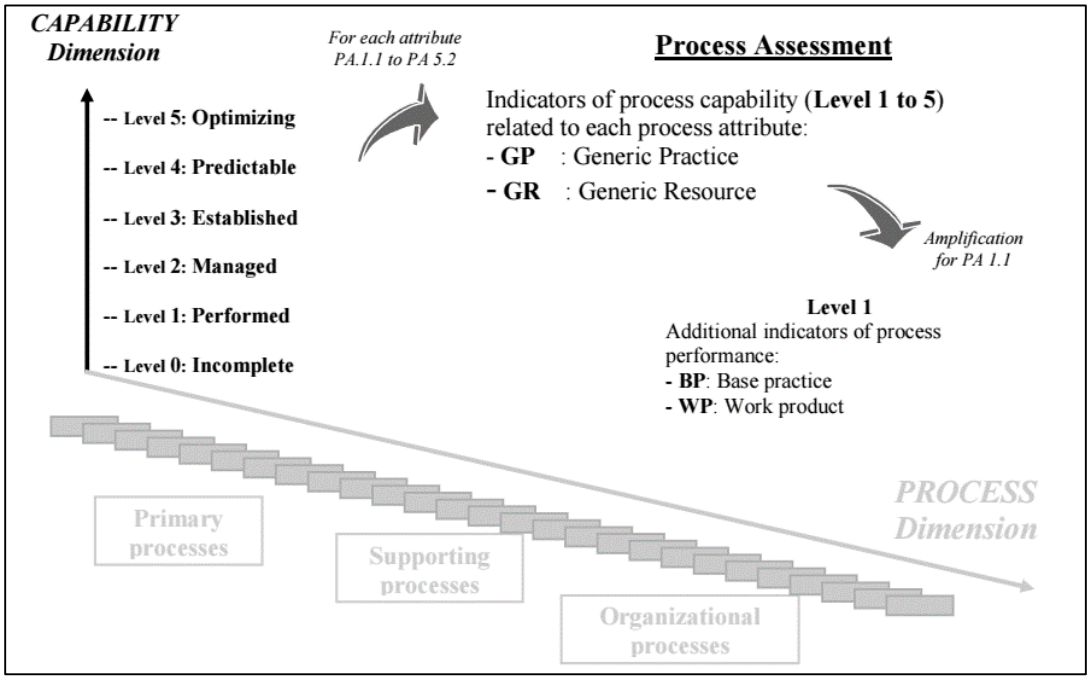

Figure 2.5: PAM - Connection between Indicators [(Automotive SIG, 2007)]

Figure 2.5 shows the mentioned indicators and the connection of them and how they influence each other.

2.3 Comparison CMMI and SPICE

The basic structure of CMMI and SPICE is mostly similar. In both frameworks “Maturity Levels” and “Best Practices” are included in the standard. In both frameworks specialized versions exist, as for the automotive-industry or software-development, to fit the need of the area even better.

In contrast to SPICE it is not possible for CMMI to test all components of the framework during a certification. Additional to mandatory components CMMI contains optional components, which can be compared with the product-reference-model from SPICE. The informative components out of CMMI do not exist in SPICE at all.

While in CMMI goals and their targets are defined as pre-defined process-areas, SPICE works with its process-references. The difference is, that on one hand in CMMI each process-area is linked to an unique Maturity-Level as on the other hand in SPICE such a connection does not exist at all.

Both frameworks rely on classification of an existing process. In CMMI via five “Maturity Levels”, in SPICE the Process-Assessment-Model with its six levels is in charge. Here the major difference between those frameworks is, that in CMMI the conditions to reach the next level are much stricter in the lower levels, compared to SPICE.

Both frameworks have valuable approaches, which will be taken into account while setting up the Key Performance Indicators (KPIs) and measures for the optimization example project’s process setup of the.

2.4 Detailed Findings for Optimization

In this chapter the main process for the example project as introduced in chapter 1.3.1 will first be described in a way, so it can be compared and measured due to the needs of the frameworks. Afterwards, the optimization-strategy will be explained as a basis for the selection

of the process-templates and goals from the frameworks CMMI and SPICE described in chapter 2.4.3.

2.4.1 Example Process Analysis

The initial situation of the example project processes is historically grown and developed. According to the measurement rules analyzed in chapter 2. the processes are already quite good. The classification of the processes is done in a very general and experimental way, due to the finding during the analysis in chapter 2, to identify problems and spots for optimization. It was not proceeded according the rules of an official audit, as the goal of this thesis is to find new approaches of improvement, supported by the ideas of existing frameworks. An audit and certification is not part of this thesis.

According to the Maturity Levels in CMMI the process can be described as a ML 3 (Defined) process, as described in chapter 2.1.1. It is already pro-actively defined due to the circumstances of the Software and customers, but it is not measured yet in a standardized and proper way. Also the quality-gates are not standardized and missing on some important positions of the process. The CL of the process is 2 (Managed) as described in chapter 2.1.2. The setup of the process, to fulfill the needs of the customer works out in many ways. The only problem is, that sometimes the quality of the outcome is not high enough. Under some circumstances the outcome is faulty and contains errors, leading to a high number of defects raised by the customer. If this defects are fixed, the majority of the finished requirements fit the needs of the customer. So the problem is not the integrity, it is the quality. In both measurement-schemas, ML and CL, according to Figure 2.2, the challenges to reach ML 4 are very high. The goal is to concentrate on quality and measurement, and try to find a way to increase the CL in those areas. The challenges are to raise the quality and to setup a continuously developing process environment.

On SPICE side the process can be rated as PAM Level 2 (Managed), according to chapter 2.2.1. Actually the quality of the process setup is higher in many parts, but as many different Stakeholders are involved in the example project, the responsibility is not always clear, which is

a mandatory requirement to reach level 3 (Established). Additional to find a way to clear the responsibility, the PAs to reach the next level are to concentrated on “Performance Management” and “Work Product Management”.

2.4.2 Optimization Strategy

Due to the analysis of the processes of the example project, based on the frameworks described in chapter 2, the optimization should concentrate on following areas:

- **Clarify Responsibility:** Due to the high number of involved Stakeholders, it is not always clear who is responsible for a requirement at which time. This downside of the current process setup in the example project should be improved.
- **Introduce Quality-Gates:** There is a need for quality gates at key points in the project setup to enhance the quality of the output.
- **Raise Performance:** The historically grown setup of the example project has already developed into the right direction. But still there should be looked for a way to raise the average performance.
- **Implement Measurement Strategy:** The biggest down-side of the processes in the example project is the lack of predefined measurement points with standardized KPIs, to measure the processes and to identify bottlenecks and opportunities for improvement.

This four goals are the result of the optimization analysis described in chapter 2.4.1. Based on existing process templates and inspired by the frameworks analyzed in chapter 2, an optimization approach should be created for the example project. The goal is to create new ideas of process improvements, taking the findings of the standards in mind. It is not the goal to implement the standards and make the processes ready for an audit as basis for an official certification.

2.4.3 CMMI and SPICE Process Templates

In CMMI different Process-Areas will be used as basis for further optimizations. In SPICE the PAM contains more than 500 templates for main processes. In both frameworks these processes contain multiple sub-processes. The following sub-chapters show the used processes as basis for the following optimization of the example project processes.

2.4.3.1 CMMI Process-Areas

Requirements Development (RD): This area is a critical process in the lifecycle of every software-development process, containing principles how to do standardized requirements engineering. Following goals are used for the optimization of the example project.

Goal	Description
RD Specific Goal (SG) 1 Develop Customer Requirements	Specific Practice (SP) 1.2 Develop the Customer Requirements
RD SG 3 Analyze and Validate Requirements	SP 3.3 Analyze Requirements SP 3.5 Validate Requirements

Table 2.4: CMMI PA Requirements Development – Used Goals [cf., (Software Quality Assurance.org, 2015)]

Requirements Management (REQM): This process is in charge to identify possible inconsistencies between requirements inside the project scope, which is an important process to save the integrity of the output.

Goal	Description
REQM SG 1 Manage Requirements	SP 1.5 Identify Inconsistencies between Project Work and Requirements

Table 2.5: CMMI PA Requirements Management – Used Goals [cf., (Software Quality Assurance.org, 2015)]

Process and Product Quality Assurance (PPQA): This is the main Software Quality Assurance (SQA) Process in CMMI and is also in charge to guarantee the quality of the outcome.

Goal	Description
PPQA SG 1 Objectively Evaluate Processes and Work Products	SP 1.1 Objectively Evaluate Processes
PPQA SG 2 Provide Objective Insight	SP 2.2 Establish Records

Table 2.6: CMMI PA Process and Product Quality Assurance - Used Goals [cf., (Software Quality Assurance.org, 2015)]

Validation (VAL): This Process stands for Software Quality Control (SQC) and addresses the question, if the right product is built with the implemented solution.

Goal	Description
VAL SG 1 Prepare for Validation	SP 1.1 Establish Validation Procedures and Criteria

Table 2.7: CMMI PA Validation - Used Goals [cf., (Software Quality Assurance.org, 2015)]

Verification (VER): Also this is a SQC process, measuring if the solution was built correctly due to the needs of the customer.

Goal	Description
VER SG 2 Perform Peer Reviews	SP 2.1 Prepare for Peer Reviews SP 2.2 Conduct Peer Reviews
VER SG 3 Verify Selected Work Products	SP 3.1 Perform Verification SP 3.2 Analyze Verification Results

Table 2.8: CMMI PA Verification - Used Goals [cf., (Software Quality Assurance.org, 2015)]

[cf., (Software Quality Assurance.org, 2015)]

2.4.3.2 SPICE Process Assessment Model

Engineering Process Group (ENG): This group contains processes which take care of requirement gathering as well as implementation and testing of a software product.

Process Identification	Process Name
ENG.1	Requirements elicitation
ENG.2	System requirements analysis
ENG.3	System architectural design
ENG.4	Software requirements analysis
ENG.5	Software design
ENG.6	Software construction
ENG.7	Software integration test
ENG.8	Software testing
ENG.9	System integration test
ENG.10	System testing

Table 2.9: SPICE PAM Engineering Process Group – Used Processes [cf., (Automotive SIG, 2010)]

This process should be the basis for the optimization of the main process of the example project. In the newer version of the Automotive SIG document (v3) the ENG process was replaced by a SWE process in 2015. The ENG process layout from the 2010 PAM (v2.5) was chosen for the optimization strategy as it fits the processes in a large environment, as of the example process, better.

[cf., (Automotive SIG, 2010); cf., (Automotive SIG, 2015)]

Support Life Cycle Processes (SUP): These processes can be used by any other category of processes at various points in the lifecycle as an additional improvement of the big picture.

Process Identification	Process Name
SUP.1	Quality assurance
SUP.2	Verification

SUP.9	Problem resolution management
SUP.10	Change request management

Table 2.10: SPICE PAM Supporting Life Cycle Processes – Used Processes [cf., (Automotive SIG, 2015)]

Additional to the main Process this group should raise the quality of the output and provide standardized ways to handle problems.

[cf., (Automotive SIG, 2015)]

Management Process Group (MAN): This type of process is important for those who manage and maintain existing processes.

Process Identification	Process Name
MAN.6	Measurement

Table 2.11: SPICE PAM Management Process Group – Used Processes [cf., (Automotive SIG, 2015)]

This processes should be used to permanently increase the quality of the existing example project and measure it according to the standard.

[cf., (Automotive SIG, 2015)]

Process Improvement Process Group (PIM): This group contains a process how to improve an existing process.

Process Identification	Process Name
PIM.3	Process improvement

Table 2.12: SPICE PAM Process Improvement Process – Used Processes [cf., (Automotive SIG, 2015)]

Additional to the MAN processes this process should be in charge to monitor and improve the existing processes in the example project.

[cf., (Automotive SIG, 2015)]

3. Process-Optimization Approach

In this chapter the individual work, based on the example project and the findings from chapter 2. starts. As basis for the aligned optimization approach there is a need to concentrate on the example project and its unique requirements. In the example project the work is done based on already existing processes, which can be improved based on the finding in chapter 2.4, to raise the quality and performance in a general way. These two pillars, to rise the quality and performance, are not the only goals.

The management in the example project believes in a dynamic way of work. The team is working together, sharing its knowledge, being creative and innovative in finding solutions. This spirit should not be harmed by any optimization. People are thinking and solving problems in their own way, using the processes as a guideline, not as a strict set of steps which needs to be followed. Besides the advantages of the creative work, the downside is, that many requirements are solved in an unique way. It is hard to compare the workflows, as they are individual in many cases. This leads to chaos in the history, making it hard to measure the history and identify the problems.

This leads to the question: What is a process?

"A business process is a collection of activities designed to produce a specific output for a particular customer or market. It implies a strong emphasis on how the work is done within and organization, in contrast to a product's focus on what. A process is thus a specific ordering of work activities across time and place, with a beginning, an end, and clearly defined inputs and outputs: a structure for action.", [(Sparx Systems, 2014)*]*

The definition of a process can be interpreted as a standardized way of doing a certain type of work. But why is a standardization needed? There are many advantages of standardizing a process. This thesis is concentrating on the advantage of measurement to find opportunities for optimization. As basis for measurement you need a standardized way of work. The total chaos cannot be measured. The disadvantage of a standardized process is, that it puts the employee into a corset, limiting his freedom and harming his creativity.

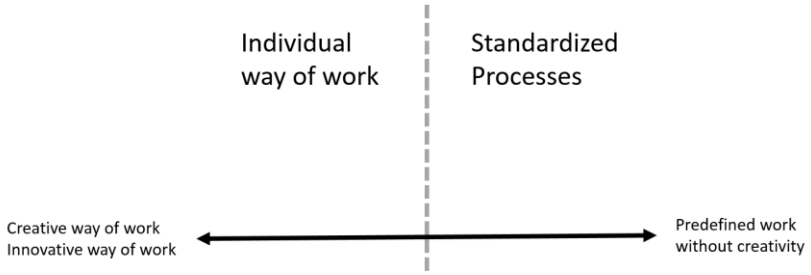

Figure 3.1: Advantages and Disadvantages of standardized Processes

For many jobs there is no need for the employees to be creative, as for example in a factory hall every step is predefined, the corset is very tight. As the input and output is always the same, there is no need for innovative thinking and creative solution management on factory-worker level.

On the other hand, in the software engineering business every requirement is unique. The output is always a unique "product". In this type of a process, innovative and creative thinking is mandatory. But if a project is growing, reaching a high number of employees, there is a need to establish processes. Each process contains steps. The employee has still a freedom to act as he wants inside each step, but the chronological order of the steps is predefined in the processes. This is the maximum freedom each employee has due to the process frameworks.

Strict Processes

The easiest way to improve this situation would be to implement the findings from chapter 2.4, introducing quality-gates, setting up defined measurement points and requesting the employees to stick to the processes in every situation. As described before each employee has the freedom to act as he wants to inside a process-step. The chronological order of the steps is

strictly predefined. This approach would conflict the creative way of work, which should not be harmed.

Aligned Approach

A standardized process, which can be measured, is mandatory. This fact is mandatory to the management and cannot be changed. The mission is to find a way of moving towards "Individual way of work", as described in Figure 3.1, keeping the innovative and creative way of work, but still establishing a new process due to the findings in chapter 2.4, which should be followed step by step. The goals are clear:

1. Establish a standardized optimized process due to the findings of chapter 2.4, which contains more quality gates and predefined measurement points to make monitoring possible, as basis for identifying problems and continuous improvements.
2. The employees should stick strictly to the process.
3. The creative and innovative way of work should not be harmed by strict processes.

As described before especially goal 2 and 3 stay in conflict to each other. During a standardized process, there is just one predefined way of doing things. One the other hand, if there are no processes there is an infinite number of solving a problem. Due to goal 1, a strict process is mandatory. So the solution is to find a way to raise the number of possible standardized workflows inside a process via modularity. This approach will be described in detail in chapter 3.1.

The modular approach (as described in chapter 3.1.) solves the problem about keeping the creative way of work inside the project. As the number of individual processes within this approach is very high, the problem, how to measure such an individual process still exists. The sub-chapter 3.2 describes a classified multidimensional way of measuring the modular process, not only identifying problems, but also providing individual suggestions for the right optimization.

3.1 Modular Approach

A standardized process is always a strict way of solving a problem or creating a product. Even if there is a split inside the workflow, it is a predefined one, leading to a fixed way of work. The number of possible paths in a workflow is not flexible, it is predefined and static. To make the process more dynamic a high number of question-points and splits can be introduced, leading very fast to a confusing complex layout which is hard to overlook.

The idea of the modular approach is, splitting a process into different modules. A module should be a part of the process which describes a whole work-step until reaching a milestone. E.g. a module should be the whole Sales-Process, but it would be meaningless to make a module partly manufactural and sales-processual. The module, with the work which is done in it, should stay a unit of integrity.

Instead of having one large workflow-layout it is cut into multiple smaller parts which represent a part of the whole workflow. These modules are not linked to each other and can be used whenever they are needed. The fact that there is no linking between the modules is leading to getting rid of a hardwired layout which should be avoided. This leads to the creation of artificial question-points between these modules and a high number of possible workflow-paths. The usage of a module should be completely individual.

On the other hand, the workflow-layout inside a module stays strict and hardwired. If a module is used, each *workflow-step* needs to be finished as in a normal process.

Every module can have each module as successor, with some restrictions. There are three classifications of modules:

Finishing-Module: Every process has a defined module at the end. This module should contain quality gates, ensuring that the modules before were not misusing the modular system and that the work was done in a proper way.

Optional-Module: Every other module is an optional module which can be used at every time inside a process. Reusing an optional module is also allowed. There is no order in which modules should be used. At the end of each module the following module is chosen due to the expertise of the employee. Some optional modules require a successor-module, which needs to be used before the finishing-module. A module never requires a predecessor to make the layout of the process as dynamic as possible. To avoid misuse, the responsible employee can deny to start his work, at the beginning of each module sending the requirement back to the last step of the predecessor-module.

Successor-Module: Every optional-module can be a required Successor for another optional-module. For example, a module "Creation Design Document", requires the optional-module "Implement Solution" as Successor, because a new design goes hand in hand with the implementation. The implementation on the other hand can also be done without a new design document. This could happen if the process is used for a defect, in which case the design document was written properly but the implementation was faulty, leading to a skip of the module "Creation Design Document".

3.1.1 Exemplary Explanation

To explain the modular approach in a proper way a simple "Requirements Identification" process is used to describe the approach with a practical example. Following process will be used afterwards for the description:

A – Requirement Identification: In this process the Requirements are identified at the customer side.

B – Requirement Validation: These identified requirements are validated within an expert team to identify if they are valid due to the architectural standards, or if they are already solved in a different way.

C – Requirement Refinement: The requirement is refined or described in a more detailed way in agreement between the project-expert-team and the customer.

D – Requirement Analysis: The requirement is analyzed by the functional Designer, who has to create the Design document afterwards, creating an outline of the proposed solution which is aligned with the customer.

E – Requirements Estimation: Due to the aligned solution outline, the Designer gathers all estimated efforts for the implementation of the requirement (from development-team, test-team and himself).

F – Requirement Prioritization: Due to the findings from the previous processes, the project-manager schedules the solution of the requirements involving the customer and other stakeholder.

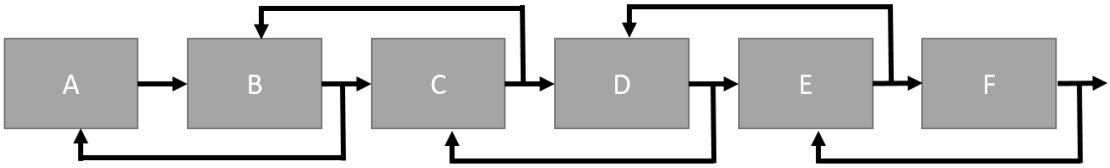

Figure 3.2: Modular Approach Exemplary Explanation - Normal Process

Figure 3.2 shows how this sub-processes would interact in a normal process layout for "Requirements Identification". There is always a way forward, and in case of problems also a way backwards to the predecessor to get feedback. Every sub-process is mandatory, which is also the case in the majority of all requirements. There is just one workflow and no space for exceptions.

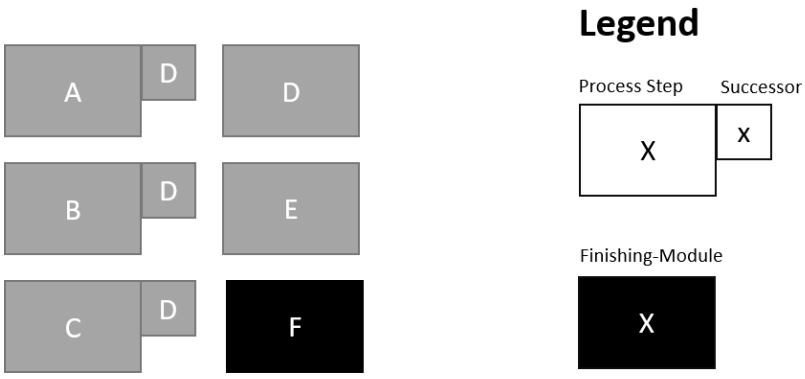

Figure 3.3: Modular Approach Exemplary Explanation - Modular Process

Figure 3.3 shows how the process would look like according the modular approach. Instead of a hardwired layout every step stays as an independent module, which can be used at any time. The only restriction is, that module F “Requirement Prioritization” has to be the Finishing-Module. If the project manager is not satisfied during module F, he has the possibility to deny the request for the module and send the requirement back to the predecessor. The Finishing-Module always contains quality gates checking if the modular approach was misused during the whole process. If one of the Optional-Modules A “Requirement Identification”, B “Requirement Validation or C “Requirement Refinement” is in charge, the Successor-Module D “Requirement Analysis has to be used.

As the finishing-module is the only mandatory module in this example, a process can also only contain the module F “Requirement Prioritization”. This would make sense in case the project manager himself launches a project internal requirement. In such case, there is no need for the steps A – E, and the requirement would still be documented with the standardized modular workflow.

In case the modules A – C are used, the successor-module D is mandatory, because every external requirement needs to be analyzed by an internal Designer as basis for the prioritization in the finishing-module. If the designer is unsatisfied with the requirement, because the customer created it in module A, but skipped the validation in module B, the designer can forward it either to module B or C, according to his expertise. If he is already satisfied with the requirement, he can forward it to prioritization in the finishing-module F. The project manager can request an effort estimation, calling module E, or in case of a production-error, the estimations are irrelevant as the problem has to be fixed as soon as possible, just finish the process, saving a lot of time.

To make the selection of the modules easier for new employees an expert team can setup best-practices for different cases, which would look like Figure 3.4 for the example process.

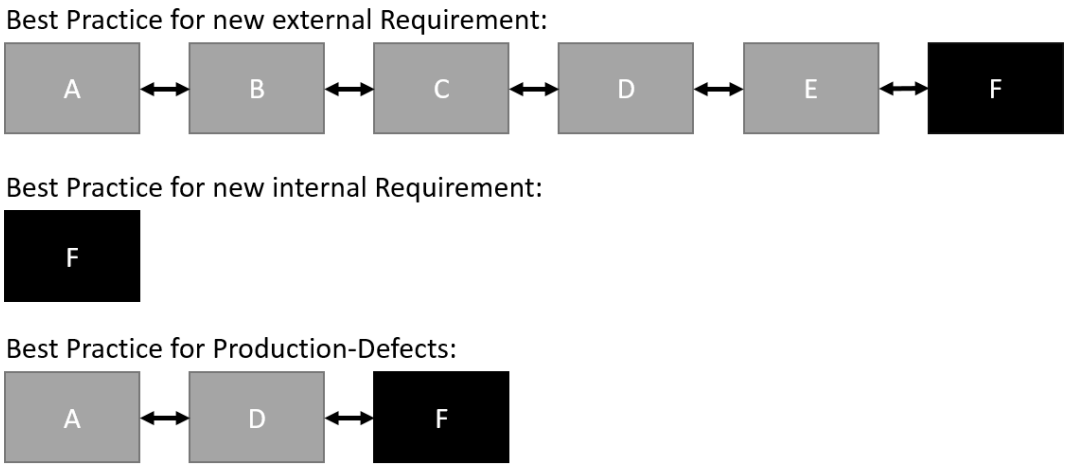

Figure 3.4: Modular Approach Exemplary Explanation - Best Practices

With this modular approach the number of different workflows grew from one to infinite, as every module can be used multiple times. The fact that a module has to be used multiple times would be caused by faulty work in most cases (as for example a bad analysis, which has to be done a second time). In most cases it can probably be compared with a step-back during a traditional process.

In more complex processes it is also possible that a module is called multiple times due to other reasons. For example, in a complex big process, the requirement could change in some parts, leading to a re-analysis, without questioning the quality of the first result. Either way, the modular approach leads to more possibilities in the process.

If every status is just used at most once, the number of theoretically possible process-workflows can be calculated (not every of those paths would make sense, e.g. C-B-A-D-F). Therefore, the process needs to be divided in module-groups, always ending with a successor-module. In the example process two module-groups exists

- A-C with the successor-module D
- E with the finishing-module F

The number of all available different workflows per group can be calculated using following formula:

$$WFs_{group} = n! \left(1 + \sum_{k=1}^{n-1} \frac{1}{(n-k)!}\right) + 1$$

$$WFs_{group} \dots Number\ of\ available\ Workflows\ in\ a\ Module\ Group$$

$$n \dots Number\ of\ Optional\ Modules\ inside\ a\ group$$

The results from all Module-Groups then need to be multiplied, leading to the number of available workflows:

$$\left(\sum_{group}^{Number\ of\ groups} WFs_{group}\right) + 1$$

In a small process, as the example process using this formula already 31 workflows are possible, instead of 1.

[cf., (Mathematik für Informatik, 2010)]

3.2 Step Classification and Measurement setup

Already with the example process from chapter 3.1.1 instead of one standardized workflow inside the process there are 31 possible individual ways in case every module is just used once, according the need of a requirement. This leads to the basic problem, how to setup a standardized measurement strategy as basis for improvements.

The only information which is available in every process for each requirement is the information gathered out of the finishing-modules and the number of used optional-modules.

On the other hand, completely opposite to the dynamic workflow inside a process, the layout of a module is hardwired. Every step has to be used strictly, leading to a basis for overall

measurement on module level. Problems will not be identified on process-level, but on module level.

In opposite to the process-strategy, the module-strategy is based on hardwired steps, which are the deepest entity inside this approach. Quality-Gates should be implemented as a step inside a module, to ensure a standardized use. This leads to the question: What is a quality gate?

A Quality-Gate seems to be a step inside a workflow, which is used to ensure the quality, this could be a revision or a validation. In both analyzed frameworks from chapter 2. quality gates are mentioned as a way of improving the quality of the output. This leads to a classification of some steps inside a workflow as “Quality-Gates” and leads to the question: What kind of classification has the other status in a workflow?

Additional to classifying the Quality-Gates, the idea is to classify all steps inside a workflow as basis for the measurement strategy. The classification approach will be described more detailed in the following chapter.

3.2.1 Workflow Step Classification

A workflow is build up in steps which are connected to each other, ensuring a standardized way of work inside a process, or in this case inside a module. Every step has an input and an output, which is forwarded to the next step as input. You can say a product is entering the step, proceeded and leaves it as output in a changed way. In case of Software-Development the product is the requirement.

For example, in the Step “Writing functional Design” the output is a written document as a description for implementing a solution. Something was created which can be described as progress in the creation of the final output, which is the new software-version.

In the next Step “Revision functional Design” another designer checks the created document, giving his approval, sending it to the next step. In this step the output is the same, as in the step before, there was no progress in the product, towards the final output. But something else happened, the risk for a fault was decreased, increasing the quality of the output.

This example shows two different statuses, which both have an effect on the output, but in two different ways, two different dimensions. One influences the Quality, the other one the progress, leading to a basis for two-dimensional rating. These two steps are different. There are three different classifications for workflow steps:

Progress-Step: Steps with this classification are the most common steps. They modify the input towards the final outcome as output.

Quality-Gates: This classification has no impact on the progress towards the final outcome, but increases the quality of the product.

Clarification: This step usually happens outside the strict *workflow-steps.* If someone is irritated with the input of a step, as for example a developer has a problem understanding a design document as basis for the implementation, they normally write a mail or call the creator of the misunderstanding, asking for advice. The design document is not wrong and has not changed, it is just not detailed enough. Such a step is described as clarification. If the design document would be written in a proper way, there would be no room for a clarification. It can be described as identifying a quality problem and solving it outside the workflow. As in most workflows the clarification is a hidden step, as it happens outside the strict *workflow-steps*, the mission is to try to include these kind of steps in the workflow as well, so they can be measured as another *workflow-step* classification.

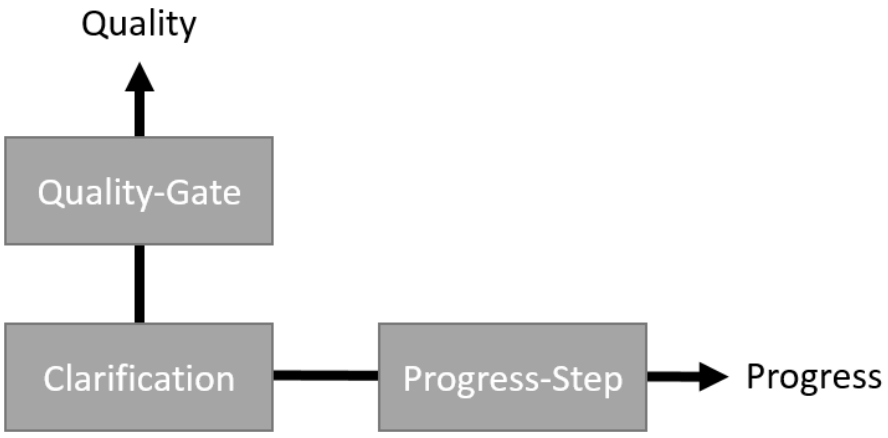

Figure 3.5: Workflow Step Classifications

Every existing workflow step can be marked as one of those three classifications. In the end every process has a certain number of "Progress-Steps", "Quality-Gates" and "Clarifications". The following ratio of classifications provides the management possible switches per module, how to improve them in case of problems. In chapter 5. the ratio of those classifications is compared to the quality of the outcome, showing a best-practice for the ratio.

3.2.2 Workflow Step-Change Classification

The classification of workflow steps is a static information, showing the management possible switches where to improve, according to the quality of the outcome. Additional to this static information, which does not change, the dynamic information which varies with the different usage, describes the movement between those steps. Starting from a step the Successor can be in four different directions. The rating of Step-Change is always according to the output of the Predecessor.

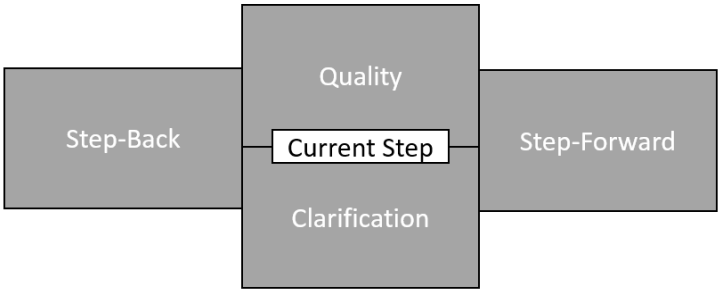

Figure 3.6: Workflow Step Change Classifications

Every of those steps is rated in a different way. At the end of each process the "costs", according the work can be calculated with two measurements: Efficiency (stands for the progress) and the Quality. The rating of each Step-Change is described in the detailed explanation of the different Step-Change Classifications.

Step-Forward: *Workflow-Step-Changes* of this classification have a positive impact on the progress of the outcome. The quality is not affected by this *workflow-step-change*.

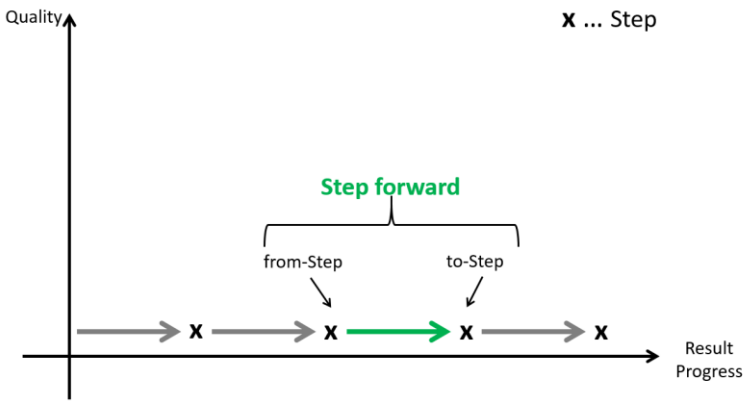

Figure 3.7: Workflow Step-Change Classification - Step-Forward

Quality: This type of Step-Change is always in charge after a quality-gate, independent from the Successor step. The progress is not effected, but the quality is raised.

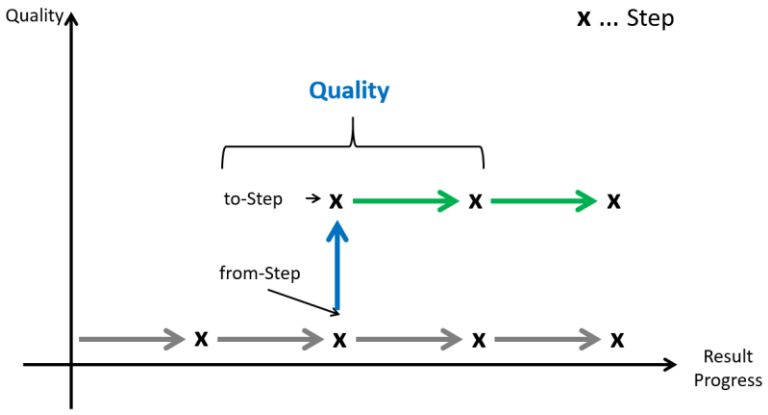

Figure 3.8: Workflow Step-Change Classification – Quality

Clarification: This *workflow-step-change* occurs before and after a "Clarification" step, as the Predecessor-Step has no valid Output due to the needed Clarification. This *workflow-step-change* is making a discovered quality-problem right without an effect on the progress.

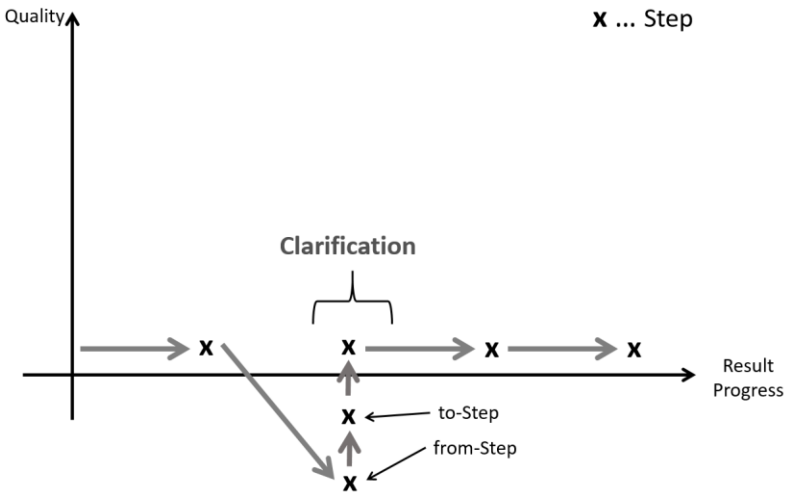

Figure 3.9: Workflow Step-Change Classification – Clarification

Step-Back: In case there is not just a clarification, but the output of the predecessor has to be recreated, a "Step-Back" *workflow-step-change* is in charge. This ping-pong between two steps is the worst *workflow-step-change* which can happen, leading to two *workflow-step-changes* with no positive effect on quality or progress.

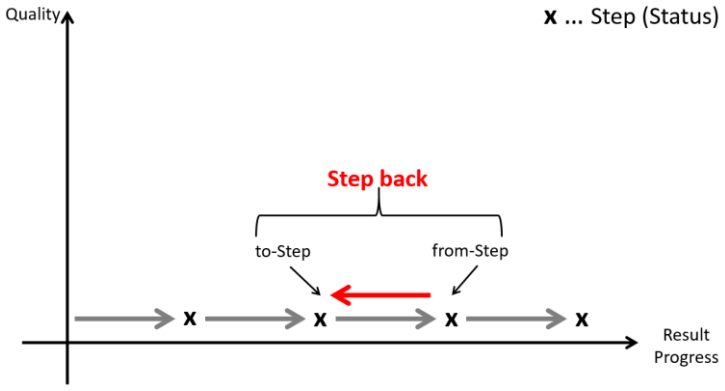

Figure 3.10: Workflow Step-Change Classification - Step-Back

3.2.3 Measurement Parameters

In this sub-chapter according the *workflow-step* classification and the *workflow-step-change* classification, all available needed parameters will be described as basis for the thereafter key figure creation.

3.2.3.1 Measurement on Module Level

According the *workflow-step* classification, following measurement parameters exist on module level.

Measurement parameters on module level	
Parameter Name	**Description**
sNWS **s**tatic – **N**umber of **W**orkflow-**S**teps	This static parameter provides a fixed number of existing *workflow-steps* inside a module.
sNWSmin **s**tatic – **N**umber of **W**orkflow-**S**teps **min**imal	This static parameter provides a fixed number, which is the sum of all *workflow-steps* needed to finish a module in the minimal time (the most efficient way to go through a module). If there are no splits in the workflow layout this number is equal the sNWS parameter.
sNWSmax **s**tatic – **N**umber of **W**orkflow-**S**teps **max**imal	This static parameter provides a fixed number, which is the sum of all *workflow-steps* needed to finish a module in the maximal time the most efficient way to go through a module. If there are no splits in the workflow layout this number is equal the sNWS parameter. If there are loops in the Workflow this parameter is infinite.
sNWS-Q **s**tatic – **N**umber of **W**orkflow-**S**teps **– Q**uality-Gate	This static parameter provides a fixed number of existing *workflow-steps* with the classification Quality-Gate inside a module.
sNWS-C **s**tatic – **N**umber of **W**orkflow-**S**teps **– C**larification	This static parameter provides a fixed number of existing *workflow-steps* with the classification Clarification inside a module.
sNWS-P **s**tatic – **N**umber of **W**orkflow-**S**teps **– P**rogress	This static parameter provides a fixed number of existing *workflow-steps* with the classification Progress inside a module.
dNWU **d**ynamic – **N**umber of **W**orkflow **U**sage	This dynamic parameter provides the increasing count of times a module was finished.
dNWS **d**ynamic – **N**umber of **W**orkflow-**S**teps	This dynamic parameter provides the total number of used *workflow-steps* after a module is finished.
dNWS-C **d**ynamic – **N**umber of **W**orkflow-**S**teps **– C**larification	This dynamic parameter provides the number of used *workflow-steps* with the classification Clarification during a module, after it is finished.

dNWS-F **d**ynamic – **N**umber of **W**orkflow-Steps **– P**rogress	This dynamic parameter provides the number of used *workflow-steps* with the classification Progress during a module, after it is finished.
dNWS-Q **d**ynamic – **N**umber of **W**orkflow-**S**teps **– Q**uality-Gate	This dynamic parameter provides the number of used *workflow-steps* with the classification Quality-Gate during a module, after it is finished.
sMNSC **s**tatic – **M**inimum **N**umber of **S**tep-**C**hanges	This static parameter provides a fixed number of existing *workflow-step-changes* inside a module.
dNSC **d**ynamic – **N**umber of **S**tep-**C**hanges	This dynamic parameter provides the total number of *workflow-step-changes* during a module, after it is finished.
dNSC-C **d**ynamic – **N**umber of **S**tep-Changes – **C**larification	This dynamic parameter provides the number of used *workflow-step-changes* with the classification Clarification during a module, after it is finished.
dNSC-F **d**ynamic – **N**umber of **S**tep-Changes – Step-**F**orward	This dynamic parameter provides the number of used *workflow-step-changes* with the classification Step-Forward during a module, after it is finished.
dNSC-Q **d**ynamic – **N**umber of **S**tep-Changes – **Q**uality Gate	This dynamic parameter provides the number of used *workflow-step-changes* with the classification Quality-Gate during a module, after it is finished.
dNSC-B **d**ynamic – **N**umber of **S**tep-Changes – Step-**B**ack	This dynamic parameter provides the number of used *workflow-step-changes* with the classification Step-Back during a module, after it is finished.
dTT **d**ynamic – **T**otal **T**ime	This dynamic parameter provides the total time used to finish a workflow.
dTT-C **d**ynamic – **T**otal **T**ime **– C**larification	This dynamic parameter provides the sum of time needed for all "Clarification" *workflow-steps* inside the module, after it is finished.
dTT-P **d**ynamic – **T**otal **T**ime **– P**rogress	This dynamic parameter provides the sum of time needed for all "Progress" *workflow-steps* inside the module, after it is finished.
dTT-Q **d**ynamic – **T**otal **T**ime **– Q**uality-Gate	This dynamic parameter provides the sum of time needed for all "Quality-Gate" *workflow-steps* inside the module, after it is finished.

Table 3.1: Step Classification Measurement Parameters – Module

These parameters are the basis for the further creation of the key figures in chapter 3.2.4. The combination of those 21 parameters makes it possible to measure the modules in many different ways, finding problems in efficiency and quality, but also making possible switches for optimizations visible.

3.2.3.2 Measurement on Process Level

As most of the precise measurement – including Quality-Gates – is set up on workflow level, every iteration of the process looks different. This fact is also based on the modular approach and the requirement's needs. This modular layout makes it hard to create measurement on process level. Most of the key figures are based on comparison on module level. The little usable information on process level is to identify which modules are obsolete and which are used most. Based on this information, priorities for optimizations can be created or barely used modules can be removed. For this following parameters are available.

Measurement parameters on Process Level	
Parameter Name	**Description**
sNM **s**tatic – **N**umber of **M**odules	Describes the static number of available modules due to the explanation in chapter 3.1.
dNM **d**ynamic – **N**umber of **M**odules	Describes the dynamic total number of finished modules.
dNFP **d**ynamic – **N**umber of **F**inished **P**rocesses	Describes the dynamic total number of iterations of the Process.
dPTT **d**ynamic – **P**rocess **T**otal **T**ime	Describes the needed time for the process.

Table 3.2: Step Classification Measurement Parameters – Process

Additional to these parameters, the modular parameters can be used for the whole process as well, to provide an overall summed overview, as described in chapter 3.2.4.2.

3.2.4 Measurement Key Figures

Using the parameters from the previous sub-chapter, following meaningful key figures should be provided.

3.2.4.1 Key Figures on Module Level

Most of the measurements will be done on module level. To make the overview easier various figure-groups are established using various parameters. There are two classifications of key figures.

The first one represented by the figure-group "Static Workflow-Layout Key Figures" measures the current layout of the workflow, without any dynamic data. It should support the management in finding the right triggers in case the dynamic analysis of the quality and efficiency of the usage of the workflows leads to an unsatisfying result. The second area, the dynamic measurement of the workflows is represented by the other figure-groups "Dynamic General Efficiency Key Figures" and "*Workflow-Step* Classification Specific Key Figures".

Static Workflow-Layout Key Figures

These key figures should describe the current layout of the workflow. They are static, supporting the management in finding the right switches to modify the layout due to the outcome of the dynamic key figures. Each of these key figures should be available for every single requirement up to the overall amount of usages, depending on the need of the analysis.

Static Workflow-Level Key Figures		
Name of Key Figure	**Equation**	**Description**
sML-nWS **s**tatic – **M**odule **L**ayout Key Figures **n**umber – **W**orkflow-**S**teps	sNWS	To give the management a better understanding for the scale of following ratios, the total number of all available *workflow-steps* should be displayed as well. **0 < x <= ∞**
sML-rWS-C **s**tatic – **M**odule **L**ayout Key Figures **r**atio –**W**orkflow-**S**teps **– C**larification	sNWS-C / sNWS	This ratio provides the key figure of how big the amount of Clarification *workflow-steps* is inside the Workflow-Layout. **0% <= x <= 100%**

sML-rWS-P **s**tatic – **M**odule **L**ayout Key Figures **r**atio –**W**orkflow-**S**teps **– P**rogress	sNWS-P / sNWS	This ratio provides the key figure of how big the amount of Progress *workflow-steps* is inside the Workflow-Layout. **0% <= x <= 100%**
sML-rWS-Q **s**tatic – **M**odule **L**ayout Key Figures **r**atio –**W**orkflow-**S**teps **– Q**uality-Gate	sNWS-Q / sNWS	This ratio provides the key figure of how big the amount of Quality-Gate *workflow-steps* is inside the Workflow-Layout. **0% <= x <= 100%**
sML-rMinP **s**tatic – **M**odule **L**ayout Key Figures **r**atio – used Workflow-Steps for **Min**imal **P**ath	sNWSmin / sNWS	This ratio shows how much of the whole workflow-layout is used in case of the most efficient path. The maximum value is 100%, in case there are no splits in the workflow-layout **0% < x <= 100%**
sML-rMaxP **s**tatic – **M**odule **L**ayout Key Figures **r**atio – used Workflow-Steps for **Max**imal **P**ath	sNWSmax / sNWS	This ratio shows how much of the whole workflow-layout is used in case of the least efficient path. The minimal value is 100%, in case there are no splits in the workflow-layout. In case there are loops this ratio can go up to infinite. **100% <= x <= ∞%**
sML-rPD **s**tatic – **M**odule **L**ayout Key Figures **r**atio – **P**ath-**D**ifference	sML-rMaxP – sML-rMinP	This ratio shows the difference between the two key figures sML-rMaxP and sML-rMinP. It is an indicator of the unpredictable variance of the final length of each individual workflow. The smaller this ratio is, the more predictable the future time used in this workflow (for measurements) can be. In a fix workflow without split the used steps are totally predictable, leading to a value of 0. If the maximum path contains loops this number can go up to infinite. **0% <= x <= ∞%**

Table 3.3: Step Classification Measurement Key Figures – Module (static Workflow Level)

Dynamic General Efficiency Key Figures

This key figures stand for a general overview of the efficiency, which will be followed by a more detailed overview for each *workflow-step* classification in the two following figure-group "*Workflow-Step* Specific Key Figures".

Dynamic General Efficiency Key Figures		
Name of Key Figure	**Equation**	**Description**
dGE-rASU **d**ynamic – **G**eneral **E**fficiency Key Figures **r**atio – **A**verage **S**tep **U**se	dNWS / dNWU	This ratio shows the average use of *workflow-steps* until a workflow is finished. If it is far below 100% it can be an indicator, to show that parts of the workflows are not used anymore. If its above 100% it means, that existing loops are often used, identifying a problem in the quality of the work. **0% < x <= ∞%**
dGE-tATU **d**ynamic – **G**eneral **E**fficiency Key Figures **t**ime – **A**verage **T**ime **U**sed	dTT / dNWU	This duration shows the average use of time to finish the workflow. It can be used to set new goals and to measure the efficiency of optimizations.
dGE-tATU-C **d**ynamic – **G**eneral **E**fficiency Key Figures **t**ime – **A**verage **T**ime **U**sed – **C**larification	dTT-C / dNWU	This duration shows the average use of time to finish a step with the classification “Clarification”. **0 <= x <= ∞**
dGE-tATU-P **d**ynamic – **G**eneral **E**fficiency Key Figures **t**ime – **A**verage **T**ime **U**sed – **P**rogress	dTT-P / dNWU	This duration shows the average use of time to finish a step with the classification “Progress”. **0 <= x <= ∞**
dGE-tATU-Q **d**ynamic – **G**eneral **E**fficiency Key Figures **t**ime – **A**verage **T**ime **U**sed – **Q**uality-Gate	dTT-Q / dNWU	This duration shows the average use of time to finish a step with the classification “Quality-Gate”. **0 <= x <= ∞**
dGE-rSC-C **d**ynamic – **G**eneral **E**fficiency Key Figures **r**atio – **S**tep-**C**hange – **C**larification	dNSC-C / dNSC	This ratio shows the number of “Clarification” *workflow-step-changes* needed in average to finish a workflow compared to the total average usage of needed workflow-steps. It identifies a quality problem, as the value should be 0. **0% <= x <= 100%**

dGE-rSC-F **d**ynamic – **G**eneral **E**fficiency Key Figures **r**atio – **S**tep-**C**hange – Step-**F**orward	(dNSC-F - dNSC-B) / dNSC	This ratio shows the number of "Step-Forward" *workflow-step-changes* needed in average to finish a workflow compared to the total average usage of needed workflow-steps. In a totally effective process, without Quality-Gates it would be 100%. The equation deducts the number of Step-Back classifications from the Step-Forward Classification, as every Predecessor of a Step-Back is a Step-Forward, without any progress, as discovered afterwards leading to the Step-Back. **0% <= x <= 100%**
dGE-rSC-Q **d**ynamic – **G**eneral **E**fficiency Key Figures **r**atio – **S**tep-**C**hange – **Q**uality	dNSC-Q / dNSC	This ratio shows the number of "Quality" *workflow-step-changes* needed in average to finish a workflow compared to the total average usage of needed workflow-steps. **0% <= x <= 100%**
dGE-rSC-B **d**ynamic – **G**eneral **E**fficiency Key Figures **r**atio – **S**tep-**C**hange – Step-**B**ack	2 * dNSC-B / dNSC	This ratio shows the number of "Step-Back" *workflow-step-changes* needed in average to finish a workflow compared to the total average usage of needed workflow-steps. This value is toxic und should be 0, otherwise a quality problem is identified. The number of "Step-Back" classification is doubled, as every predecessor Step-Change was faulty, leading to the "Step-Back" and will also be classified as such (therefore the number for "Step-Forward" classifications is deducted by the original number of "Step-Backs"). **0% <= x < 100%**

Table 3.4: Step Classification Measurement Key Figures – Module (dynamic General Efficiency)

Workflow-Step Classification Specific Key Figures

As described in chapter 3.2.1 there are three *Workflow-Step* classifications: C – "Clarification", P – "Progress" and Q – "Quality-Gate". For each of those classifications the following key figures should be created. Additional to the generic description in Table 3.5, every key figure contains an example on how the name and equation would look like for the *workflow-step* "Clarification".

Clarification / Progress / Quality – Step Key Figures		
Name of Key Figure	**Equation**	**Description**
dC-<CLASSIFICATION>-tAD (e.g. dC-C-tAD) **d**ynamic – <CLASSIFICATION> **t**ime – **A**verage **D**uration	dTT-<CLASSIFICATION> / dNWS-<ClASSIFICATION> e.g.: dTT-C / dNWS-C	This duration shows the average time needed for a step of this classification. **0 <= x <= ∞**
dC-<CLASSIFICATION>-rAD (e.g. dC-C-rAD) **d**ynamic – <CLASSIFICATION> **r**atio – **A**verage **D**uration	(dTT-<CLASSIFICATION> * dNWS) / (dTT * dNWS-<ClASSIFICATION>) e.g.: (dTT-C * dNWS) / (dTT * dNWS-C)	This ratio shows the relation between the average used time for a step of this classification due to the average used time per every kind of workflow-steps. If it is below 100% it means this *workflow-step* classification is already more productive than the average. If it is above 100% it is an indicator for possible improvements. **0% <= x <= ∞%**
dC-<CLASSIFICATION>-rM-C **(e.g. dC-C-rM-C)** **d**ynamic – <CLASSIFICATION> **r**atio – **M**ovement **– C**larification	$dNSC\text{-}C_{<CLASSIFICATION>}$ / $dNSC_{<CLASSIFICATOIN>}$ e.g.: $dNSC\text{-}C_C$ / $dNSC_C$	This ratio shows how many of the *workflow-step-changes* away from this kind of *workflow-step* go into the direction of the *workflow-step-change* classification “Clarification”. **0% <= x <= 100%**
dC-<CLASSIFICATION>-rM-F **(e.g. dC-C-rM-F)** **d**ynamic – <CLASSIFICATION> **r**atio – **M**ovement **–** Step-**F**orward	$dNSC\text{-}F_{<CLASSIFICATION>}$ / $dNSF_{<CLASSIFICATOIN>}$ e.g.: $dNSC\text{-}F_C$ / $dNSC_C$	This ratio shows how many of the *workflow-step-changes* away from this kind of *workflow-step* go into the direction of the *workflow-step-change* classification “Step-Forward”. **0% <= x <= 100%**
dC-<CLASSIFICATION>-rM-Q **(e.g. dC-C-rM-Q)** **d**ynamic – <CLASSIFICATION> **r**atio – **M**ovement **– Q**uality	$dNSC\text{-}Q_{<CLASSIFICATION>}$ / $dNSC_{<CLASSIFICATOIN>}$ e.g.: $dNSC\text{-}Q_C$ / $dNSC_C$	This ratio shows how many of the *workflow-step-changes* away from this kind of *workflow-step* go into the direction of the *workflow-step-change* classification “Quality-Gate”. **0% <= x <= 100%**
dC-<CLASSIFICATION>-rM-B **(e.g. dC-C-rM-B)** **d**ynamic – <CLASSIFICATION> **r**atio – **M**ovement **–** Step-**B**ack	$dNSC\text{-}B_{<CLASSIFICATION>}$ / $dNSC_{<CLASSIFICATOIN>}$ e.g.: $dNSC\text{-}B_C$ / $dNSC_C$	This ratio shows how many of the *workflow-step-changes* away from this kind of *workflow-step* go into the direction of the *workflow-step-change* classification “Step-Back”. **0% <= x <= 100%**

Table 3.5: Step Classification Measurement Key Figures – Module (Workflow-Step Classification Specific)

3.2.4.2 Key Figures on Process Level

As most of the precise measurement is done on module level, the process measurement shows an overview to identify priorities for improvement due to usage of modules. Additional to this comparison of the different modules the process key figures contain some summed up information, which cannot be provided by each individual module.

General Process Key Figures

This key figures show general information due to the use of the process, which cannot be measured on module level. For example, the time needed to finish a whole process iteration, which can be described as the sum of all used modules, additional to the waiting time in between.

General Process Key Figures		
Name of Key Figure	**Equation**	**Description**
sGP-nTM **s**tatic – **G**eneral **P**rocess **n**umber – **T**otal **M**odules	sNM	This figure shows the total number of available modules to support the management to put the following ratios and figures in relation. **0 < x <= ∞**
dGP-nAMI **d**ynamic – **G**eneral **P**rocess **n**umber – **A**verage **M**odules per **I**teration	dNM / dFP	This figure shows the average number of modules used per iteration. **0 < x <= ∞**
dGP-tATI **d**ynamic – **G**eneral **P**rocess **t**ime – **A**verage **T**ime per **I**teration	dPTT / dFP	This figure shows the average time an iteration of the process needs until the process is finished. This figure additional to the key figure dGP-nAMI are the roughest levels of indicators for efficiency and quality. **0 < x <= ∞**

Table 3.6: Step Classification Measurement key figures – Process (General)

Process Overview Key Figures

This key figures show the summed modular level key figures for the most important areas. This summed view should create an additional high-level overview upon the most important workflow key figures. This summed key figures are also a guideline of how to balance the ratios of the *workflow-steps* inside a new module, or in which direction to develop an already existing module, in case it should be changed.

Process Overview Key Figures		
Name of Key Figure	**Equation**	**Description**
sPO-nWS **s**tatic – **P**rocess **O**verview Key Figures **n**umber – **W**orkflow-**S**teps	sNWS	This figure shows the total number of available Workflow-Steps in all modules. **0 < x <= ∞**
sPO-rWS-C **s**tatic – **P**rocess **O**verview Key Figures **r**atio –**W**orkflow-**S**teps **– C**larification	sNWS-C / sNWS	This ratio shows how many Clarification Workflow-Steps are inside the process. **0% <= x <= 100%**
sPO -rWS-P **s**tatic – **P**rocess **O**verview Key Figures **r**atio –**W**orkflow-**S**teps **– P**rogress	sNWS-P / sNWS	This ratio shows how many Progress Workflow-Steps are inside the process. **0% <= x <= 100%**
sPO -rWS-Q **s**tatic – **P**rocess **O**verview Key Figures **r**atio –**W**orkflow-**S**teps **– Q**uality-Gate	sNWS-Q / sNWS	This ratio shows how many Quality-Gate Workflow-Steps are inside the process. **0% <= x <= 100%**

Table 3.7: Step Classification Measurement Key Figures – Process (General Overview)

Additional to these summed key figures there should be some ratios for each existing module. The following KPIs compare the module itself with the overview value of the whole process providing the management the information about the importance of each module.

Process Overview per Module Key Figures		
Name of Key Figures	**Equation**	**Description**
sPOM$_{<MODULE>}$-rMU **s**tatic – **P**rocess **O**verview per **M**odule **r**atio – **M**odule **U**sage	dNWU$_{<MODULE>}$ / dNFP	This figure shows the ratio on how often the specific module is used, compared to all used modules. **0% <= x <= 100%**

sPOM$_{<MODULE>}$-rTU **s**tatic – **P**rocess **O**verview per **M**odule **r**atio – **T**ime **U**sage	dTT$_{<MODULE>}$ / dPTT	This figure shows the ratio on how much time this module takes compared to all used modules. If this ratio is comparable with the key figure sPOM$_{<MODULE>}$-rMU this is an indicator for a possible problem. **0% <= x <= 100%**

Table 3.8: Step Classification Measurement Key Figures - Process (Module Overview)

3.2.5 Measurement KPIs

Additional to the Key Figures described in chapter 3.2.4, standardized KPIs should be used to create a complete overview of the status of a process as basis for continuous improvement.

There are multiple views from different perspectives to look at a process and measure it. It could be from a financial perspective, a customer perspective or an internal perspective. As this thesis concentrates on the optimization of a process from the inside, the internal KPIs will be taken into account. For this reason, M. Steiner created more than 50 KPI characteristics which were based on the findings and definitions out of CMMI or ISO/UEC 15939. To identify which of those characteristics are the most important in real projects, he created a survey asking the opinion of a group of experts. The characteristics of three of the highest rated process KPI characteristics are used in this thesis on top of the predefined Key Figures. The following Sub-Chapters describe the used KPIs.

[cf., (Steiner, 2015)]

3.2.5.1 As-Is / To-Be Deviation

Strategical Goal: Raise the efficiency and effectivity of a Process
Purpose: Control the compliance based on effort
Unit: Person Days
Formula: As-Is-Expenses – Planned-Expenses

[cf., (Steiner, 2015)]

This KPI gives the management the chance to rate the Process outside the Workflow, using the data out of the employee-time-booking system. This figures can be compared with the defined Key Figures to provide an additional indicator for problems.

3.2.5.2 Test Efficiency

Strategical Goal: Raise the efficiency and effectivity of a Process
Purpose: Steer the test-activity during the software-development-live-cycle
Unit: Percent
Formula: Number-of-production-defects / (Number-of-production-defects – Rejected-defects-in-production)

[cf., (Steiner, 2015)]

This KPI indicates the quality of the outcome. Not every logged defect has to be a real defect, so the rejection-rate is also taken into account. It actually indicates the quality of testing from the customer. Not always the problem has to be on the project-side. While 100% describe the perfect situation in which every logged defect is really a defect, every percent more shows problems on customer side. E.g. 200% already means that every second error could be a wrong call from the customer.

3.2.5.3 Software-Defect Ratio

Strategical Goal: Raise the efficiency and effectivity of a Process
Purpose: Measure quality of software-product based on defects which occurred per unit of the Software
Unit: Defect-Unit
Formula: Number-of-identified-defects / Size-of-software

[cf., (Steiner, 2015)]

The unit “Size-of-software” can be described as total number of requirements. This KPI gives a better understanding of the number of defects. If 100 Defects occur, it sounds a lot in the first moment. But if this number is logged in an environment with multiple thousands of finished requirements it is very fast relativized. This figure prevents from misjudgment of the absolute number of defects.

4. Process-Optimization Implementation

Based on the findings out of chapter 2. the optimization strategy was created in chapter 3. In this chapter based on both chapters the example process is analyzed and modified using the optimization approaches from the standard framework and the aligned approach.

For this, one process from the example project is chosen and the starting situation is analyzed. Based on this as-is situation, the points for optimization are identified leading to the creation of the to-be process.

In the last part of this chapter, the defined measurement-points to analyze the impact of the changes with the starting-situation are defined. In the following chapter 5. a strategy of how to compare the values with the historical data based on this measurement-points is created. Build on this strategy the new process is compared with the historical data of the old layout and the efficiency of the new approach is evaluated.

4.1 Optimization Starting Situation

First, the whole example process is compared with the best-practices from the frameworks CMMI and SPICE. It is described which processes are missing on which position in a general way. Afterwards, the current process layout is modified due to the findings out of the standardized processes and the modular approach from chapter 3.1.

4.1.1 Comparison of example process with standardized Frameworks

If the processes from the example project are compared with the findings and the important points out of the optimization strategy, based on the process frameworks in chapter 2.4.2 only, most of the processes already exist as equivalent to the best-practice of the standards as shown in Figure 4.1.

Current Process Setup		SPICE	CMMI
Demand Management	Identification of Requirements	**ENG.1** Requirements elicitation	**RD SG 1** Develop Customer Requirements
	Refinement / Validation of Requirements	**ENG.2** System requirements analysis	**RD SG 3** Analyze and Validate Requirements
	Identification of efforts		
	Prioritization of Requirements		
Solution Conception and Design	Definition of Solution Outline		
	Creation of Functional Design	**ENG.5** Software design	
	Assurance Quality and Handover	**SUP.1** Quality assurance	
Configuration and Build	Creation of technical Design	**ENG.6** Software Construction	
	Configuration of Software		
	Customization of Software		
Test	Perform Software Test	**ENG.8** Software testing	
	Perform System Test	**ENG.9** System itegration test	
		ENG.10 System testing	
	Perform functional Integration Test		
	Perform non functional Integration Test		
	Perform UAT		
Rollout	Manage Go Live		
	Guide Handover		

Figure 4.1: Process-Optimization - Comparison of Processes

In Figure 4.1 the processes were mapped on a gross scale due to their definitions. Not all framework processes could be mapped leading to following processes which need to be modified.

4.1.1.1 Missing CMMI Process Templates for Main Processes

Due to the finding described in chapter 2.4.3.1 following CMMI process templates are missing and the position they should be added. All unmentioned processes are used outside the main processes to build accompany processes in chapter 4.3.

REQM SG 1 Manage Requirements

- SP 1.5 Identify Inconsistencies Between Project Work and Requirements

This process should be added in the example process “Demand Management” after “Refinement / Validation of Requirements” and “Identification of efforts”.

VER SG 2 Perform Peer Reviews

- SP 2.1 Perform Peer Reviews

This verification process is a quality-gate which can be used at many positions during the process and is not relying on a particular position of the lifecycle.

VER SG 3 Verify Selected Work Products

- SP 3.1 Perform Verification
- SP 3.2 Analyze Verification Results

This verification process is a quality-gate which can be used at many positions during the process and is not relying on a particular position of the lifecycle.

VAL SG 1 Prepare for Validation

- SP 1.1 Establish Validation Procedures and Criteria

This validation process is a quality-gate which can be used at many positions during the process and is not relying on a particular position of the lifecycle.

4.1.1.2 Missing SPICE Process Templates for Main Processes

Also from the SPICE framework some processes could not be mapped, which should be included due to chapter 2.4.3.2. All unmentioned processes are used outside the main processes to build accompany processes in chapter 4.3.

ENG.3 System architectural Design: This process can be compared to the missing CMMI process REQM SG 1 and should also be included in the process “Demand Management” between “Refinement / Validation of Requirements” and “Identification of efforts”.

ENG.4 Software requirements analysis: This process is the successor of ENG.3 and should be right afterwards in the process “Demand Management”, also before “Identification of efforts”.

ENG.7 Software integration Test: This process is missing in the example project process “Test” as first sub-process.

SUP.2 Verification: This process is not depending on a certain position in the process like the CMMI process VER SG 2 and can be used on multiple various locations.

SUP.9 Problem resolution management: This process should prevent problems before they occur as a last quality check. Therefore, it should be implemented before testing, but after implementation as last step of the process "Configuration and Build".

4.1.2 Description of Example Process

To analyze the impact of the optimizations from the standardized frameworks and the modular approach, a part of the example project processes will be modified as basis for measureable data. To this end, parts of the process "Solution Conception and Design" (as described in chapter 1.3.1) will be optimized and modified.

4.1.2.1 Process Solution Conception and Design – Definition of Solution Outline

This process starts directly after the last step of the "Demand Management" process, which is "Prioritization of Requirements".

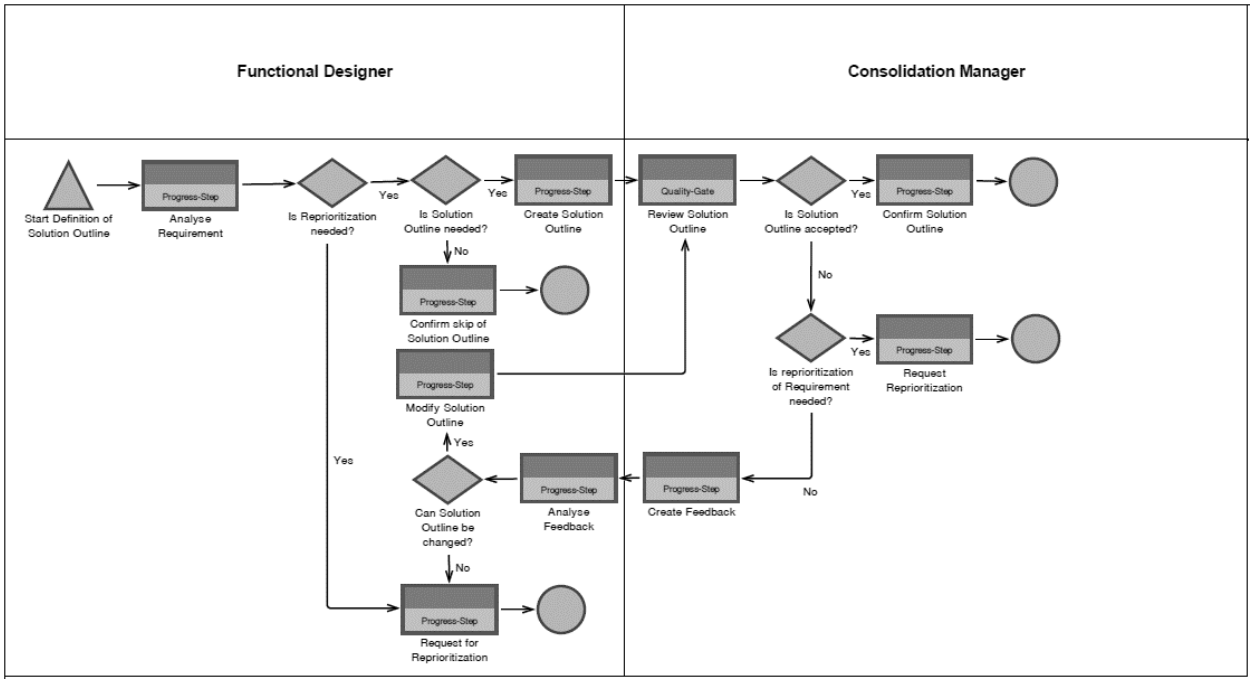

Figure 4.2: Definition of Solution Outline - Workflow Layout

Description

As described in Figure 4.2, this workflow contains the two roles “Functional Designer” and “Consolidation Manager”. At the beginning the functional designer checks the requirement for which he needs to create a solution outline.

It could be the case, that the predecessor-process “Prioritization of Requirements” is already long done, and at the start of the process it is already clear that the circumstances have changed, so the designer can send the requirement back to prioritization right at the start of the process. Also based on the analysis of the requirement, the designer decides, if a solution outline is needed at all, giving him the chance to skip this process.

If both of those possibilities are not in charge, the solution outline is finally created by the functional designer and reviewed by the consolidation manager. If the consolidation manager is satisfied with the proposed solution, he confirms the outline and ending the process. If he identifies any problems, he can send the requirement back to reprioritization or write a feedback for the functional designer as basis for a modification of the outline.

Based on this feedback, the functional designer has to decide if he is able to modify the solution outline, without harming the scope of the requirement, or if a reprioritization is needed.

The following Table 4.1 shows the classification of the *workflow-steps* from the process shown in Figure 4.2.

Measurement	Value
Workflow-Steps	10
Clarification	0
Progress-Steps	9
Quality-Gates	1

Table 4.1: Definition of Solution Outline - Key Figures

Identified Problems

1. **Process should be modified due to modular approach:** Directly at the start of the process there is the question “Is Solution Outline needed?”. In case the answer is no, the whole process would be obsolete for this particular requirement, wasting time by the hardwired layout. Due to the modular approach, the designer (probably the same person creating the analysis for the requirement) can already judge if a solution outline is needed. Based on this information, the whole process “Definition of Solution Outline” could be skipped in this case.

2. **Responsibilities should be clarified:** There are two involved parties in this process, both having the possibility of finishing the process, once even for the same reason (reprioritization needed). The responsibilities are not clear, which is a problem according the findings from the CMMI and SPICE analysis. In this case just the consolidation manager should be able to end the process, either by confirming the solution outline or triggering a reprioritization.

3. **Clarifications should be added:** The last identified problem lies within the communication between both parties. Both have the chance to trigger a reprioritization without even involving the other party in their decision. Most probably there is a chat between both, but outside this workflow. A perfect example for a hidden-clarification. This discussion should be added to the process, to be monitored as well and also questioning the feedback from the consolidation manager, as he could also make a mistake, questioning the solution outline.

4.1.2.2 Process Solution Conception and Design – Creation of functional Design

This sub-process is the direct successor of “Definition Solution Outline”.

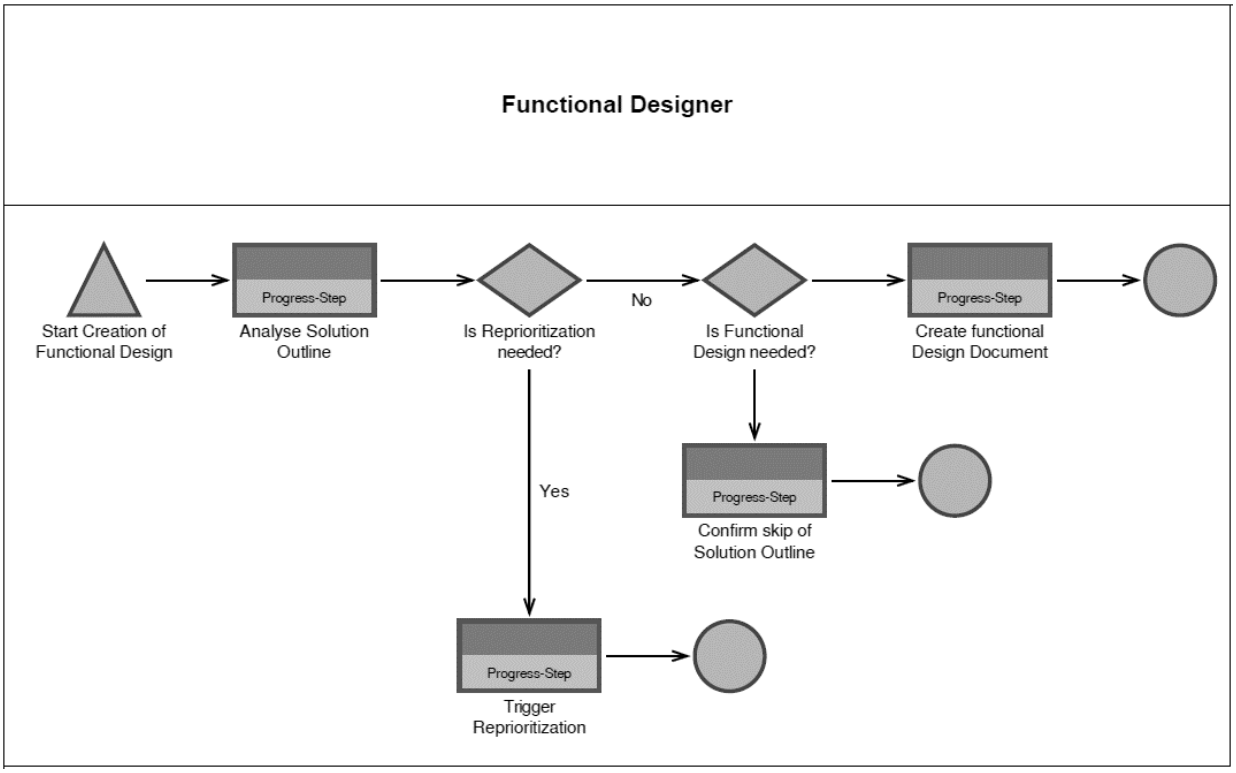

Figure 4.3: Creation of functional Design - Workflow Layout

Description

As described in Figure 4.3, this process only contains the role “Functional Designer”. First, the designer analyses the solution outline, which is the basis for the creation of the functional design document. It could happen, due to scheduling problems, that already at the start of the process the deadlines cannot be kept, leading to a reprioritization directly at the beginning.

It is also possible, that the solution outline only suggests a configuration, which makes the whole "Creation of functional design" process obsolete. If this is the case the designer just confirms the skip of the process, ending the process.

If a functional design is really needed, the designer writes the document as basis for the next process, finishing "Creation of functional Design".

Measurement	Value
Workflow-Steps	4
Clarification	0
Progress-Steps	4
Quality-Gates	0

Table 4.2: Creation of functional Design - Key Figures

Identified Problems

1. **Process should be modified due to modular approach:** Like the process before "Definition of Solution Outline", this process also wastes time at the beginning to identify if the process is actually needed. This information is already available at the creation of the solution outline, which could lead to skip this whole process.
2. **Peer Review:** After the creation of the functional design document there is no review cycle. Therefore, a solution based on the process VER SG 2 Perform Peer Review from CMMI as identified before should be implemented.

4.1.2.3 Process Solution Conception and Design – Assurance Quality and Handover

This process is the last sub-process in "Solution Creation and Design" before the successor process "Configuration and Build".

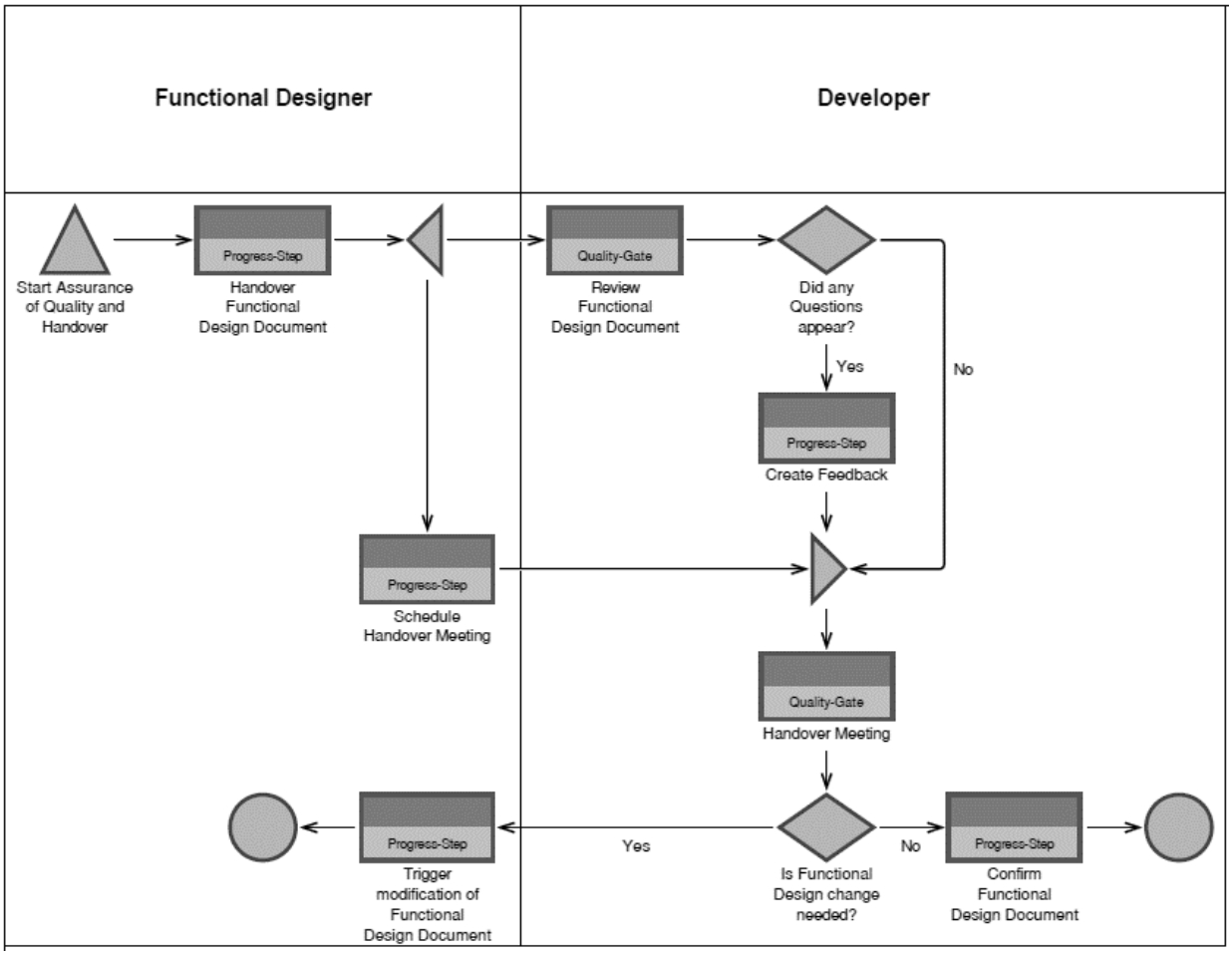

Figure 4.4: Assurance Quality and Handover - Workflow Layout

Description

As described in Figure 4.4, two stakeholder-groups “Functional Designer” and “Developer” are involved. This sub-process is the last step in the process “Solution Conception and Design”, in which the functional designers are mainly in charge. In the next process “Configuration and Build” the developers are mainly in charge, based on the output of “Solution Conception and Design”.

For this reason, this last sub-process concentrates on the handover of the output – the functional design document – from designer to developer. Therefore, the designer hands over the design document for a revision by the developer. Afterwards, the document is always discussed in a handover-meeting between both parties. If any problems were identified the functional designer would request another sub-process of the process “Solution Conception

and Design". If the developer is satisfied he will confirm the design document, finally finishing the process "Solution Conception and Design".

Measurement	Value
Workflow-Steps	7
Clarification	0
Progress-Steps	5
Quality-Gates	2

Table 4.3: Assurance Quality and Handover – Key Figures

Identified Problems

1. **Setup Finishing Module due to Modular Approach:** This process is already almost a perfect finishing-module due to the modular approach. The only thing missing is an additional step which checks, if the modular approach was misused before, which is mandatory for a finishing-module.
2. **Make Process Outcome more dynamic due to Modular Approach:** As it is possible, that only a small configuration or a creation of a detailed description is needed to fulfill a requirement. In such a case, the whole successor process "Configuration and Build" would be obsolete. The process should also question, if development's participation is needed at all. If it is not needed, the whole handover, as basis for the next process "Configuration and Build" should be skipped.

4.2 Optimized Process

Based on the identified problems in the sub-processes of the example process "Solution Conception and Design" and due to the findings out of SPICE and CMMI as described in chapter 4.1.1 and the modular approach described in chapter 3.1 the processes should be optimized.

4.2.1 Optimization Process Solution Conception and Design

The whole process is modified due to the needs of the modular approach. Additional to these modifications, the optimization approaches from the standardized frameworks out of chapter

4.1.1 are implemented in the process layout. The processes “Definition of Solution Outline” and “Creation of Functional Design” stay as optional-modules. The final sub-process “Assurance Quality and Handover” will represent the finishing-module.

4.2.1.1 Optimization Sub-Process Definition of Solution Outline

The process is optimized in Figure 4.5 due to the investigated problems described in chapter 4.1.2.1.

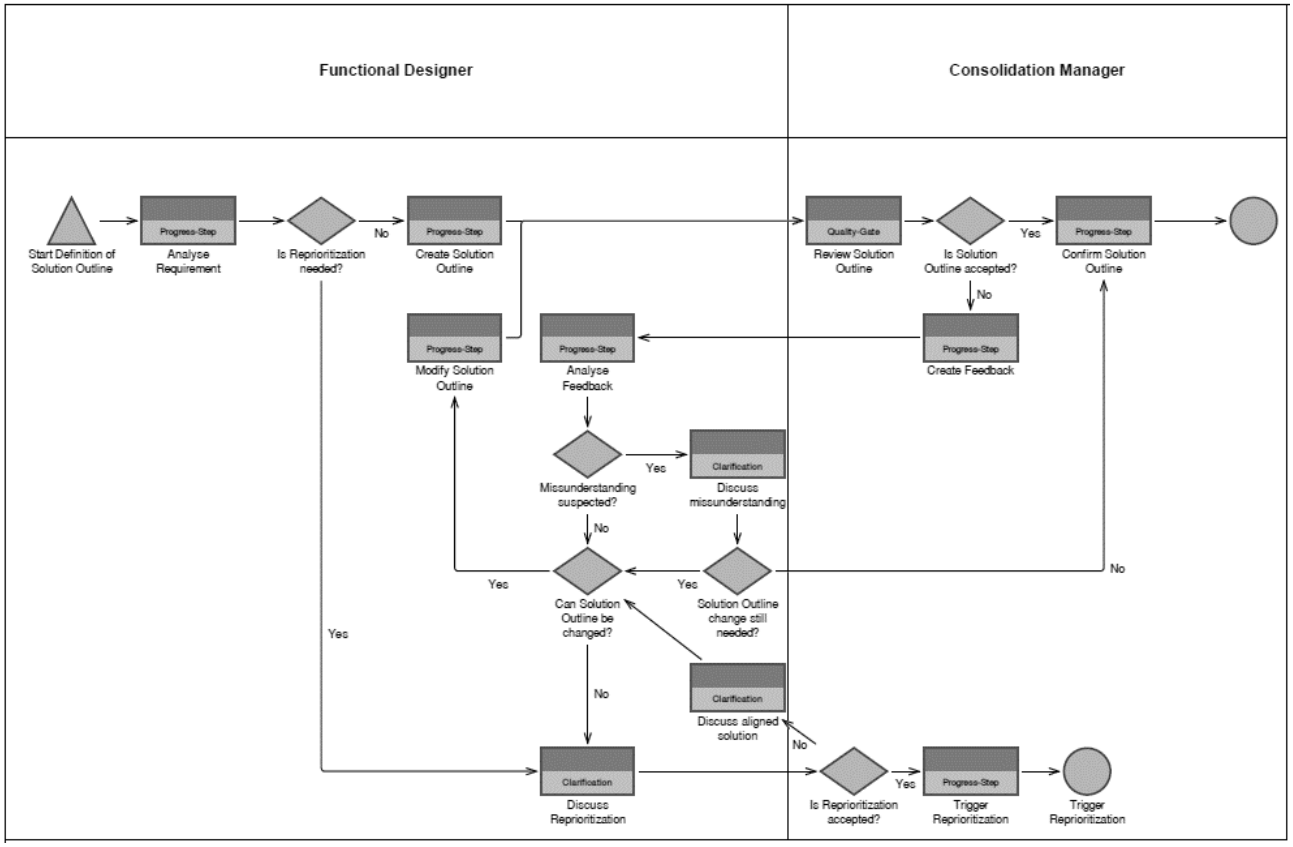

Figure 4.5: Definition of Solution Outline – Optimized Workflow Layout

The following list shows the problems identified in chapter 4.1.2.1 and a description how they were optimized in Figure 4.5.

1. **Process should be modified due to modular approach:** The working-steps responsible to identify the need of this sub-process are removed. In case this sub-process (which now is a module) is not needed, it would be skipped. In the first step "Analyze Requirement" the functional Designer also needs to check for possible misuse. In case of misuse, the work is send back to the predecessor process.
2. **Responsibilities should be clarified:** Out of the two parties in this process the consolidation manager is the only remaining who can end the process now. He is responsible for the quality-review and approves which further steps to take.
3. **Clarifications should be added:** In case a problem is discovered in a quality gate, instead of a possible hidden-clarification the problem should be discussed before taking next measures. Therefore, there is always a discussion step between the creator and the reviewer after a negative review.

In the following Table 4.4, the *workflow-steps* of the original workflow and the optimized layout are compared. As the beginning-part with the check for the need of the whole process is removed due to goal 1, the number of Progress-steps decreased. Three clarifications were introduced. The number of Quality-Gates stayed the same. In total due to the new clarifications the number of steps increased by one.

Measurement	Value before Optimization	Value after Optimization
Workflow-Steps	10	11
Clarification	0	3
Progress-Steps	9	7
Quality-Gates	1	1

Table 4.4: Definition of Solution Outline – Comparison of Key Figures

4.2.1.2 Optimization Sub-Process Creation of functional Design

The process is optimized in Figure 4.6 due to the investigated problems described in chapter 4.1.2.2.

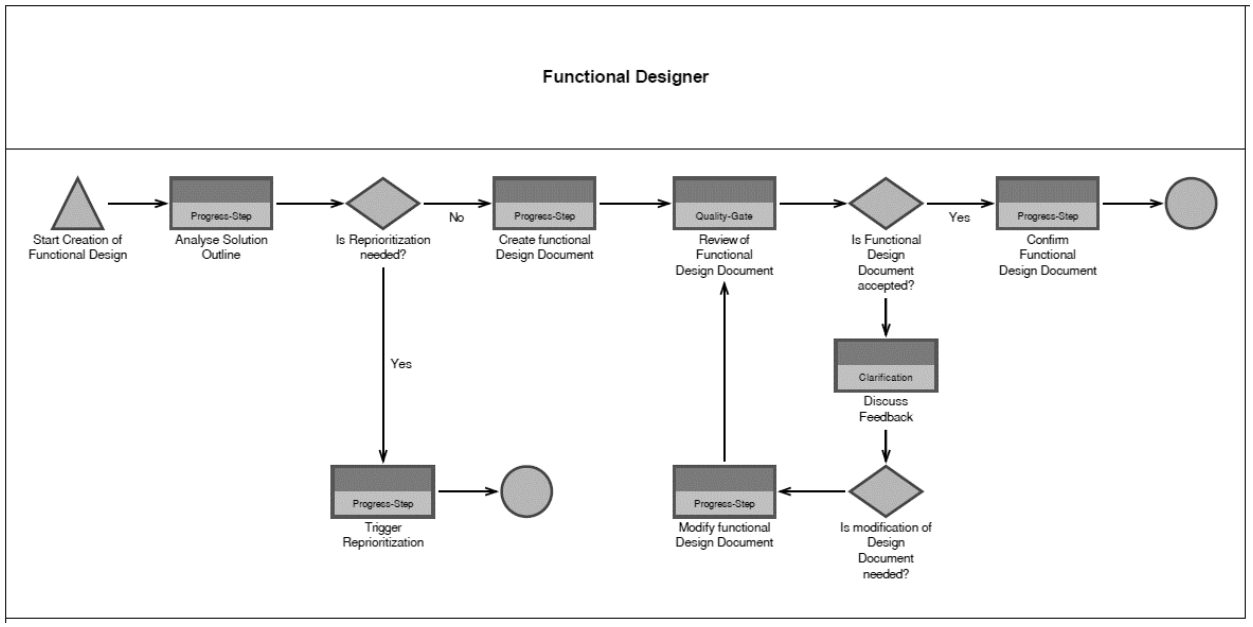

Figure 4.6: Creation of functional Design – Optimized Workflow Layout

The following list shows the identified problems out of chapter 4.1.2.2 and a description of how they were optimized in Figure 4.6.

1. **Process should be modified due to modular approach:** Due to the modular approach, the first question in the original workflow, if this step is needed or not, could be removed. If this sub-process is not needed, it will not be used in the future.
2. **Peer Review:** After the creation of the functional design document due to the findings out of CMMI and SPICE, a peer review was missing. In such a critical document even inside a team a review should be made by a colleague. If the colleague identifies a problem, it should be discussed in a clarification step and be solved.

In the following Table 4.5, the *workflow-steps* of the original workflow and the optimized layout are compared. The reprioritization question stays as before, as there is a possibility that during the first step the functional designer already identifies that the deadline cannot be fulfilled.

After the creation, a review routine was implemented, leading to a new quality-gate and a new clarification workflow-step. This new routine extends the number of the workflow-steps in the workflow layout by two, compared to the original workflow layout.

Measurement	Value before Optimization	Value after Optimization
Workflow-Steps	4	6
Clarification	0	1
Progress-Steps	4	5
Quality-Gates	0	1

Table 4.5: Creation of functional Design – Comparison of Key Figures

4.2.1.3 Optimization Sub-Process Assurance Quality and Handover

The process is optimized in Figure 4.7 due to the described problems investigated in chapter 4.1.2.3.

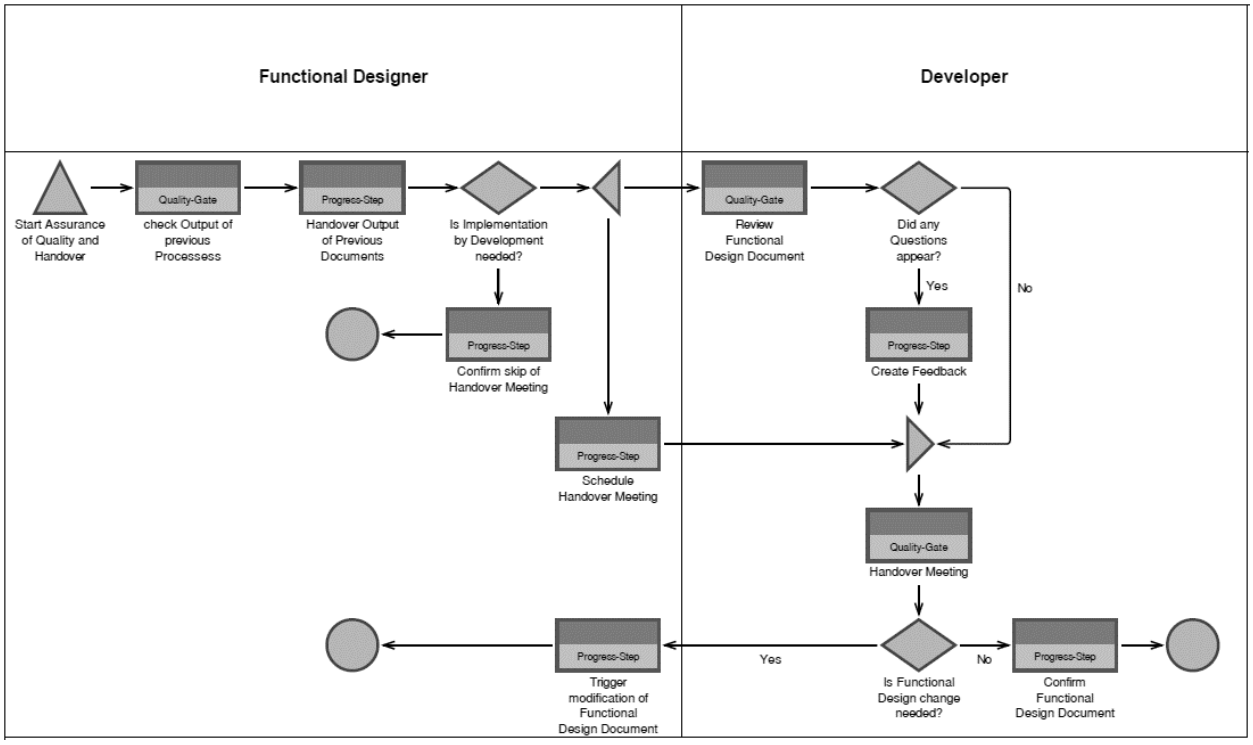

Figure 4.7: Assurance Quality and Handover – Optimized Workflow Layout

The following list shows the identified problems in chapter 4.2.1.3 and a description of how they were optimized in Figure 4.7.

1. **Setup Finishing Module due to Modular Approach:** This sub-process will represent the finishing-module of the whole process. Due to this fact, a sanity check is needed at the beginning to ensure, if the previously used sub-processes made sense and if the modular approach was not misused. Although, the rules of the modular approach say that at every first step of a sub-process, it can be decided to return the requirement to the predecessor sub-process no question-point is needed here, as it is the first step.
2. **Make Process Outcome more dynamic due to Modular Approach:** Additional to this sanity-check at the start of the process, it is possible that the whole following process "Configuration and Build" can be skipped in case the requirement does not need any implementation. In such a case the whole handover process can be skipped.

In the following Table 4.6 the *workflow-steps* of the original workflow and the optimized layout are compared. Compared to the original workflow layout the layout grew. The number of quality-gates and clarifications was not changed, but due to both required goals to fulfil the modular approach two new progress-steps had to be introduced, increasing the total number of workflow-steps by two. It is very likely that the finishing-module will be extended, compared to the original layout in any case, due to the fact that additional checks have to be implemented.

Measurement	Value before Optimization	Value after Optimization
Workflow-Steps	7	9
Clarification	0	0
Progress-Steps	5	7
Quality-Gates	2	2

Table 4.6: Assurance Quality and Handover – Comparison of Key Figures

4.2.2 Optimization Process Test

In the predecessor process “Solution Conception and Design” the three analyzed processes from chapter 4.1.2 were optimized. Due to the findings out of chapter 4.1.1.2, an equivalent for the SPICE process “Problem Resolution Management” should also be implemented. This kind of process cannot be optimized, as it did not exist in the original layout so far. It will be introduced in the process “Test” as a new sub-process.

4.2.2.1 Implementation Sub-Process Problem Resolution Management

The process is described in Figure 4.8, due to the findings out of chapter 4.1.1.2 “Problem Resolution Management”. In the process “Solution Conception and Design”, the functional designer creates the functional design document, which is the basis for the implementation of a requirement into the software by the development-team. After the implementation, the test-team tests the functionality of the implementation also based on the functional design document. So, all following steps in the implementation and testing are based on the functional design document, which is best known by its creator, the functional designer. As it is partly written in natural language, which always includes space for individual interpretation, it could happen that in all following implementation steps hidden misunderstandings occur. To ensure this does not happen, the first person starting the whole implementation – the functional designer – will also be the last quality gate, even after a successful test case as he is the only one who could identify misunderstandings and possible problems.

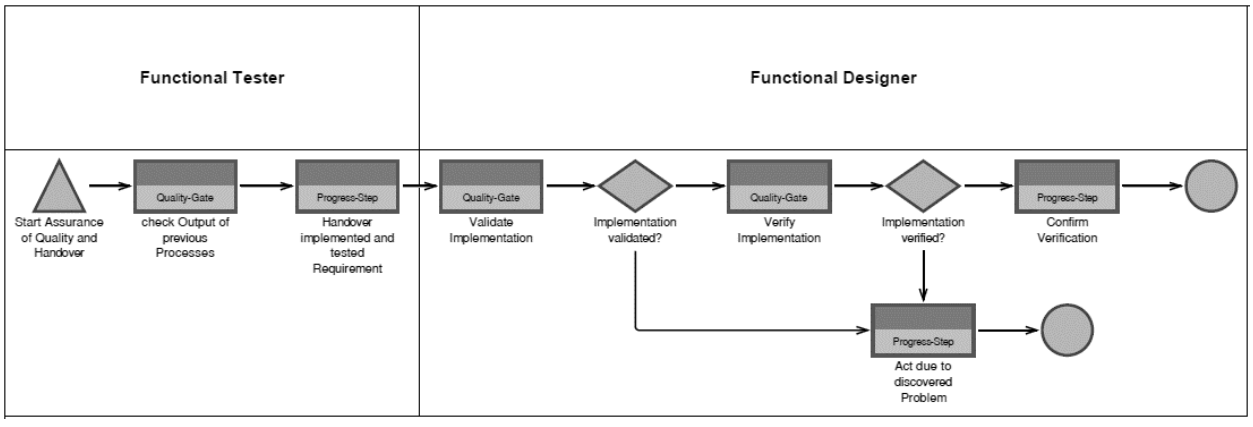

Figure 4.8: Problem Resolution Management – Optimized Workflow Layout

The workflow layout described in Figure 4.8 is a very simple quality check. As described before the functional designer checks the implementation by himself after it is successfully implemented by the development-team and tested by the test-team, which also hands it over to the designer in the first two steps of the process. The successor for this process can be chosen very dynamically by the functional designer according to possible occurred misunderstandings in the interpretation of the functional design document. It could be the case, that the document has to be rewritten. It is also possible that just the implementation has to be modified or that only the test has to be recreated, as the test-team missed to test the real core-functionality of the requirement. In case everything was done according to the functional design document and the understanding of the functional designer, the whole implementation process is verified, leading to the next sub-process.

In the following Table 4.7 the *workflow-steps* inside the new sub-process are shown. As the process did not exist before, the old values are zero in very case.

Measurement	Value before Optimization	Value after Optimization
Workflow-Steps	0	6
Clarification	0	0
Progress-Steps	0	3
Quality-Gates	0	3

Table 4.7: Problem Resolution Management – Comparison of Key Figures

4.2.3 Implemented Measurement Points

Each of the defined quality gates in the previous chapter is a measurement point for its own module. On Process-Level the quality gates in the finishing module can be in charge as general quality gates. This components, starting from the workflow steps, followed by the modules and the whole process on top, will be measured according to the strategy described in chapter 3.2.4.

The result identifies possible problems. The output of the static measurement parameters supports the management with taking the right actions to optimize the process.

4.3 Measurement Processes

The accompany Process, based on the processes identified in chapter 2.4.3.2, PIM.3 and MAN.6, should describe how to measure a process in the right way and how to react to the results, leading to continuous optimizations.

MAN.6 “Measurement” describes how to setup a successful measurement environment. First of all, the commitment should be established. It is also important to measure the right information for the individual needs. This step was prepared in chapter 3.2.4 and 3.2.5 by identifying all relevant key figures and KPIs for the internal optimization of the processes.

Due to PIM.3 “Process improvement” describes which steps to take to improve a process. First of all, the current situation has to be measured, to this end, the key figures and KPIs were introduced in the previous chapter. PIM.3 also describes that after an optimization was introduced, the measurement results should be compared with those from the starting situation to evaluate, if the optimization worked out.

According to this findings, the introduced key figures and KPIs should be raised for the original process as basis for the judgement of the impact of the optimizations. This figures should be compared to the figures after the optimization. This comparison of a part of the example project processes will be described in chapter 5.

5. Measurement Strategy and Results

In chapter 4.2, the example project processes were optimized due to the findings of the analysis in chapter 2. and the modular approach out in chapter 3.1. Parts of this described optimizations were also implemented in the real environment of the example project leading to the findings described in this chapter. Following measurements will be performed in this chapter.

1. **Measurement of existing process:** The sub-processes as described in chapter 4.1 will be measured using the workflow-step classification and the introduced key-figures from chapter 3.2.4. On top, the whole Process “Solution Conception and Design” will be measured. These figures will identify if there is a need for the modular approach in a real environment.

2. **Measurement on Overall Level:** The general impact of the change followed by the optimization will be measured on top process level, using the KPIs from 3.2.5.

3. **Measurement of new Sub-Process:** The process “Problem Resolution Management” as described in chapter 4.2.2.1 was introduced in the real example project environment. The usage of this step and the effects should be measured using the key figures described in chapter 3.2.4.

4. **Measurement of introduced Quality-Gates:** Also two quality gates were introduced in the example project environment as described in chapter 4.2 and will be measured on workflow-step level.

In every of the following sub-chapters, one of the described points will be measured with a final conclusion of the identified figures.

In the last sub-chapter “Measurement Conclusion” all four measurement sub-chapters will be summarized and described.

5.1 Measurement of existing Process

In this sub-chapter, the existing workflows inside the sub-processes as described in chapter 4.1 will be measured using the workflow-step classification and the key figures described in chapter 3.2.4. The key figures on module level as on process level will be used to identify, if there are sub-processes which are not used as much as others. This would identify a need for the modular approach.

The equations and description of the used key figures are located in chapter 3.2.4. In the following sub-chapters, the value of the key figures and KPIs is displayed as measured in the example project environment followed by a final conclusion of the value.

5.1.1 Sub-Process Definition of Solution Outline

The following key figures show the result of the analysis of the sub-process described in chapter 4.1.2.1 in the example project.

Static Workflow-Level Key Figures

Name of Key Figure	Value
sML-nWS Total Number of Workflow Steps	10
sML-rWS-C Clarification Workflow Steps	0%
sML-rWS-P Progress Workflow Steps	90%
sML-rWS-Q Quality-Gate Workflow Steps	10%

sML-rMinP Minimal Workflow	20%
sML-rMaxP Maximal Workflow	∞%
sML-rPD Spectrum of Workflow Usage	∞%

Table 5.1: Definition of Solution Outline - Static Workflow-Level Key Figures

The old workflow layout did not contain any Classification workflow steps. This problem is already solved in the optimized proposal in chapter 4.2.1.1.

The fact that sML-rMaxP is infinite is leading to an infinite spectrum of workflow-steps. Due to this figure, sML-rPD shows an unpredictable time effort until this workflow is finished. This figure was designed to have a reference point to forecast the needed time until a workflow is finished. The smaller the percentage is the less deviation exists. The loop leads to an infinite value and so to an unpredictable time effort. Loops are not always bad and in many cases necessary, but there should be a rule for the **maximal usage of one loop after which an escalation** should be triggered, making the needed time a bit more predictable.

Dynamic General Efficiency Key Figures

Name of Key Figure	Value
dGE-rASU Average Workflow Step Usage	84%
dGE-tATU Average Time Used	21.3 days
dGE-tATU-C Average Time Used Clarification	0 days
dGE-tATU-P Average Time Used Progress	2.7 days
dGE-tATU-Q Average Time Used Quality-Gates	1.9 days

dGE-rSC-C Usage Ratio Clarification	0%
dGE-rSC-F Usage Ratio Step Forward	86%
dGE-rSC-Q Usage Ratio Step Quality	14%
dGE-rSC-B Usage Ratio Step Back	0%

Table 5.2: Definition of Solution Outline - Dynamic General Efficiency Key Figures

The general efficiency key figures show that the minimal path (sML-rMinP) of workflow step usage of 20% is highly exceeded by the usage ratio of 84%. This is an indicator that some of the optional steps should be included into the shortest path as they are used in most cases. Also the fact that the quality gate usage with 14% is higher than the available quality gates (10%) shows that the loop containing the quality gate is used more than just once during an iteration. As the current workflow layout does not support the measuring of step backs and no clarifications are available, this values are always zero. The total of 21.3 days in average also contains waiting times, as the current layout does not support measuring, when the actual work was started and finished. Just the total time per workflow step is measured. This waiting time is also included in the time figures for the workflow steps classifications.

Clarification – Step Key Figures

Name of Key Figure	Value
dC-C-tAD Average Duration of Workflow Step	0 days
dC-C-rAD Average Duration Compared to Total Average Duration	0%
dC-C-rM-C Movements towards Clarification	0%
dC-C-rM-F Movements towards Step Forward	0%

dC-C-rM-Q Movements towards Quality-Gate	0%
dC-C-rM-B Movements towards Step Back	0%

Table 5.3: Definition of Solution Outline - Clarification - Step Key Figures

As there are no clarification workflow steps this value is always zero.

Quality-Gate – Step Key Figures

Name of Key Figure	Value
dC-Q-tAD Average Duration of Workflow Step	1.9 days
dC-Q-rAD Average Duration Compared to Total Average Duration	8.9%
dC-Q-rM-C Movements towards Clarification	0%
dC-Q-rM-F Movements towards Step Forward	100%
dC-Q-rM-Q Movements towards Quality-Gate	0%
dC-Q-rM-B Movements towards Step Back	0%

Table 5.4: Definition of Solution Outline - Quality-Gate - Step Key Figures

This figures show that the quality gate only requires 8.9% of the total progress time. With not even 10% of time usage the quality can be increased. This value can be important to support the management in introducing additional quality gates or decreasing the number of quality gates. As all successor workflow steps of the quality gate in this process are step forward workflow step changes, this figure is 100%. This figures will be valuable for measuring the new introduced quality gates in chapter 5.3.

Progress – Step Key Figures

Name of Key Figure	Value
dC-P-tAD Average Duration of Workflow Step	2.7 days
dC-P-rAD Average Duration Compared to Total Average Duration	91.1%
dC-P-rM-C Movements towards Clarification	0%
dC-P-rM-F Movements towards Step Forward	86%
dC-P-rM-Q Movements towards Quality-Gate	14%
dC-P-rM-B Movements towards Step Back	0%

Table 5.5: Definition of Solution Outline - Progress - Step Key Figure

As only one quality gate and progress workflow step exist in this workflow it is an interesting fact, that the figures dc-P-rM-F and dc-P-rM-Q are equal to the figures dGE-rSC-F and dGE-rSC-Q, as no clarification workflow step exists and no step back workflow step change can be measured by the example project environment.

5.1.2 Sub-Process Creation of functional Design

The following key figures show the result of the analysis of the sub-process described in chapter 4.1.2.2 in the example project.

Static Workflow-Level Key Figures

Name of Key Figure	Value
sML-nWS Total Number of Workflow Steps	4
sML-rWS-C Clarification Workflow Steps	0%
sML-rWS-P Progress Workflow Steps	100%
sML-rWS-Q Quality-Gate Workflow Steps	0%
sML-rMinP Minimal Workflow	50%
sML-rMaxP Maximal Workflow	50%
sML-rPD Spectrum of Workflow Usage	0%

Table 5.6: Cresation of functional Design - Static Workflow-Level Key Figures

In this sub-process are no quality-gates and clarification workflow steps. This problem is already solved in the optimization proposal in chapter 4.2.1.2. The spectrum of usable workflow steps until finishing the process is very small, leading to the sML-rPD value of 0%. This means that the time usage until this process is finished is very well predictable, although there are multiple available paths. From the predictability point of view, the starting process is already perfect.

Dynamic General Efficiency Key Figures

Name of Key Figure	Value
dGE-rASU Average Workflow Step Usage	50%
dGE-tATU Average Time Used	13.4 days
dGE-tATU-C Average Time Used Clarification	0 days

dGE-tATU-P Average Time Used Progress	6.7 days
dGE-tATU-Q Average Time Used Quality-Gates	0 days
dGE-rSC-C Usage Ratio Clarification	0%
dGE-rSC-F Usage Ratio Step Forward	100%
dGE-rSC-Q Usage Ratio Step Quality	0%
dGE-rSC-B Usage Ratio Step Back	0%

Table 5.7: Cresation of functional Design - Dynamic General Efficiency Key Figures

As this workflow does not contain workflow steps of different kinds and the spectrum of possible paths (sML-rPD) is 0%, leading to just one possible amount of workflow steps until finishing the process, the most figures are zero. The average duration of 13.4 for the whole process and the average duration of 6.7 days per workflow step also contains waiting periods, as the current measurement points do not provide the possibility to separate this figures.

Clarification – Step Key Figures

Name of Key Figure	Value
dC-C-tAD Average Duration of Workflow Step	0 days
dC-C-rAD Average Duration Compared to Total Average Duration	0%
dC-C-rM-C Movements towards Clarification	0%
dC-C-rM-F Movements towards Step Forward	0%
dC-C-rM-Q Movements towards Quality-Gate	0%

dC-C-rM-B Movements towards Step Back	0%

Table 5.8: Cresation of functional Design - Clarification - Step Key Figures

As there are no clarification workflow steps this value is always zero.

Quality-Gate – Step Key Figures

Name of Key Figure	Value
dC-Q-tAD Average Duration of Workflow Step	0 days
dC-Q-rAD Average Duration Compared to Total Average Duration	0%
dC-Q-rM-C Movements towards Clarification	0%
dC-Q-rM-F Movements towards Step Forward	0%
dC-Q-rM-Q Movements towards Quality-Gate	0%
dC-Q-rM-B Movements towards Step Back	0%

Table 5.9: Cresation of functional Design - Quality-Gate - Step Key Figures

As there are no quality gate workflow steps this value is always zero.

Progress – Step Key Figures

Name of Key Figure	Value
dC-P-tAD Average Duration of Workflow Step	6.7 days
dC-P-rAD Average Duration Compared to Total Average Duration	100%

dC-P-rM-C Movements towards Clarification	0%
dC-P-rM-F Movements towards Step Forward	100%
dC-P-rM-Q Movements towards Quality-Gate	0%
dC-P-rM-B Movements towards Step Back	0%

Table 5.10: Cresation of functional Design - Progress - Step Key Figure

As this process just contains progress steps, this values do not provide any meaningful information in this case.

5.1.3 Sub-Process Assurance Quality and Handover

The following key figures show the results of the analysis of the sub-process, described in chapter 4.1.2.3 in the example project.

Static Workflow-Level Key Figures

Name of Key Figure	Value
sML-nWS Total Number of Workflow Steps	7
sML-rWS-C Clarification Workflow Steps	0%
sML-rWS-P Progress Workflow Steps	71%
sML-rWS-Q Quality-Gate Workflow Steps	29%
sML-rMinP Minimal Workflow	71%
sML-rMaxP Maximal Workflow	86%
sML-rPD Spectrum of Workflow Usage	15%

Table 5.11: Assurance Quality and Handover - Static Workflow-Level Key Figures

Also in this process, the clarification workflow steps are missing. In this case this problem is already solved in the optimized approach in chapter 4.2.1.3. The spectrum of usable workflow paths with a deviation of 15% is a good value for predicting the time usage for forecasts.

Dynamic General Efficiency Key Figures

Name of Key Figure	Value
dGE-rASU Average Workflow Step Usage	77%
dGE-tATU Average Time Used	11.6 days
dGE-tATU-C Average Time Used Clarification	0 days
dGE-tATU-P Average Time Used Progress	1.1 days
dGE-tATU-Q Average Time Used Quality-Gates	4.1 days
dGE-rSC-C Usage Ratio Clarification	0%
dGE-rSC-F Usage Ratio Step Forward	65%
dGE-rSC-Q Usage Ratio Step Quality	35%
dGE-rSC-B Usage Ratio Step Back	0%

Table 5.12: Assurance Quality and Handover - Dynamic General Efficiency Key Figures

This figures show that the spectrum value sML-rPD makes sense, showing that the average usage of workflow steps lies with 77% almost in the middle between the possible usage of 71% - 86% of all available workflow steps. The fact that the quality gate in this process needs the participation of multiple stakeholders was leading to a longer average duration of 4.1 days. This indicates that it sometimes takes time to organize an appointment at which everyone has time, which could be an identified point for optimization.

Clarification – Step Key Figures

Name of Key Figure	Value
dC-C-tAD Average Duration of Workflow Step	0 days
dC-C-rAD Average Duration Compared to Total Average Duration	0%
dC-C-rM-C Movements towards Clarification	0%
dC-C-rM-F Movements towards Step Forward	0%
dC-C-rM-Q Movements towards Quality-Gate	0%
dC-C-rM-B Movements towards Step Back	0%

Table 5.13: Assurance Quality and Handover - Clarification - Step Key Figures

As there are no clarification workflow steps this value is always zero.

Quality-Gate – Step Key Figures

Name of Key Figure	Value
dC-Q-tAD Average Duration of Workflow Step	4.1 days
dC-Q-rAD Average Duration Compared to Total Average Duration	35%
dC-Q-rM-C Movements towards Clarification	0%
dC-Q-rM-F Movements towards Step Forward	29%
dC-Q-rM-Q Movements towards Quality-Gate	71%

dC-Q-rM-B Movements towards Step Back	0%

Table 5.14: Assurance Quality and Handover - Quality-Gate - Step Key Figures

In this process, the quality gate can have a progress step or a quality gate as successor workflow step. The ratio of 29% step forward workflow step changes shows, that the optional progress workflow step "Create Feedback" is just used rarely, which indicates a high quality of the created design document. The average duration for each quality gate compared to the total average time for the process is very high with a value of 35%. In case both available quality gates are used in this process, the total time for quality gates lies by 70%, which is a very high value. As the sub-process is called "Handover and Quality Assurance", this value is probably valid, as it fulfills the need of the process.

Progress – Step Key Figures

Name of Key Figure	Value
dC-P-tAD Average Duration of Workflow Step	1.1 days
dC-P-rAD Average Duration Compared to Total Average Duration	9.4%
dC-P-rM-C Movements towards Clarification	0%
dC-P-rM-F Movements towards Step Forward	76.2%
dC-P-rM-Q Movements towards Quality-Gate	23.8%
dC-P-rM-B Movements towards Step Back	0%

Table 5.15: Assurance Quality and Handover - Progress - Step Key Figure

Each progress workflow step just requires 9.4% of the total average process duration. This value is very low, as the progress step tasks are very small in this process, which concentrates on quality assurance and handover.

5.1.4 Process Solution Conception and Design

General Process Key Figures

Name of Key Figure	Value
sGP-nTM Number of available sub-processes	3
dGP-nAMI Average number of used sub-processes	1.6
dGP-tATI Average duration until finishing the process	20.9 days

Table 5.16: Solution Conception and Design - General Process Key Figures

This figures show that actually not all “modules” (or sub-processes) are used in each iteration. The average time for finishing an iteration is with 20.9 days even below the duration to finish just the first sub-process “Definition of Solution Outline”, indicating that this sub-process is not used very often, proving a need for the modular approach.

Process Overview Key Figures

Name of Key Figure	Value
sPO-nWS Number of all workflow steps	21
sPO-rWS-C Number of clarification workflow steps	0
sPO -rWS-P Number of progress workflow steps	18
sPO -rWS-Q Number of quality gate workflow steps	3

Table 5.17: Solution Conception and Design - Process Overview Key Figures

The ratio of all quality gates is 14.3%. No clarification workflow steps exist yet.

Process Overview per Module – Definition Solution Outline

Name of Key Figures	Value
$sPOM_{<M1>}$-rMU Sub-Process usage ratio	15%
$sPOM_{<M1>}$-rTU Duration compared to other sub-processes	46%

Table 5.18: Solution Conception and Design - Process Overview Definition Solution Outline

The sub-process “Definition of Solution Outline” is barely used. If it is used it takes almost the half of the time compared to all sub-processes together. This indicates that due to the little usage the process was not optimized yet, leading to the long durations. On the other hand, the long duration can also just be based on a more complex topic than in the other sub-processes.

Process Overview per Module – Creation of functional Design

Name of Key Figures	Value
$sPOM_{<M2>}$-rMU Sub-Process usage ratio	45%
$sPOM_{<M2>}$-rTU Duration compared to other sub-processes	29%

Table 5.19: Solution Conception and Design - Process Overview Creation of functional Design

The sub-process “Creation of functional Design” is used in almost every second iteration of the whole process. It takes almost one third of the time, compared to the duration of all sub-processes together.

Process Overview per Module – Quality Assurance and Handover

Name of Key Figures	Value
$sPOM_{<M3>}$-rMU Sub-Process usage ratio	100%
$sPOM_{<M3>}$-rTU Duration compared to other sub-processes	25%

Table 5.20: Solution Conception and Design - Process Overview Quality Assurance and Handover

The last sub-process "Quality Assurance and Handover" is used in every iteration. More than 50% of all iterations just contain this sub-process. This shows that in more than 50% of all cases, besides a documented handover to the next process (and working step) nothing is done.

5.1.5 Conclusion

The figures from the sub-process analysis show that quality-gates are often used and need less time than normal progress workflow steps. Not all of the sub-processes contain quality-gates, leading to an average value of just 14% quality-gates. Compared to the usage of quality-gates if they exist in a module (around 29%), this value is low. The number of quality-gates and clarifications should be raised and harmonized between the sub-processes, estimating a best-practice scenario.

In two out of three sub-processes, the spectrum of possible paths until the sub-process is finished is very big (15% and infinite) leading to a hardly predictable time usage for forecasts. The infinite value is caused by workflow-loops, which are toxic as they are unpredictable. It is important to implement a counter for loops, allowing a maximum number of iterations before escalating the problem to the management. This would lead to a more structured process and a predictable time usage.

The fact that the number of usage of the different sub-processes varies a lot is an indicator for the need of a more modular process approach. This variety leads to a hard creation of meaningful data for analysis in the current setup. Using the modular approach, the measurement strategy could be worthier and the environment would fulfill the needs of dynamic processes.

5.2 Measurement of new Sub-Process

In this sub-chapter the new introduced process "Problem Resolution Management", as described in the SPICE standard, is analyzed. This process was introduced as mandatory module to ensure the quality of the outcome. The analysis should show the legitimacy of the new introduced process. In case 100% of all cases are forwarded to the next sub-process such an approval-process is set up on the wrong position, leading to a waste of time.

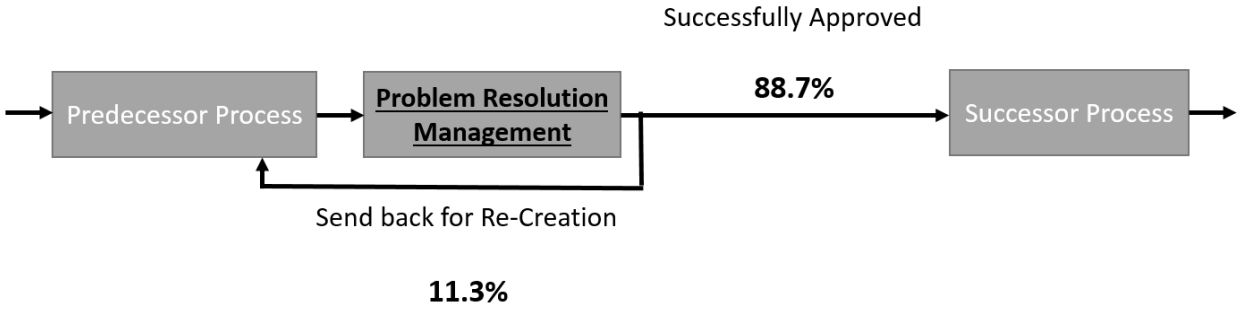

Figure 5.1: Problem Resolution Management - Problem Identification Ratio

The Figure 5.1 shows that in 11.3% of all cases a problem was identified before reaching the customer. This is an immense value, as these 11.3% would otherwise reach the customer and cause a defect, decreasing the quality and the reputation of the program. All of this identified problems could be solved by a repetition of the predecessor process.

This ratio shows, that it will also lead directly to a future decrease on defects logged by the customer, as most of the problems can be identified already during the internal software engineering process. This ratio proves the need and legitimation of the implementation of this sub-process.

5.3 Measurement of Quality-Gates

Two quality-gates were introduced as described in chapter 4.2. It is to be analyzed, if the quality-gates are leading to a step-back, fulfilling their duty of identified problems analogous to the introduced sub-process "Problem Resolution Management" as described before.

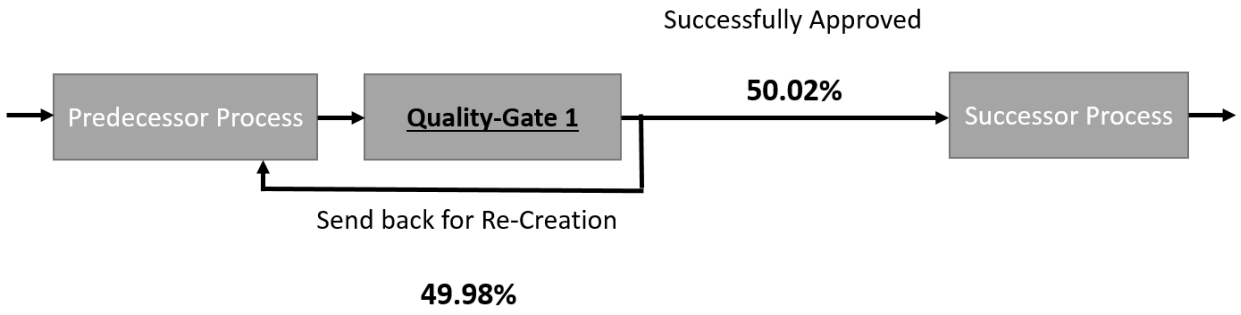

Figure 5.2: Quality Gate Implementation - Problem Identification Ratio QG1

In the first quality-gate almost exactly the half of all iterations got send back to the predecessor workflow-step, using the feedback routine. This shows that there is a major leak in quality which got identified via the quality gate.

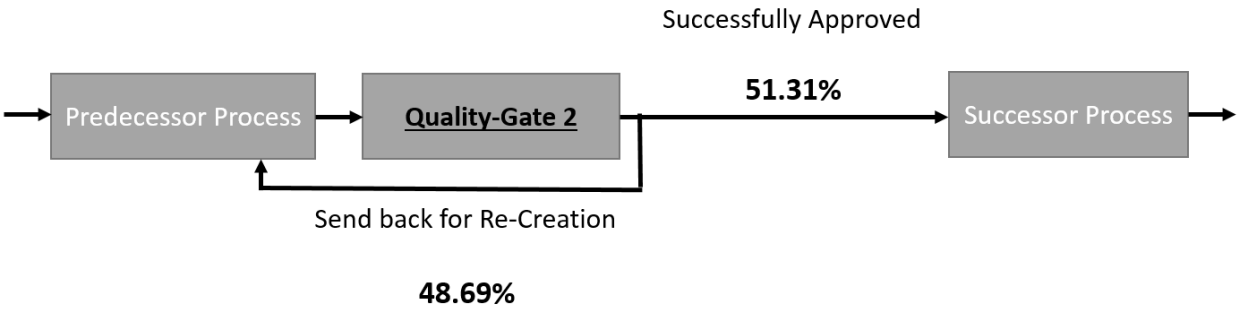

Figure 5.3: Quality Gate Implementation - Problem Identification Ratio QG2

Also in the case of the second introduced quality-gate, almost the half of all iterations which reached the quality-gate were send back to the previous status as a problem was identified. In both cases the implementation of the quality gate identified major problems in quality on the particular position, leading to a quality-increase.

The recommendation by the standardized process frameworks CMMI and SPICE, that the introduction of quality-gates is not a waste of time, but a reasonable step rising the quality was proven in both practical cases.

5.4 Measurement Conclusion

The measurement in this chapter identified problems in the current workflow-layout, which were not identified during the theoretical analysis, as the infinite-loop-problem. Loops in the workflow-layout are toxic, leading to an unpredictable period of time a sub-process needs until it is finished. To countermeasure this problem, a loop counter could be installed, allowing a maximum number of iterations within a loop, before being forced to escalate the problem.

Also the fact, that the usage of sub-processes within the iterations varies a lot, identifies a need for a more dynamic layout, as described in the modular-approach.

The recommended quality-gates within the standardized frameworks CMMI and SPICE, which were introduced at two locations in the example environment worked out perfectly. In both cases almost 50% of all iterations were send back to the predecessor workflow-steps to fix an identified problem. The recommendation of both frameworks, that the simple increase of quality-gates is the right solution to work against quality problems is right. Nevertheless, it should not be forgotten, that fighting against quality-issues with quality-gates is fighting against symptoms, but not the root of the problem. To decrease the step-back rate within quality-gates the quality of work in the predecessor steps should be raised by introducing supporting processes as trainings and workshops.

The implemented sub-process recommended by the SPICE framework “Problem resolution Management” also worked out perfectly. The whole sub-process can be described as a huge final quality-gate to identify a problem before it reaches the client, not only raising the quality but also supporting the reputation of the whole project. As this step is in charge at the very end of the implementation, after all quality-gates already identified possible problems, the rate of problems should be very low, leading to just 11.3%, compared to the value of almost 50% at the quality-gates before.

5.4.1 Interpretation for Modular Approach

The need for a dynamic approach as the modular approach was identified within the analysis on process-level, as the number of usage between the sub-processes varies a lot. The implemented sub-process “Problem Resolution Management” also showed, that a final quality-sub-process at the end of a major process (as the whole implementation process) is very important. This final quality-assurance sub-process is introduced in the modular approach as the finishing-module.

Also the fact, that the average need for workflow-steps until finishing a sub-process is far below the total number of available workflow-steps shows that many sub-processes are skipped ending with the first question-point e.g. “Is Reprioritization needed?”. The modular approach makes such a skip obsolete as the sub-process would not have even begun, as the final-workflow-step in the predecessor sub-process is in charge to decide which next module is in charge.

5.4.2 Interpretation for Workflow Step Classification

The workflow-step classification and the introduced key figures revealed to be a very useful tool to measure the maturity of a workflow in a real example environment. Not only that the figures identified problems which were not identified until the analysis in a productive environment, they also provide triggers of how to improve the process, comparing the static key figures of the workflow-step classification with best practices or average values.

6. Summary and Findings

The purpose of this thesis was to investigate for new optimization approaches in the process-layout of a large Software-Engineering project. For this an example process was identified and used as basis for the analysis in chapter 1.3.

As starting situation for new approaches the two standardized process frameworks were investigated and analyzed in chapter 2. leading to a basis of findings and ideas for the creation of the new approaches.

Based on the real challenges in production by the management, and the needs of the employees and the ideas of the standardized frameworks, two new optimization approaches were developed and introduced in chapter 3. The first approach is the Modular-Approach, which tries to find a solution for the need of more dynamic processes but also fulfilling the need for standardized processes and measuring. Instead of using mandatory sub-processes in a predefined hardwired way, various optional modules should be available which can be used dynamically due to the needs of each particular requirement.

The second approach is the workflow step classification, which challenges the intention of the setup of workflows in production. Instead of just concentrating on progress and performance, the quality and clarification during the creation of a product should not be forgotten. To this end, each and every step inside a workflow should be tagged with one of the identified classifications of workflow-steps. Based on this approach, a whole measurement strategy was developed, leading to key figures which provide the management information about the current maturity of the workflow-layouts and proposals on how to introduce optimizations.

To prove the theoretical feasibility in a productive environment, one process with all its sub-processes was analyzed and optimized due to the best-practices out of the standardized frameworks CMMI and SPICE and the new optimization approaches (out of chapter 3. in chapter 4.

As the example processes were gathered in a real example process, not all ideas could be implemented. Some particular parts where introduced in the example process, as described in chapter 5. Two quality-gates and one sub-process was introduced to challenge the statements by the standardized frameworks CMMI and SPICE. As most of the workflow-step classification (without the workflow step-change classification “Step-Back”) could be measured with a change in the real productive environment the new developed measurement strategy was tested proving its quality and revealing invisible problems.

6.1 Conclusion on Modular Approach

The new created dynamic approach on how to increase the creativity and flexibility of processes in a software-development environment is described in detail in chapter 3.1. In chapter 4.2 one process from the example project is modified, discovering room for optimizations.

The downside of the modular-approach is, that the modification of the processes in a real environment would be a huge step with a high risk of failure. This fact got cleared step by step during the creation of chapter 4. and 5. Due to the high risk, it was not possible to test the approach in a practical way in the real example project.

Anyhow, the need for a dynamic approach, as the modular approach, with all its parts as the module classification, was proved by the analysis of the real example project processes in chapter 5.

If the modular approach is the right solution or a too risky change could not be proven within this thesis.

6.2 Conclusion on Workflow Step Classification

In contrast to the modular approach, described before the workflow-step classification (described in chapter 3.2) could be tested in the real practical test environment.

This is most probable the biggest advantage and makes this approach practicable, as every existing process can already be measured from the start using this approach, identifying problems and leading to a step-by-step optimization.

The description of the example project's processes using the workflow-step classification is described in chapter 4.1 and measured in chapter 5.1. This approach with its key figure portfolio was a total success, leading to the identification of invisible problems and giving a better understanding of the maturity of the process.

Although the approach worked out great, it is still a first draft with a lot of important missing figures, which has to be further developed to cover all needs of measurement.

6.3 Outlook

The measurements in the real example environment in chapter 5. showed that the analysis and the developed optimization approach did not cover all challenges, which occur in the real practical environment. After measuring all described measurement points in the example project, there were additional fields for improvement discovered, which were not identified during the theoretical analysis of this thesis.

Also the fact that the modular approach could not be tested in a real environment leads to a big question-mark, if the elaborated approach is a success or a failure. This production-test in a real environment would be the next step to be achieved as a basis for a development of this approach.

The workflow-step classification with its key figure portfolio has been identified as success. Nevertheless, just one major process in the example project out of five was analyzed within this

thesis. To substantiate this success, further existing processes should be measured using this approach. Also the key figure portfolio needs further development to cover the big picture of the situation in a project environment. In any case, this approach has proven that it is worth continuing to work on it.

Bibliography

Automotive SIG. (2007, May 5). *Automotive SPICE - Process Assessment Model v2.3.* The SPICE User Group 2005 - 2007. Retrieved from Automotive SPICE: http://www.itq.ch/pdf/AutomotiveSPICE_PAM_v23.pdf

Automotive SIG. (2010, May 10). *Automotive SPICE - Process Assessment Model v2.5.* The SPICE User Group 2005 - 2010. Retrieved from Automotive SPICE: http://www.automotivespice.com/fileadmin/software-download/automotiveSIG_PAM_v25.pdf

Automotive SIG. (2010, August). *Prozessassessmentmodel - Prozessbewertung gemäß Automotive SPICE in der Entwicklung von softwarebestimmten Systemen.* VDA Quality Measurement Center. Retrieved from Automotive SPICE: http://vda-qmc.de/fileadmin/redakteur/Publikationen/Download/Automotive_SPICE_Prozessassessmentmodell.pdf

Automotive SIG. (2015, July 16). *Process Reference Model / Process Assessment Model Version 3.0.* VDA Quality Management Center. Retrieved from Automotive SPICE : http://www.automotivespice.com/fileadmin/software-download/Automotive_SPICE_PAM_30.pdf

Capgemini, Sogeti and HP. (2015). *World Quality Report 2015-2016.* Capgemini, Sogeti and HP.

Carnegie Mellon University. (2016, March 6). *Brief History of CMMI.* Pittsburgh: Carnegie Mellon University. Retrieved from http://resources.sei.cmu.edu/asset_files/Brochure/2009_015_001_28416.pdf

CMMI Product Team. (2010, November). *CMMI® for Development, Version 1.3 - Improving processes for developing better products and services.* CarnegieMellon. Retrieved from http://www.sei.cmu.edu/reports/10tr033.pdf

M. Drmota, B. Gittenberger, G. Karigl, A. Panholzer. (2010). *Mathematik für Informatik* (3. Auflage ed.). Heldermann Verlag.

Schlingloff, P. H. (2015, July 07). *Präsentation, Qualitätssicherung von Software (SWQS).* Retrieved from http://slideplayer.de/slide/215104/

Software Quality Assurance.org. (2015). *Software Quality Assurance.org*. Retrieved December 27, 2015, from http://www.software-quality-assurance.org/

Sparx Systems. (2014). *The Business Process Model.* Sparkx Systems 2004. Retrieved from http://www.sparxsystems.com.au/downloads/whitepapers/The_Business_Process_Model.pdf

Steiner, M. (2015). *Wie effizient ist die Matrix? Anforderungen an eine Balanced Scorecard zur Steuerung einer als Matrixorganisation strukturierten Softwareentwicklung.* Vienna.

Ziegler, P. (2015). *Big Data Studie Österreich 2015/16.* Novomatic Forum.

Appendix A – PAM BP Example

Process ID	ACQ.3
Process Name	Contract agreement
Process Purpose	The purpose of Contract agreement process is to negotiate and approve a contract / agreement with the supplier.
Process Outcomes	As a result of successful implementation of this process: 1) a contract / agreement is negotiated, reviewed, approved and awarded to the supplier(s); 2) the contract / agreement clearly and unambiguously specifies the expectations, responsibilities, work products / deliverables and liabilities of both the supplier(s) and the acquirer; 3) mechanisms for monitoring the capability and performance of the supplier(s) and for mitigation of identified risks are reviewed and considered for inclusion in the contract conditions; and 4) proposers / tenderers are notified of the result of proposal / tender selection.
Base Practices	**ACQ.3.BP1: Negotiate the contract / agreement.** Negotiate all relevant aspects of the contract / agreement with the supplier. [Outcome 1] NOTE 1: Relevant aspects of the procurement may include
	• system requirements • acceptance criteria and evaluation criteria • linkage between payment and successful completion of acceptance testing • process requirements, process interfaces and joint processes. **ACQ.3.BP2: Specify rights and duties.** Unambiguously specify the expectations, responsibilities, work products/deliverables and liabilities of the parties in the contract / agreement. [Outcome 2] **ACQ.3.BP3: Review contract / agreement for supplier capability monitoring.** Review and consider a mechanism for monitoring the capability and performance of the supplier for inclusion in the contract / agreement conditions. [Outcome 3] **ACQ.3.BP4: Review contract / agreement for risk mitigation actions.** Review and consider a mechanism for the mitigation of identified risk for inclusion in the contract / agreement conditions. [Outcome 3] **ACQ.3.BP5: Approve contract / agreement.** The contract / agreement is approved by relevant stakeholders. [Outcome 1] **ACQ.3.BP6: Award contract / agreement.** The contract / agreement is awarded to the successful proposer / tenderer. [Outcome 1] **ACQ.3.BP7: Communicate result to tenderers.** Notify the result of the proposal / tender selection to proposers / tenders. After contract award inform all tenderers of the decision. [Outcome 4]

[(Automotive SIG, 2007)]

Appendix B – PAM WP Example

Output Work Products
13-09 Meeting support record [Outcome 1]
02-01 Commitment / agreement [Outcome 1]
02-00 Contract [Outcomes 1, 2, 3]
13-05 Contract review record [Outcome 1]
13-04 Communication record [Outcome 4]

[(Automotive SIG, 2007)]