

Requirements engineering and design  
of human-centered information services  
utilizing near-eye display devices

Master's Thesis

by

Maximilian Johannes Walter Sachs

under supervision of

Univ.-Prof. Dipl.-Ing. Dr.techn. Siegfried Vössner

and

Dipl.-Ing. Dr.techn. Wolfgang Vorraber

Institute of Engineering and Business Informatics  
Graz University of Technology

Graz, May 2014



# Abstract

Recent developments in the space of Near-Eye Display Devices provide new venues for the development of information services. These new devices and services could lead to process improvements with regard to efficiency and effectiveness. This Master's Thesis explores the possibilities for process improvement with regard to medical surgeries by introducing information services based on such devices. It will provide a theoretical perspective on Requirements Engineering as a tool for system planning both in historical and in current context. Alternatives to Requirements Engineering in its traditional and linear form are explored by discussing the concepts of Agile Development Methods. Possible approaches to combine these different software development and planning methods are mentioned. Incremental Requirements Engineering is then used as a tool to gather information for the design of a system which aims to utilize previously identified potential for improvement. Based on the collected requirements, a system architecture is described and a Proof-of-Concept implementation is provided, which satisfies the fundamental requirements posed by previously identified use cases and serves as a basis for real-world tests.

Deutsche Fassung:  
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008  
Genehmigung des Senates am 1.12.2008

## EIDESSTÄTTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am .....

.....  
(Unterschrift)

Englische Fassung:

## STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

.....  
date

.....  
(signature)

# Acknowledgments

I would like to take this opportunity to thank all people who supported me during the creation of this Master's Thesis.

First and foremost, I would like to thank Dipl.-Ing. Dr.techn. Wolfgang Vorraber and Univ.-Prof. Dipl.-Ing. Dr.techn. Siegfried Vössner for giving me the opportunity to write this thesis and for supporting me in the process.

In addition, I would like to thank the Krankenhaus der Elisabethinen Graz GmbH, Univ. Prof. Ing. Dr. Gerhard Stark and the whole staff, who provided helpful insight into the well-established processes in medical environments over the course of the overall project, which this Master's Thesis is a part of. I would like to thank DGKP Michael Weldi, Karlheinz Söls and Josef Lemberger specifically, for answering all my questions regarding technical and medical aspects.

Of course, I would also like to thank my family and friends without of whom I would not be where I am today.

# Table of Contents

1	Introduction .....	1
1.1	Motivation .....	1
1.1.1	Requirements Engineering .....	1
1.1.2	Near-Eye Display Devices .....	3
1.2	Problem Statement and Limitations.....	4
1.3	Structure of the Thesis .....	5
2	Requirements Engineering: Theory .....	7
2.1	A General Perspective on Requirements Engineering .....	7
2.2	On the History of Requirements Engineering .....	8
2.3	Definitions.....	9
2.3.1	Requirements Engineering .....	10
2.3.2	Requirement.....	10
2.3.3	Functional Requirement.....	11
2.3.4	Non-Functional Requirement.....	12
2.3.5	Stakeholder .....	13
2.3.6	System, Context and Environment.....	13
2.4	General Remarks on Requirements Engineering Approaches .....	14
2.5	State of the Art Requirements Engineering .....	16
2.5.1	Pre-Elicitation Phase.....	16
2.5.2	Elicitation Phase .....	17
2.5.3	The SOPHIST-REgelwerk for Requirement Sanitization .....	20
2.5.4	Requirement Specification .....	21
2.5.5	Requirement Validation .....	22
2.5.6	Requirements Management .....	24
2.6	Criticism .....	26

2.6.1	Psychological Factors with regard to Requirements.....	26
2.6.2	The lack of a clear definition of Non-Functional Requirements .....	26
2.7	A Completely Different Approach: Agile Software Development.....	27
2.7.1	A Short Introduction to Agile Software Development Techniques.....	29
2.7.2	Natural Emergence of Requirements in Agile Software Development Teams..	35
2.7.3	Introducing Agile Aspects to Requirements Engineering .....	36
2.7.4	Traditional (linear) Requirements Engineering and Agile Software Development – It depends on the Project.....	37
3	Requirements Engineering: Application .....	39
3.1	Categorization of Use Cases .....	40
3.2	Project Goals.....	41
3.3	Monitoring of Real Time Patient Data.....	42
3.3.1	Use Case Instance Explanation – Percutaneous Transluminal Angioplasty.....	42
3.3.2	Classification and Explanation.....	43
3.3.3	AS-IS-Analysis and Identification of Potential for Improvement .....	43
3.3.4	Description of Resulting Use Case.....	44
3.4	Navigation and Medical Imaging .....	47
3.4.1	Use Case Instance Explanation – Laparoscopic Cholecystectomy.....	47
3.4.2	Classification and Explanation.....	47
3.4.3	AS-IS-Analysis and Identification of Potential for Improvement .....	48
3.4.4	Description of Resulting Use Case.....	50
3.5	Viewpoint of Surgeon for Improved Assistance .....	52
3.5.1	Use Case Instance Explanation – Open Cholecystectomy .....	52
3.5.2	Classification and Explanation.....	53
3.5.3	AS-IS-Analysis and Identification of Potential for Improvement .....	53
3.5.4	Description of Resulting Use Case.....	55

3.6	Recording of Medical Processes for Post-Evaluation and Teaching Purposes.....	57
3.6.1	Use Case Instance Explanation – Surgery Simulation Training .....	57
3.6.2	Classification and Explanation.....	58
3.6.3	AS-IS-Analysis and Identification of Potential for Improvement .....	58
3.6.4	Description of Resulting Use Case.....	60
3.7	Virtual Consultations .....	61
3.7.1	Use Case Instance Explanation – Open Cholecystectomy with Consultation....	61
3.7.2	Classification and Explanation.....	61
3.7.3	AS-IS-Analysis and Identification of Potential for Improvement .....	62
3.7.4	Description of Resulting Use Case.....	63
3.8	Elicitation and Identification.....	65
3.8.1	Hardware Requirements .....	65
3.8.2	Functional Requirements .....	66
3.8.3	Non-Functional Requirements .....	66
3.9	Analysis and Specification .....	67
3.9.1	Indexing Format .....	67
3.9.2	Definitions .....	68
3.9.3	Functional Requirements .....	68
3.9.4	Non-Functional Requirements .....	70
3.9.5	Hardware Requirements .....	71
3.9.6	Conflicts.....	72
3.10	Applied Requirements Engineering – Conclusion .....	72
4	Design and Architecture .....	73
4.1	The Platform – Google Glass Explorer Edition.....	73
4.2	General Structure .....	75
4.2.1	Requirements for the system with regard to its general architecture .....	75

4.2.2	Possible solutions .....	76
4.2.3	The choice behind the Thin-Client-Architecture .....	78
4.3	System, Context and Environment .....	80
4.4	Glass Client .....	81
4.4.1	Requirements for the Client Implementation .....	81
4.4.2	Possible solutions .....	82
4.4.3	The choice behind the GDK and the Immersion .....	82
4.5	Glass Server .....	82
4.6	Interfaces .....	84
4.6.1	Internal Interface – Glass Messaging Protocol .....	84
4.6.2	External Interfaces .....	90
5	Implementation – Proof-of-Concept .....	93
5.1	General Choices .....	94
5.1.1	Glass Server .....	94
5.1.2	Glass Client .....	95
5.1.3	Communication .....	95
5.2	Capabilities .....	97
5.2.1	Glass to Server Video Transmission .....	97
5.2.2	Glass to Server Picture Transmission .....	97
5.2.3	Glass to Server Audio Transmission .....	98
5.2.4	Server to Glass Video and Still Frame Transmission .....	98
5.2.5	Server to Glass Audio Transmission .....	98
5.2.6	Miscellaneous Features .....	99
6	Evaluation .....	100
7	Future Work and Conclusion .....	102
7.1	Data Security and Standards Conformity .....	102



7.2	Further Use Cases and Functionality .....	102
7.3	Extension of Proof-of-Concept .....	103
7.4	Interfacing with already established systems .....	104
7.5	Conclusion .....	104
8	References .....	I
9	List of Figures .....	VI
10	List of Tables.....	IX
11	Appendix.....	X
11.1	BPMN Diagrams.....	X
11.1.1	BPMN Legend .....	X
11.1.2	Percutaneous Transluminal Angioplasty.....	XI
11.1.3	Laparoscopic Cholecystectomy .....	XII
11.1.4	Open Cholecystectomy .....	XIII
11.1.5	Surgery Simulation Training .....	XIV
11.1.6	Open Cholecystectomy with Consultation.....	XV

# 1 INTRODUCTION

---

In the following section the motivation behind this Master's Thesis will be explained. In addition to this, an overview over Requirements Engineering and Near-Eye Display Devices will be given. The problem this thesis focuses on along with its limitations will be stated and finally a structural overview over the individual parts will be given.

## 1.1 MOTIVATION

The central motivation behind this Master's Thesis is the improvement of medical processes by utilizing Near-Eye Display Devices. This is achieved by an analysis of the processes as they are currently implemented, the identification of potential for improvement and finally the development of a concept that utilizes this potential. The core process behind the analysis of potential use cases during the development of this system is Requirements Engineering.

### 1.1.1 Requirements Engineering

Requirements Engineering has been recognized as an essential part of the software development process early on. In fact, the following quote by Frederick P. Brooks can be found in many articles on the subject:

*“The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.” (Brooks 1987)*

Many examples can be called upon to provide proof for the statement by Brooks, especially in the field of software development where – mainly because of the complexity of large projects – the stakes are usually high and problems occur frequently.

Generally, it can be said that the cost of fixing an error during the development of a software system is almost inversely exponential to the amount of time invested in the development (see Figure 1), with an extreme peak once the software is in operation. (Boehm 1981) Therefore, the early detection of potential problems is vital for project success. (Stecklein, Dabney et al. 2004, Boehm 1981)

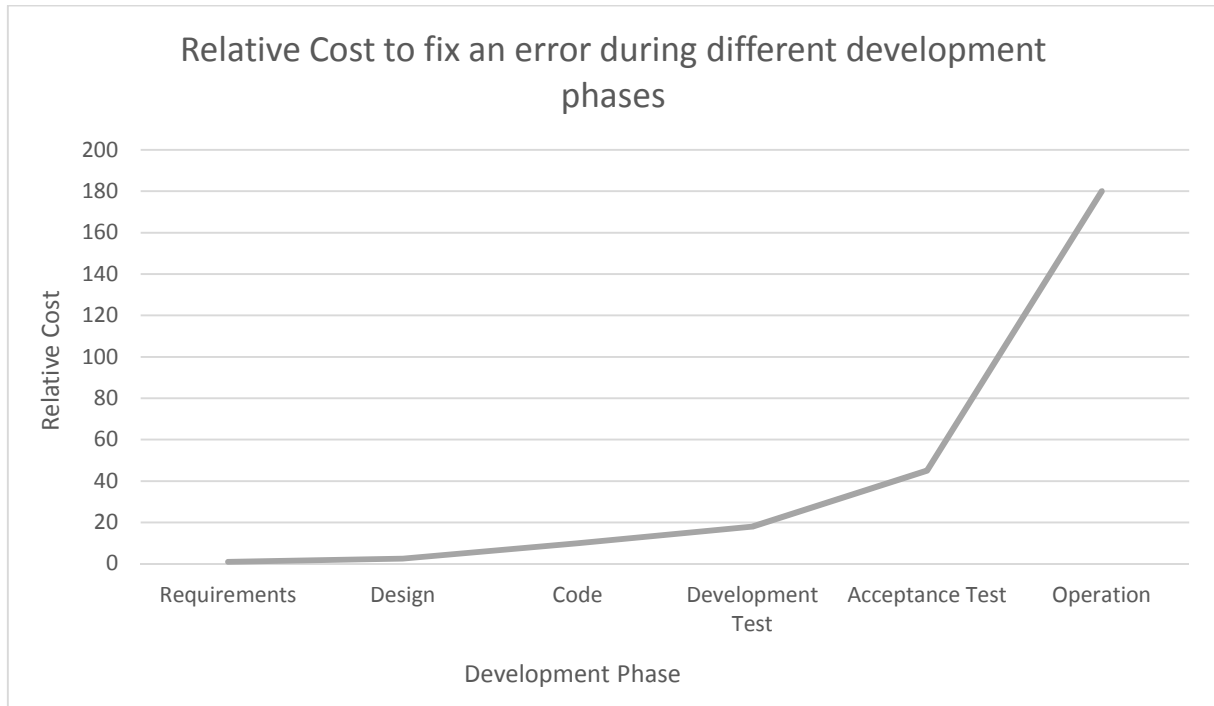


Figure 1: The relative cost to fix an error during different project development phases (figure modified from (Boehm 1981)).

For example, if a critical error is identified when the product is already available on the market, the cost of fixing it might quickly be in the millions. Although not a software system, a notable example for this phenomenon, which is passionately mentioned in software testing and verification courses at universities, is the rather infamous Intel Pentium-FDIV-Bug which caused wrong results for certain floating point calculations. (Nicely 1994)

Precisely because of this difficulty in rectifying errors in late stages of software development, strategies have been proposed that involve extensive planning-phases, which ultimately led to the birth of Requirements Engineering as a field of research.

The necessity for a solid understanding of the problem before or during system development is one of the reasons why Requirements Engineering has been chosen as the basis of this Master's Thesis.

### 1.1.2 Near-Eye Display Devices

Especially in the past five to seven years, the field of mobile computing underwent tremendous changes and improvements. With the emergence of a mass market for Smart Devices – Phones in particular –, small-form-factor hardware became more and more important for the industry. This focal shift has led to massive developments with regard to power consumption, networking and, most importantly, raw computing power.

Building a device that can be worn and operated with voice alone seems almost like the logical next step and is exactly what Google is aiming at with Glass.



*Figure 2: Google Glass Explorer Edition. The device which serves as the mobile platform for development during this thesis.*

Figure 2 shows the Google Glass Explorer Edition which currently serves as the mobile platform for the information system developed further down in this thesis (see section 4 below). A detailed overview over the platform and its features will be given in section 4.1 below.

With Google behind the development of such a device for the consumer market, it seems reasonable to assume that this type of device will gain significance quickly in the coming months and years. However, not only consumers may benefit from this technology, as potential for a wide variety of projects, especially with regard to the improvement of already existing processes, can be seen. Google Glass, as an early contender in this particular space, serves throughout this thesis and the surrounding overall project as the basis for potential improvements in medical environments.

With the underlying Android platform (Google Inc. 2013a), Google Glass is essentially a smart phone built into the frame of glasses. Besides all the features that are common in today's devices, like voice control and specifically designed user interfaces, Glass has one unique feature: A semi-transparent display that is mounted in such a way that its image appears in the corner region of the wearers eye. This screen, which is constantly in the user's field of vision, along with the capability to use speech as a means of interaction and common hardware elements like a microphone and a camera, provides a whole new experience. Starner (2013) describes this as the solution for what he sees as a fundamental problem of Smart Phones. He argues, that by reducing the delay between what the user wants to do and when he actually does it to a value below a certain threshold, the device becomes an "extension of the self". (Starner 2013)

This new development, where these small devices provide a means to display an image without obstructing reality too much for the wearer, open up a wide variety of new possibilities. The traditional HUD ("Heads-Up Display") has been implemented in various ways already, but always with the need for heavy machinery and the goal was often to provide a way to overlay images onto the entire field of vision of the wearer. In these cases, the goal was then, for example, to provide a way to view the skeleton of a patient whilst looking at him, without having to alternate between the patient and a display with the medical image. (Thomas, Sandor 2009)

The approach taken in this thesis is slightly different. Google Glass, as one of the early contenders in this new space of electronics, serves as a potential platform for the systematic exploration of potential for process improvement in medical processes. This will be achieved by providing information in a fixed spot in the visual field of the user, rather than on top of his entire field of vision, thus providing an additional information channel, instead of altering an existing one.

## 1.2 PROBLEM STATEMENT AND LIMITATIONS

The aim of this thesis is the improvement of well-established processes in medical environments with regard to performance, efficiency and / or safety.

The process used to achieve this is a very systematic one. By utilizing Requirements Engineering, especially with a focus on the Status-Quo and thus the analysis of processes as they are in place at the current time and therefore the possibility to find weaknesses or potential for optimization by working very closely with people who are involved (nurses, technical and medical personnel and surgeons), use case scenarios are identified, which may be improved by the introduction of an information service based on Near-Eye Display Devices.

It is the core task of this thesis to explore potential for improvement with regard to processes as they are currently implemented in hospitals, specifically the Hospital of Elisabethinen in Graz. Identified problems will then be used as a basis to develop solutions by utilizing Near-Eye Display Devices such as Google Glass. To prove the viability of these solutions, a Proof-of-Concept is developed that implements all capabilities that are fundamental to the overall system design. In summary, the primary aim is to identify use cases in a methodological manner, document them, design a system and finally implement a Proof-of-Concept which demonstrates the potential of the technology.

### 1.3 STRUCTURE OF THE THESIS

This thesis is structured into three fundamental parts (see Figure 3): **Theory**, **Requirements Engineering** of human-centered information services for use in medical environments and, in part, the **Development** of these services.

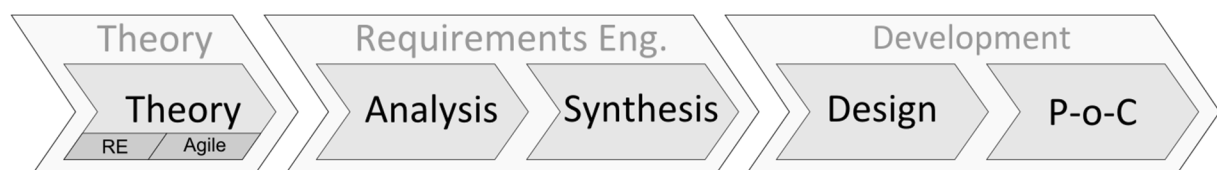


Figure 3: The three fundamental parts of this thesis, along with a more detailed sub-division into five essential sections.

The first part of the more detailed structure outlined in Figure 3 above, **Theory**, provides a theoretical overview over **Requirements Engineering**, both in historical and in current context. The fundamental principles of the Requirements Engineering process will be explained and new developments, especially with regard to **Agile Software Development** approaches, will be discussed.

The first of the three following parts is the **Analysis** of currently implemented processes in medical environments which will focus strongly on how these processes are presently carried out. What follows next is called **Synthesis**. At the core of this part lies the identification of potential for improvement along with the specification of use cases that utilize such potential. These use cases then provide the basis for the design of the system which is done in the next part: **Design**. It aims to design a system which allows for the use cases outlined in the Synthesis section to be performed.

The last section, **Proof-of-Concept (PoC)**, will then describe the implementation of a Proof-of-Concept of the system. This implementation aims to prove the viability of the overall system design, will serve as a prototype for tests and aims to implement all important features at least to such an extent that the full solution can be perceived as feasible. Furthermore, the Proof-of-Concept will also serve as a tool for requirements validation during real world test cases. The core criterion for the Proof-of-Concept is the ability to test the ideas behind all the use cases that have been described throughout the Synthesis. It therefore has to provide all the functionality that is necessary to test the viability of these use cases.

## 2 REQUIREMENTS ENGINEERING: THEORY

---

This section will focus on the theoretical part of the Requirements Engineering process. It will give a general overview of the topic, provide insight on the historical background, will then discuss state-of-the-art approaches and finally attempts to draw comparisons with modern Agile Software Development techniques.

### 2.1 A GENERAL PERSPECTIVE ON REQUIREMENTS ENGINEERING

Before the topic can be discussed in greater detail or even just in its historical context, a general overview is necessary to be able to put it into perspective. What exactly is Requirements Engineering? Why is it important? These are the questions that this section aims to provide answers to.

As the publication of the IEEE Computer Society Press Tutorial in 1990 (Dorfman, Thayer 1990) marked an important step in the history of Requirements Engineering, the definition by Dorfman and Thayer may be the most fitting to answer the first question:

*“Requirements Engineering is the science and discipline concerned with analyzing and documenting requirements.” (Dorfman, Thayer 1990)*

This, of course, begs the question of what exactly Requirements are. In short, a requirement can be seen as a property or functionality that a system must have in order to be of use for the user. Typically these requirements are then separated into two sub-categories: Functional and Non-Functional. The former describing actual features of the system, the latter leaning more towards conditions the system has to conform to. More detailed definitions can be found in section 2.3 below.

Typically, the Requirements Engineering process is divided into several sub-processes which on their own focus on specific parts of the overall process: Elicitation, Specification, Validation and Management (Rupp, SOPHISTen 2009). The first process focuses on the actual identification of requirements. This is normally achieved by either observing or interviewing the people that are involved in the project (e.g. the end-users). What follows the initial elicitation phase is typically a combination of negotiation and documentation. The Requirements Engineer aims to remove any incoherencies between different requirements



and might have to negotiate and find compromises as the involved parties typically aim to achieve goals which are at least slightly divergent.

When the requirements have been documented and the system is being implemented, it is necessary to validate them in order to ensure the system which is being built actually conforms to the specified requirements and they describe the desired end result.

As such a software system is typically never complete and new challenges arise throughout the development cycle, a management process is needed to be able to adapt to changing aspects.

When building a system, there are two main problems which have to be faced: Building the correct solution and building the solution correctly. In other words: What needs to be built and how it needs to be built. (Balzert, Balzert et al. 2009) Requirements Engineering focuses on the former.

## 2.2 ON THE HISTORY OF REQUIREMENTS ENGINEERING

Although the term Requirements Engineering seems to have appeared already relatively early on as part of a technical report by Alfor and Lawson (1979), it gained substantial significance with the IEEE Computer Society Press Tutorial released in 1990 (Dorfman, Thayer 1990). The aforementioned publication contains a large collection of early standards and guidelines on the topic of Requirements Engineering.

With regard to the surrounding research field – computer science – the two aforementioned years 1990 (Dorfman, Thayer 1990), and especially 1979 (Alfor, Lawson 1979) are a remarkably early point in time for the recognition of the significance of Requirements Engineering for the long term success of a software development project.

Especially since the first conference on Requirements Engineering in 1994 (Dorfman, Byrne et al. 1994) the number of publications per year increased greatly over time (see Figure 4).

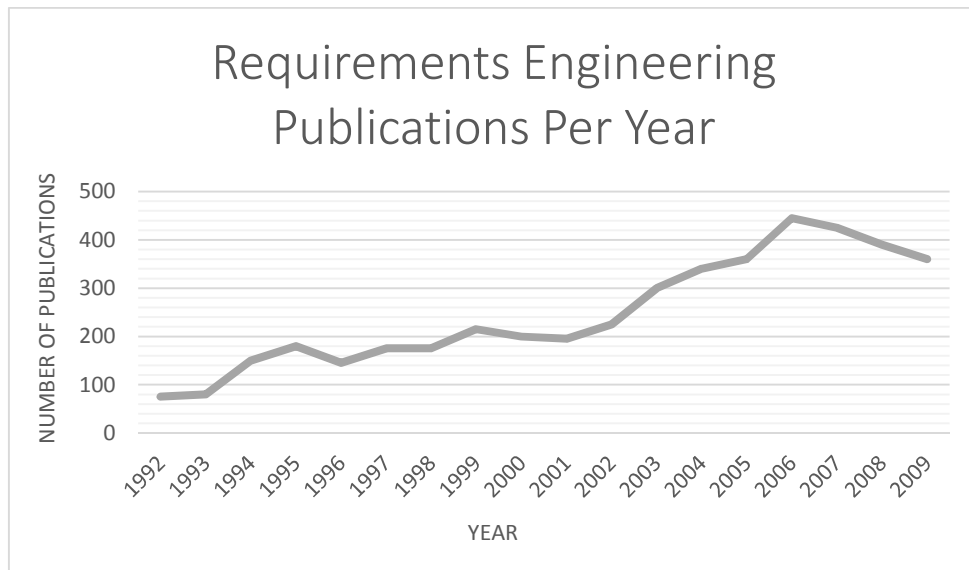


Figure 4: Requirements Engineering publications per year (figure modified from (Partsch 2010) which has been built utilizing the data provided by (Davis 2010)).

The slight decrease in recent years that can be seen in Figure 4 is explained by Partsch (2010) with the strong diversification the field underwent in the recent years which makes it difficult to compile an exhaustive list of all publications.

Another important step in the development of Requirements Engineering as a fundamental part of the software engineering process marked the foundation of the International Requirements Engineering Board (IREB). It provides publications, training and certifications (CPRE – Certified Professional for Requirements Engineering). The fundamental aim of this organization is the provision of universally acceptable standards with regard to training and qualification in the field of Requirements Engineering (IREB e.V., n.d.).

## 2.3 DEFINITIONS

In order to discuss different aspects of Requirements Engineering, it is first necessary to define the basic terminology. The following paragraphs will define the fundamental concepts in a short way for convenience and in a more elaborate way for completeness.

### 2.3.1 Requirements Engineering

For the sake of completeness, the definition that has already been given in the introductory paragraph of this section will be repeated here.

Requirements Engineering defines all activities that take place before and during system design and development which describe the capabilities, the resulting system must have. It does not, however, dictate the actual development and implementation in any way. In short, Requirements Engineering describes what a system should do, not how it should do it. (Partsch 2010) The industry recognized the significance of requirements for the software development process quite early on (Alfor, Lawson 1979, Dorfman, Thayer 1990). To tackle this rather complex task, the field of Requirements Engineering was born (Dorfman, Byrne et al. 1994).

With the creation of the term Requirements Engineering, the role of Requirements Engineer has been established as well – a person (or a group) who specifically focuses on the elicitation, documentation and management of requirements. A systematic approach to Requirements Engineering as well as highly developed communicative skills are regarded as key abilities for the Requirements Engineer, as he has to mitigate all conflicts and finally create a set of requirements all stakeholders can agree upon. (Rupp, SOPHISTen 2009)

### 2.3.2 Requirement

A requirement is a property or functionality a system must have in order to be useful to the user.

The IEEE defines *requirement* as follows:

- (1) *“A condition or capability needed by a user to solve a problem or achieve an objective.*
- (2) *A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.*
- (3) *A documented representation of a condition or capability as in (1) or (2).”* (IEEE Standards Board 1990)

In the field of Requirements Engineering, the term is typically sub-divided into two categories: **Functional Requirements** (see section 2.3.3 below) and **Non-Functional Requirements** (see section 2.3.4 below).

However, there is another categorization scheme, which is usually mentioned in various works on the topic and can even be combined with the functional / non-functional distinction mentioned above: The KANO-Model (Kano, Seraku et al. 1984). It provides the fundamental distinction between Dissatisfiers, Satisfiers and Delighters.

**Dissatisfiers** are features that are implicitly expected by the customer and therefore are often not even mentioned explicitly but are regarded as absolutely essential for the resulting system. The implementation of Dissatisfiers is therefore absolutely necessary but does not lead to any increase in customer satisfaction (e.g. the implementation of speech output in a car's navigation system). **Satisfiers** are the features that are normally implemented and negotiated between customer and developer (e.g. the calculation of the shortest route in a car's navigation system). They are directly proportional to the amount of satisfaction the customer gains out of the resulting system. Finally, **Delighters** are features, that may not be even mentioned by the customer but provide a significant increase in satisfaction (e.g. the option to display traffic in routes and to include this data into the route calculation in a car's navigation system). (Kano, Seraku et al. 1984, Pohl, Rupp 2011)



Figure 5: KANO-Model (figure modified from (Pohl, Rupp 2011)).

### 2.3.3 Functional Requirement

A functional requirement describes a capability or a feature of the resulting system: Something the system has to be able to accomplish.

Balzert et al. (2009) defines functional requirements as a function or service that the system, or one of its parts, has to provide.

Partsch (2010) provides a broader definition. According to him, functional requirements focus solely on the functional aspects of the system, thus describing the tasks of the system and the conditions it must fulfill to perform these tasks.

Pohl and Rupp (2011) again define functional requirements as those requirements that describe the functionality that the system has to offer. However, they provide an additional sub-categorization into functional, behavioral and data requirements.

In conclusion, it can be said that functional requirements describe what the system must be capable of achieving actively.

### 2.3.4 Non-Functional Requirement

Non-functional requirements are a little more difficult to define in detail as there are many definitions, some of them even a little contradictory. However, typically, non-functional requirements are regarded as conditions which the system must fulfill.

The most obvious definition, according to Rupp et al. (2009), of non-functional requirements would be simply every requirement that is not part of the set of functional requirements. (Rupp, SOPHISTen 2009) However, as this definition certainly is too diffuse for utilization during system planning and development, they provide another definition for which they divide the set of non-functional requirements into two general sub-categories: Requirements which directly affect the realization of the project and those who do not. They then provide different examples for non-functional requirements, such as: Technological boundaries, quality requirements or even legal obligations. (Rupp, SOPHISTen 2009)

The problem that has been stated in the initial short description of the term above is analyzed more closely by Glinz (2007). He provides a list of differing definitions that rely on many different publications on the topic. He then tackles this lack of a proper, universally accepted concept of non-functional requirements and even requirements in general, by the introduction of a completely new classification system which is described in greater detail in section 2.6.2 below. (Glinz 2007)

In summary, non-functional requirements can be understood as all those requirements that do not directly describe actual features of the system. Such requirements include factors like

technical constraints, legal obligations and requirements with regard to the environment the system has to be able to operate in.

### 2.3.5 Stakeholder

A stakeholder is a person who is directly or indirectly involved in the project. Typical examples are customers, developers, managers or even IT-Support staff.

Partsch (2010) defines the term a little more generally. According to him, a stakeholder is a party who is involved in the development of a future system and therefore demands the satisfaction of certain requirements. In addition to this, he also mentions that one stakeholder may have several different perspectives on the project and thus appears in more than one stakeholder-“role”. Adding to the examples already mentioned in the short description above, he provides an extensive list of stakeholder groups. (Partsch 2010)

Rupp et al. (2009) define stakeholder in a quite similar way but also propose a classification scheme for every identified stakeholder which depends on their potential influence and motivation. Their definition of the term focuses more on the potential to provide requirements rather than on their interests with regard to the overall resulting system (see section 2.5.1.2 below). (Rupp, SOPHISTen 2009)

Glinz and Wieringa (2007) define the term stakeholder not only as a person who is involved and therefore has an influence on the resulting system, but also as being passively impacted by it. (Glinz, Wieringa 2007)

In summary it can be said that the group of stakeholders includes all parties that are directly or indirectly involved either in the development or the usage of the system.

### 2.3.6 System, Context and Environment

The differentiation between System, Context and Environment is regarded as fundamental for the analysis of already established systems and the design of future solutions. (e.g. (Pohl, Rupp 2011, Partsch 2010, Rupp, SOPHISTen 2009)

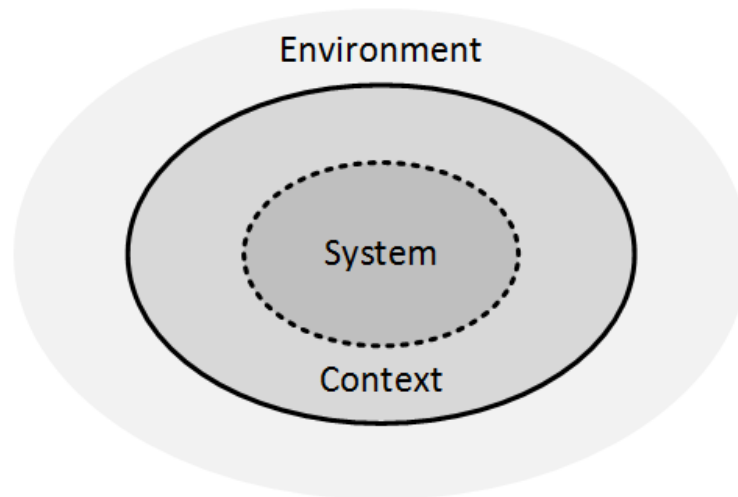


Figure 6: System, Context and Environment (figure modified from (Pohl, Rupp 2011)).

**System** describes all parts that directly belong to the system (e.g. hardware and software components).

**Context** includes everything that is of relevance for the system's requirements, but does not directly belong to the system (e.g. users, interfacing modules, etc.).

Finally, the **Environment** denotes everything that is neither part of the system nor its context and therefore is irrelevant for system development.

Based on (Rupp, SOPHISTen 2009, Partsch 2010, Pohl, Rupp 2011)

## 2.4 GENERAL REMARKS ON REQUIREMENTS ENGINEERING APPROACHES

This section will discuss general commonalities between different Requirements Engineering approaches. In addition to this, a short overview of typical Requirements Engineering processes is given.

Generally, Requirements Engineering can be subdivided into the following five fields (Paetsch, Eberlein et al. 2003): Elicitation, Analysis, Documentation, Validation and Management.

However, there are some approaches that differ slightly. For example, Rupp et al. (2009) add a Pre-Elicitation phase (see section 2.5.1 below) that contains an extensive AS-IS-Analysis of any system that might be in place before the development of a new system, or, if there is not, of the processes as they are currently carried out. In addition to this, they propose a special method to sanitize identified requirements before documentation (see 2.5.3 below).

Regardless of the specific implementation of Requirements Engineering, a small set of processes seems to be essential and appears in all of them (Paetsch, Eberlein et al. 2003, Rupp, SOPHISTen 2009, Partsch 2010). This set consists of:

- An **identification phase**, where requirements are determined for the first time (typically called “Requirements Elicitation”)
- A **documentation phase**, where the identified requirements are then written down in a formally specified manner (typically called “Documentation”)
- A **validation phase**, where the requirements are validated with regard to consistency between what has been planned and what is actually necessary to implement the system properly.
- And a **management phase** which accounts for the fact that software usually is considered to be constantly evolving (e.g. in order to react to changes in the environment).

It is worth mentioning that in traditional software development methods (e.g. the Waterfall Model) and thus traditional (linear) Requirements Engineering, these steps are considered to be sequential and directed. This means that after an initial Elicitation and Specification-Phase, the system is implemented in its entirety. The problem with this approach is obvious: If the underlying task, the environment or even the solution is too complex or dynamic to be planned out completely beforehand, the approach is not viable. To account for this, different software development models have been introduced which allow for free movement in both directions of the sequence (e.g. the well-established V-Model). In addition to this, somewhat as a counter movement to very elaborate planning processes, so-called “Agile Software Development” methods have been proposed (see section 2.7 below). These approaches focus on very short development cycles and strong involvement of the customer instead of meticulous planning phases in the beginning of the development process.

The differences and potential synergies between Requirements Engineering and Agile Software Development methods will be discussed further down in the document in section 2.7.4 below).



## 2.5 STATE OF THE ART REQUIREMENTS ENGINEERING

This section aims to provide a theoretical overview on how Requirements Engineering processes are typically utilized in current projects. There are many different implementations of the standard Requirements Engineering-Process, as has already been outlined in section 2.4 above. Because of this reason, this part will mainly focus on the standard aspects of Requirements Engineering and those which have been introduced by Rupp et al. (2009), as their approach differs slightly in a few key aspects and has been used for the applied Requirements Engineering in section 3 below. Processes which are unique or implemented differently in the SOPHIST-Approach are explained separately in the sections that follow.

As has already been mentioned in the general description, the SOPHIST-Approach (Rupp, SOPHISTen 2009) focuses not only on the traditional parts of Requirements Engineering but also on an extensive AS-IS-Analysis before the building of the system and additionally offers insight on efficient interview and observation methods. This strong focus on the pre-elicitation phase is not typical to other approaches and will be described in the next section.

### 2.5.1 Pre-Elicitation Phase

The SOPHIST-approach (Rupp, SOPHISTen 2009) differs from the typical approach to Requirements Engineering mainly by the strong focus on the AS-IS-Analysis of previously implemented systems and processes before the new system is developed or even planned.

There are three distinct phases before the actual requirements are analyzed: AS-IS-Analysis, Identification of stakeholders and definition of goals. (Rupp, SOPHISTen 2009) These phases will now be explained in greater detail.

#### 2.5.1.1 AS-IS-Analysis

This phase focuses primarily on the systematic analysis of the system as it is in place before the new system is developed. It is also used to identify problems in processes as they are currently carried out, which can then later be used as a basis for the elicitation phase. There are many different ways to model such processes (e.g. Sequence-Diagrams or Flow-Charts). The technique that will be used further down in this thesis is the Business Process Model and

Notation (BPMN). The resulting diagrams can then be used to model the process as it is currently implemented and to identify potential problems with existing routines.

#### 2.5.1.2 Identification of Stakeholders

Identifying the central people involved in a project is extremely important. However, the SOPHIST-approach (Rupp, SOPHISTen 2009) takes great care to identify and categorize all involved stakeholders in a very detailed manner, e.g. by using tables with additional fields for information like the relevance or availability of the stakeholder (Rupp, SOPHISTen 2009). This then allows for a quick identification of those parties that can be involved strongly in the development process and other parties who may not be able to invest as much time.

#### 2.5.1.3 Definition of Goals

Before the elicitation phase can begin, a general list of project goals is devised. This list is then used to provide an overall guide as to where the final system is headed (cf. Specification Level 0, section 2.5.2 below).

### 2.5.2 Elicitation Phase

In every Requirements Engineering process there is one phase where the requirements are initially gathered for the first time.

To account for the different types and interests of stakeholders, the authors of the SOPHIST-approach propose a number of different specification levels (Rupp, SOPHISTen 2009):

0. Very abstract specification, Goals, Visions
1. Use Cases, User Stories, Feature List
2. Actual Requirements, Test Cases, Feature List
3. Detailed Requirements, Technical Requirements, Interfaces, Test Cases, Feature List
4. Technical Requirements, Interface Description, Test Cases, Modules

Requirements classified under level 0 are more abstract visions and overall project goals than actual requirements. They are typically used to convey a clear, though not very detailed, picture of the complete system and its capabilities. Specification level 1 then refines these overall ideas into use cases.

The requirements, as they are typically understood, reside somewhere between levels 2 and 3. Level 4 focuses on the actual technical implementation and is therefore not essential for most parties who are involved in the process (with an exception of programmers, designers and other people who are directly involved in the development process).

For the actual elicitation of requirements, a number of different methods can be used. The remaining part of this section will focus on a small subset, which has been used in the applied Requirements Engineering following in section 3 below.

### 2.5.2.1 Interviews

The interview is arguably one of the most obvious choices for obtaining information. The big advantage of this method is the ability of the Requirements Engineer to steer the discussion into directions that seem to be the most promising ones with regard to the amount of information that can be gathered. (Rupp, SOPHISTen 2009) However, interviews are typically extremely hard to schedule as, especially if multiple stakeholders are required to be present in the same interview, the involved people usually have only very little time to spare.

If, however, an interview can be scheduled, it provides an excellent way for the Requirements Engineer to get the answers he needs.

To be completely sure that everything has been understood correctly, an interview protocol should be created by the Requirements Engineer, which can then be sent to the participating stakeholders, allowing them to read through it once again and approve of the contents. Rupp et al. (2009) suggest a timeframe of a maximum of 48 hours between the actual interview and the sending of the protocol.

Audio and video recordings can be of help, but need to be discussed with the involved stakeholders first. In addition to this, it has to be considered that answers and involvement of stakeholders might be hampered by the fact that they are recorded.

### 2.5.2.2 Questionnaires

Very similar to interviews, questionnaires are a widely used method for obtaining information. The big advantage over the interview is that there is no need to schedule appointments, as everyone involved can answer the questions whenever his schedule allows it. This inherently

includes the disadvantage of not being present during the answering of the questions and therefore providing clarification, if there are issues of understanding the content or intent of the questions, and steering the discussion is, of course, not possible. (Rupp, SOPHISTen 2009)

Questionnaires are a list of either open questions, or questions provided with multiple answer-possibilities which can then be sent to the stakeholders.

### 2.5.2.3 Observation

Observation has proven to be a very important technique for Requirements Engineering in the applied part of this thesis for two reasons:

- 1) Stakeholders often cannot afford to offer time for interviews. Scheduling appointments where all stakeholders who are essential for the topic are available is a very time consuming process on its own.
- 2) The possibility to observe processes as they are carried out by the stakeholders during their daily routine often reveals details and aspects that would not come up in interviews, as the stakeholders are extremely familiar with these routines and regard them as unimportant or simply forget to mention them.

In addition to this, the possibility of taking pictures during these processes can prove to be vital as a basis for further discussion in interviews and for documentation.

There is a sub-category for the traditional observation, which is called Apprenticing. (Rupp, SOPHISTen 2009) The method behind this approach does not solve the two problems outlined above, as the stakeholders must be available and have to actively teach the observer the actions they carry out during the process. The idea behind this method is that anything that might be omitted or that is not entirely clear would be noticed immediately as the apprentice – the Requirements Engineer – would have issues learning it. However, this method is only used rarely as it requires a lot of time both from the Requirements Engineer and the stakeholder, which at least the latter might not be able to offer. In addition to this, the approach is impossible to use in some scenarios or environments (e.g. medical environments – teaching a Requirements Engineer how to perform heart-surgery is absolutely impossible).

### 2.5.3 The SOPHIST-REgelwerk for Requirement Sanitization

After the requirements have initially been identified, they are specified in a formal way. However, one very interesting aspect to the SOPHIST-Approach is the method which they propose to sanitize requirements systematically before documenting them properly.

Typically requirements are documented and certainly gathered in natural language (see section 2.5.4 below). The big advantage is that all involved parties, especially the stakeholders, do not need to have any experience in order to understand them. (Robertson, Robertson 2006) However, this is one of the potentially biggest problems of Requirements Engineering. Since many decades, natural speech recognition and creation are regarded as very hard problems in the field of computer science. What makes this field so extremely difficult is the fact that this problem actually contains a multitude of sub-problems which are each very hard to solve in their own right. Natural language is not exact, it contains sub-text, hidden meanings and complicated constructs. In fact, there is even a field of research that focuses solely on the problems that result out of this complexity: Natural Language Processing (NLP).

Apart from the problems that arise out of the language used, it is also necessary to gather information exhaustively. The SOPHIST-REgelwerk (Rupp, SOPHISTen 2009) accounts for these shortcomings by providing 18 rules, which, in theory, should lead to a well-formed and well-defined requirements catalogue. There are several different aspects that are covered by these rules. For example, a subset focuses on the analysis of numerals in requirements in order to avoid specifications which are unspecific or too general. Another aspect that is paid close attention to by the SOPHIST-Approach is the usage of what they call “good” main verbs. They propose a number of different heuristics to identify such verbs and additional guidance with regard to their usage, such as reducing the amount of them to just one per sentence while still keeping the possibility of comprising a requirement out of multiple sentences. However, most importantly, the analysis and preparation of requirements with respect to their factual specificity and exhaustiveness is stressed repeatedly amongst these rules. Requiring all the information that is necessary to exhaustively describe a system requirement is regarded as a key factor in the Requirements Engineering process according to the SOPHIST-REgelwerk. This not only includes the exhaustive analysis and specification of data, but also the removal of any redundant information. (Rupp, SOPHISTen 2009)

Especially with regard to this aspect, there are many other very popular techniques which can be used to formulate matters of fact exhaustively.

One example would be the MECE-Framework which is used by McKinsey operatives (Rasiel 1999). MECE stands for **M**utually **E**xclusive,

**C**ollectively **E**xhaustive (see Figure 7), describing a way to convey information that covers a topic to its entirety (collectively exhaustive) while splitting the sub-topics in such a way that no intersections occur (mutually exclusive). (Rasiel 1999)

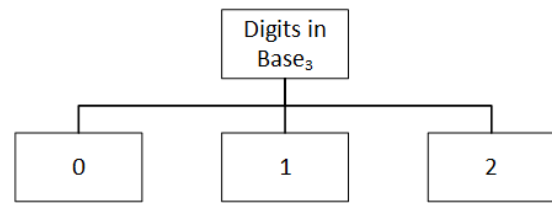


Figure 7: Typical MECE-Tree. The term at the top is split into an exhaustive list of non-overlapping sub-elements.

## 2.5.4 Requirement Specification

At some point during every Requirements Engineering process, the actual requirements have to be formally specified and documented.

The most obvious and natural way would be to just document them as they are mentioned by the stakeholders: In natural language. However, natural language is not always the best way, as it is easy to formulate requirements that are too unspecific for system planning.

The logical next step would be to highlight specific parts of the text to increase readability and to emphasize the important aspects of the requirement. (Partsch 2010) However, this does not solve the problem explained above – it is still too easy to define requirements that have no apparent value for system planning.

Moving away from natural language, but keeping text as the basis for the requirements specification, another method would be using specific formulas or patterns to facilitate comparisons and readability. One other big advantage of this approach is that there are certain aspects, which cannot be left unspecified in the description of the requirement (e.g. if the implementation of the requirement is legally binding or not).

Rupp et al. (2009) provide a simple way to achieve this goal by specifying a pattern which is used to formally construct textual requirements (see Figure 8):

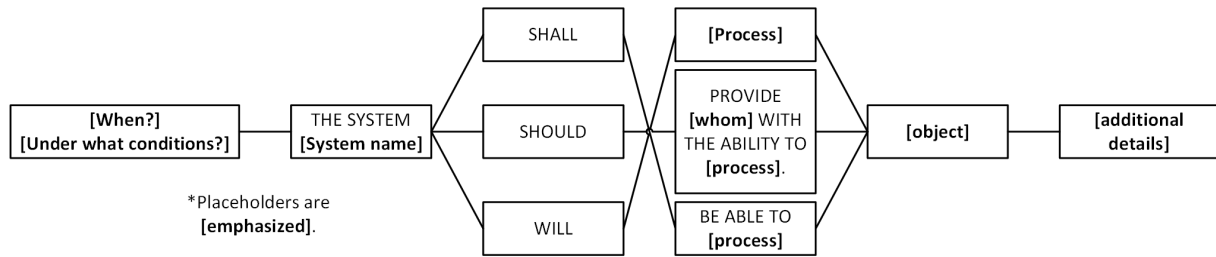


Figure 8: Requirements formula used in this thesis (figure modified from (Rupp, SOPHISTen 2009)).

The schema shown in Figure 8 ultimately leads to requirements as they have been utilized further down in the document (see 3.9 below). It assures that each requirement, if it is documented in such a formalized textual way, includes all necessary elements. This includes, for example, any legal obligations that are attached to the requirement. For example, if, as the third part, “SHALL” is chosen, the developer is legally obliged to implement the requirement (it is part of the contract). If “SHOULD” is used instead, it does not have to be implemented necessarily (this becomes especially interesting, when deadlines are strict and requirements get prioritized).

Although textual constructs are widely popular for the documentation of requirements, there are other ways to describe the desired system behavior. (Partsch 2010) mentions a number of different methods, including interaction- and sequence diagrams (see Figure 9).

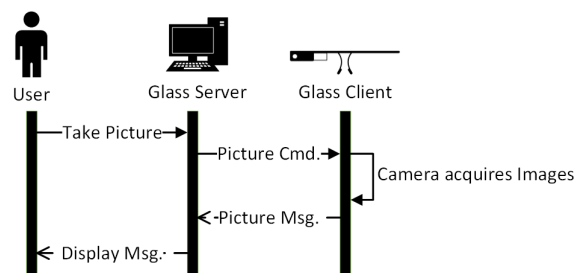


Figure 9: Typical sequence diagram as used for Requirements Engineering by Partsch (2010).

### 2.5.5 Requirement Validation

Requirements validation analyzes the documented requirements with regard to consistency to the system the stakeholders expect. In collaboration with all involved parties and the list of requirements, any remaining problems will be identified and an attempt of solving them will be made. (Paetsch, Eberlein et al. 2003)

For inadequate requirements, two categories can be devised - **Faults** and **Defects** (Rupp, SOPHISTen 2009):

- A requirement is defective, if it does not describe the desired system behavior adequately (or fully).
- A requirement is faulty, if it describes a desired system behavior wrongly.

Similar to the problem outlined in Figure 1, late changes to requirements are more costly to fix than early changes. (Partsch 2010) As a tool of ensuring the quality and validation of requirements, Rupp et al. (2009) list a few different methods. Two of them, **Reviews** and **Prototypes**, will be discussed in greater detail in the remainder of this section (Rupp, SOPHISTen 2009):

#### 2.5.5.1 Reviews

Reviews are a rather simple method for requirements validation. Typically, they are subdivided into the following three subcategories (Pohl, Rupp 2011, Rupp, SOPHISTen 2009):

- **Commenting**  
Commenting is certainly the easiest method for requirements validation. The list of identified and documented requirements is given to a third-party, who reads them carefully and checks for consistency and other issues. Any identified problems will be reviewed by the Requirements Engineer and then fixed in the document. It is worth mentioning, that this method greatly depends on the skill and experience of the third party.
- **Walkthrough**  
The walkthrough is a little bit more complicated. Instead of just giving a list of requirements to a third party, the Requirements Engineer also explains the reasons for the choices he made, thus establishing a common understanding between him and the third-party. In addition to the advantages and disadvantages of Commenting, the Walkthrough provides the ability to solve any misunderstandings during discussions, but requires significantly more time from all involved parties.



- **Inspection**

Inspections are a very formal way of validating requirements, typically including multiple third-parties (inspectors) and formal check-lists.

#### 2.5.5.2 Prototypes

Prototypes are implementations which provide an incomplete overview over the desired functionality to offer the involved stakeholders a better perspective on certain aspects of the resulting system. Especially with regard to user interface and overall usability, prototypes are a valuable method to gather further information by demonstrating functionality to the stakeholders. Implicit requirements, which did not become apparent during interviews or observations (especially with regard to non-functional requirements like system-responsiveness) can be analyzed easily by demonstrating a prototype. The Proof-of-Concept that has been provided for the system designed further down in this thesis (see section 5 below) is such a prototype which proved to be valuable as a tool especially for the elicitation and validation of non-functional requirements, such as image quality and frame rate. However, prototypes are extremely time consuming to implement and must be clearly communicated as unfinished versions to the stakeholders. For the aforementioned reasons, prototypical implementations are not always suitable for requirements validation. (Rupp, SOPHISTen 2009)

#### 2.5.6 Requirements Management

Requirements Management primarily describes the process behind managing the requirements with regard to simple organization and to changes in the project that require updates or new requirements.

Requirements Management is helpful in small, short-lived projects and it is absolutely essential in big, complex projects, especially ones that take years to develop. Rupp et al. (2009) even suggest using a dedicated Requirements Management tool to facilitate coping with a rapidly growing number of requirements and their versions.

The central aspects of Requirements Management according to Rupp et al. (2009) are:

- **Information Flow:** Exchange of information between all involved parties.
- **Sequence and Activities:** Roles and Responsibilities during the phases of the project.
- **Interconnectivity:** Relations between different Requirements Engineering elements.
- **Analysis and Assessment:** Detection of potential issues.

Especially interconnectivity and the overall information flow become very complex to handle without a specialized tool as soon as the project grows past a certain point. The market with regard to such tools (e.g. TopTeam Analyst, ReqMan, Psoda, PACE, etc.) has grown since Requirements Engineering first gained significance, which can make it difficult to choose between them. Carrillo de Gea et al. (2011) conducted a study on the topic of current Requirements Engineering tools and found that especially the traditional software solutions lack flexibility with regard to management and traceability of requirements. It is their opinion that the market of these tools is changing fast and careful consideration, especially when it comes to the sharing of data between different teams (and therefore different Requirements Engineering tools) has to be taken before choosing a solution (Carrillo de Gea, Nicolas et al. 2011).

One technique to facilitate coping with complex requirement lists that is described by Rupp et al. (2009) are Object-IDs. Every requirement receives an identifier which makes it easy to find within the requirements document. The Object-IDs typically are not only IDs but UUIDs, meaning at least some part of the identifier is unique. This is especially useful, if the requirement changes later on, as the unique part of the identifier still allows the Requirements Engineer to find the requirement and to track it during the course of its development. The method that is used further down in the applied Requirements Engineering-Part of this thesis (see section 3.9.1 below) contains not only a unique identifier, but also information with regard to the type of requirement (Functional, Non-Functional and even Hardware) and the severity of it (obligatory, optional and obligatory in future version).

As has already been mentioned above, requirements can change during development. This can be interpreted with regard to two aspects: Content and Status.

When a requirement changes its content, it is because it either did not describe a certain aspect of the system in its entirety, or it described an aspect wrongly (see section 2.5.5 above).

The status changes constantly during the life cycle of the requirement. Rupp et al. (2009) mention the following exemplary phases a requirement can go through: **Created** (see section 2.5.2 above), **Analyzed** and **Quality-Assured** (see section 2.5.3 above), **Designed, Implemented, Tested** and **Accepted** (see section 2.5.5 above). In addition to this, they stress the necessity of versioning during the Requirements Engineering process. This means that for every change, the previous version must be kept safe to allow for full traceability of changes. (Rupp, SOPHISTen 2009)

## 2.6 CRITICISM

As most established processes, Requirements Engineering, especially in its traditional (linear) form, has also been questioned and criticized by many (e.g. (Ralph 2013, Glinz 2007)). This part of the thesis will give a short overview over some key aspects, where criticism can be found and will then discuss an approach that has gained significant acceptance throughout the development community especially in the past two decades: **Agile Software Development**.

### 2.6.1 Psychological Factors with regard to Requirements

As has already been mentioned in the beginning chapter of this section, (linear) Requirements Engineering is a rather traditional approach and therefore is often considered to offer too little flexibility. But not only has this perceived inflexibility been criticized in the past, some go even further and question the very essence of the Requirements Engineering Process – the requirement. According to Ralph (2013), the biggest problem with requirements is the psychological factor and what he perceives as the lack of a clear specification of the term requirement. For example, according to his argumentation, the term “requirement” suggests a feature which is absolutely obligatory, whereas in reality, prioritization and optional requirements are used to circumvent such constraints. (Ralph 2013)

### 2.6.2 The lack of a clear definition of Non-Functional Requirements

A different perspective on the problem is described by Glinz (2007). According to his argumentation, another apparent problem in the Requirements Engineering process can be

found: The lack of a proper concept of requirements and specifically non-functional requirements.

This unclearness in the concept behind the term is then tackled by him by the introduction of a completely new classification scheme. This interesting approach keeps the term requirement at its center but at the same time ignores all well-known sub-categories (e.g. functional and non-functional requirements) and replaces them with four criteria: **Representation, Kind, Satisfaction** and **Role**. These categories are then used to determine the nature of the requirement. Apart from the inherent lack of clearness with terms like non-functional requirements, this method solves another problem which Glinz calls representation-dependence (e.g. a functional requirement can be a non-functional requirement on another abstraction level). (Glinz 2007)

## 2.7 A COMPLETELY DIFFERENT APPROACH: AGILE SOFTWARE DEVELOPMENT

As an anti-thesis to the problems that mainly appear with very linear software development techniques, a completely new approach has gained significance in the past two decades: **Agile Software Development**.

Instead of focusing on the specification and documentation of a system, agile approaches accept software as constantly evolving and therefore welcome change during the development process. In order to be able to react on changing requirements and customer needs, development cycles are very short, typically less than a month, and releases are frequent. This and the very strong involvement of the customer leads to a highly agile development process where changes can be implemented quickly and the customer has the ability to follow the development process closely.

Where Requirements Engineering values established processes, exhaustive documentation and systematic approaches, Agile Software Development takes an entirely different approach:

**Manifesto for Agile Software Development**

We are uncovering better ways of developing software by doing it and helping others do it.  
Through this work we have come to value:

**Individuals and interactions** over processes and tools  
**Working software** over comprehensive documentation  
**Customer collaboration** over contract negotiation  
**Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

© 2001, the above authors  
this declaration may be freely copied in any form,  
but only in its entirety through this notice.

*(Beck, Beedle et al. 2001)*

According to the Agile Manifesto cited above, Agile Software Development has a number of core principles:

Instead of relying on established processes and complex tools, direct communication in small teams is encouraged. Release cycles are extremely short and every build must be complete and functional. In addition to this, Test-Driven development is regarded as a better way to document code than a traditional, written documentation, that poses the risk of being outdated (of course, test cases can be outdated too, but they will stop functioning and therefore will be noticed by the programmer). Finally, instead of complex contracts, Agile Software Development tries to push in a direction where the customer is directly involved in the software development process and therefore builds trust as he can intervene at any given point in time. One of the most important factors in Agile Software Development practices is the fact that change is not feared but rather embraced. Software is not planned exhaustively in the beginning, it is developed from milestone to milestone.

However, this is of course no Silver Bullet. Agile Software Development definitely has its place in the software industry, but there are projects, where traditional methods (e.g. the V-Model)

are the better choice for the task. Examples would be projects that involve very strict guidelines, or which are very large. (Haberfellner, de Weck et al. 2012)

As already mentioned above, Agile Software Development approaches can be understood as the anti-thesis to the traditional (linear) software development process. However, efforts have been made to draw comparisons between the two approaches and to identify possible synergies by combining certain aspects. Software Development is not an exact science. There are many ways to reach the same goal. Very systematic development approaches focus on exhaustive planning in the beginning of the project, agile methods have their strengths in the ability to react to changes. But there are also cases in between, where a combination of the two approaches might become a fruitful strategy. These combinations, or rather the exploration of how both fundamentally different approaches can influence each other, will be discussed further in the sections below.

### 2.7.1 A Short Introduction to Agile Software Development Techniques

In order to be able to discuss advantages and disadvantages of both approaches, it is necessary to briefly mention the most important contenders in the Agile Software Development field.

Generally, agile techniques try to distance themselves from traditional methodologies. Especially very systematic approaches like the traditional Waterfall Model or – to a lesser extent – the V-Model are regarded as inflexible.

Instead of fixed planning and conceptualization processes in the beginning of the development cycle, change as a constant factor is accepted and even welcomed. This leads to highly agile processes where cycles have to be short and releases frequent in order to be able to minimize potential costs which arise out of the introduction of changes by factors like the environment or the stakeholders.

As has already been mentioned in the paragraph above, there are a few properties which appear in most agile methods:

- Short release cycles
- Frequent meetings

- Little Documentation
- Strong involvement of the customer
- The reduction of communication barriers within the team
- Prioritization

#### 2.7.1.1 Extreme Programming

Extreme Programming is one of the two agile approaches which can be found time and time again in the software development industry. And it is indeed *extreme* in several aspects.

Extreme Programming has its origin in the Chrysler Comprehensive Compensation System (C3) project in the late years of 1990. Kent Beck is widely regarded as the driving force behind the phenomenon and was hired by Chrysler to optimize team performance in the C3-project, during the course of which he formulated the common Extreme Programming (XP) practices. (Copeland 2001) Beck then proceeded to provide insight to this new development process by publishing a guide to XP in which he explains the fundamental processes behind the approach. (Beck 1999)

XP combines many different aspects (see Figure 10) that are common to agile software development approaches, e.g. the use of Refactoring, short release cycles, continuous integration with some aspects that might be considered extreme, like the availability of a constant *on-site* customer, a very strong reliance on testing and a process called *pair-programming*. With the strong enforcement of constant testing, the resulting development process typically involves two programmers per workstation, alternating between writing test cases and developing the software. This achieves multiple things: Firstly, a code review is done by the other programmer while the code is being written, secondly, test cases document the system and are written before the code by the other programmer (forcing the test cases to be written as meaningful, correct and self-explanatory as possible) and finally, a collective knowledge base is created where everybody in the team becomes familiar with most aspects of the system thus minimizing the risk posed by project members leaving the team and taking valuable knowledge with them.

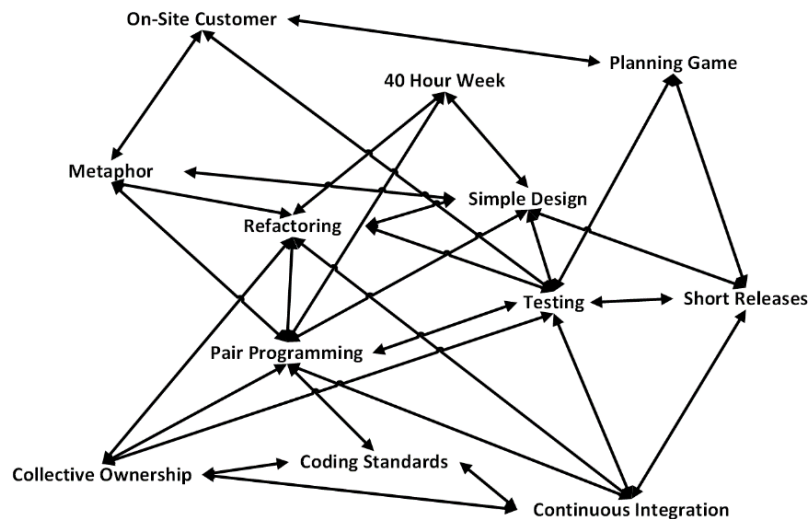


Figure 10: The inter-relationship between the elements of XP (figure modified from (Beck 1999)).

Beck goes even as far as calling the XP teams “intellectual nomads” who are constantly prepared to move, are highly flexible and are used to “traveling light”. (Beck 1999)

Another key aspect of XP is the so-called **Planning Game**. Beck (1999) divides it into three distinct phases: Exploration, Commitment and Steering. In the first phase, **Exploration**, the customer creates so-called “Story Cards”. These are sheets of paper that contain stories describing interactions with the system from the viewpoint of a specific user. Now, the development team estimates the effort that is required for the story to become reality within the system. If the team is unable to estimate the story, the creator of the Story Card has to *split* it into lesser complex sub-stories. The second phase, **Commitment**, then is used to prioritize the written and estimated story cards with regard to their value (customer) and risk (development). Because of the continuous estimation of story cards, the development-team is able to devise a *project velocity* based on the time frame of the last iteration and the amount of stories completed along with their estimated effort. This velocity is then used to either estimate a date for the amount of story cards the customer chooses, or to estimate which story cards can be realistically implemented until a deadline given by the customer. The third phase, **Steering**, is then used to make alterations during development. These changes can result out of bad estimations (features reveal a more complex side that was not part of the consideration during the Exploration-phase), new user stories or similar actions. (Beck 1999)



The combination of short release cycles, the planning game, very strict rules with regard to testing and the close involvement of the customer lead to a highly flexible software development process.

Clearly, not every company can afford to double the number of developers for pair-programming. Near-complete unit test coverage is also very difficult to achieve and it certainly is hard to explain the lack of documentation (substituted by tests and knowledge transfer) to the management department. Extreme Programming is what the name implies: Extreme. Precisely because of this reason, there is another very popular Agile Software Development approach which is even more widespread than XP: **Scrum**.

### 2.7.1.2 Scrum

When a software development company or department calls their process agile, in the vast majority of cases there is some variation of Scrum involved (see Figure 11). (VersionOne Inc. 2014)

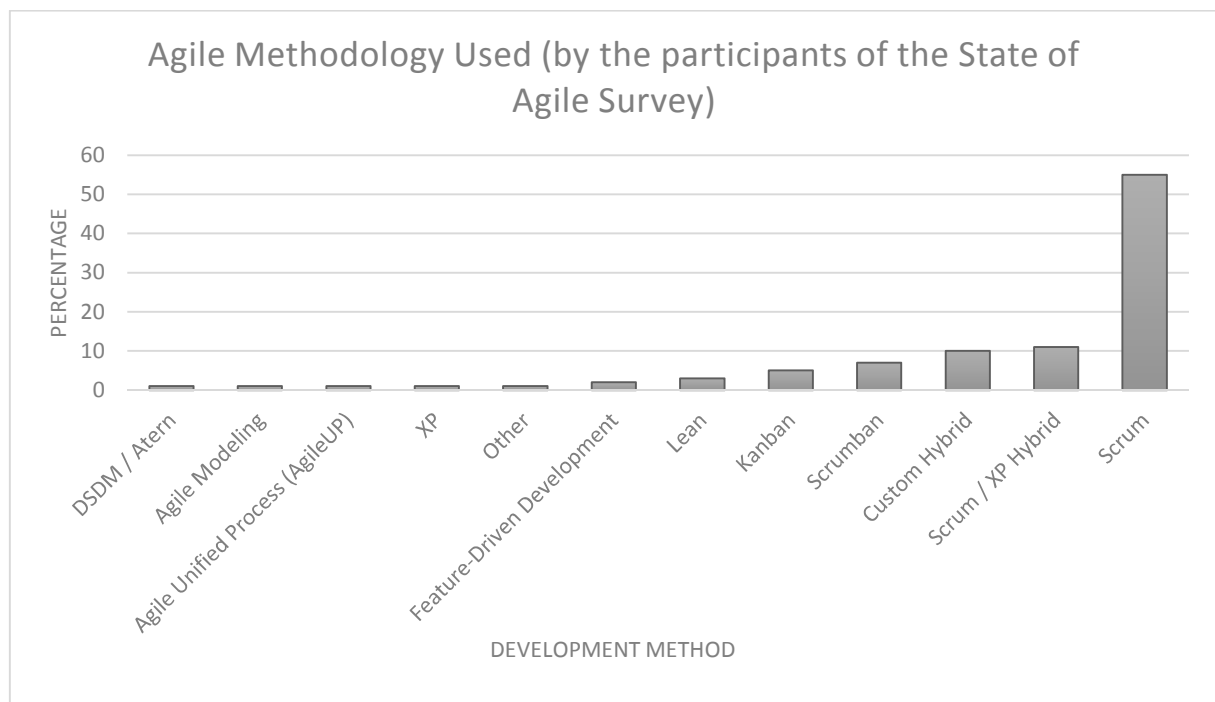


Figure 11: Scrum (and its variants) are the most adopted agile development approaches according to the State of Agile Survey (figure modified from (VersionOne Inc. 2014)).

In the remaining part of this section a brief introduction to the Scrum-process and its most important aspects will be given. The following explanations of the process are based on the

official **Scrum Guide** (Schwaber, Sutherland 2013) by Ken Schwaber and Jeff Sutherland who are regarded as the founders of the Scrum approach. (Schwaber 1997)

In general, there are three different roles within the Scrum approach

- The **Product Owner** manages the **Product Backlog**, a central list which includes all features of the currently planned product release.
- The **Developers (and Testers)** are responsible for estimation and implementation of the features described in the Product Backlog.
- Finally, the **Scrum Master** ensures that all processes run smoothly and the Scrum approach is implemented properly within the team.

The core principle behind the development process is called **Sprint**. These sprints are very short time periods, typically ranging between one and four weeks in duration, where development takes place.

At the beginning of each sprint there is a meeting called "**Sprint Planning**" where the tasks for the coming build are chosen by the team. A central aspect of the Scrum approach is the fact, that every build is fully functional (although not feature complete, as the Sprint Backlog is only a subset of the Product Backlog) at the end of the sprint and would potentially be shippable. During the development period, all team members keep track of every work item that is currently in development or that has been finished. An overview over this data is often provided by so-called Burndown Charts (see Figure 12), which serve as a tool for future estimations and help to identify problems quickly during development.

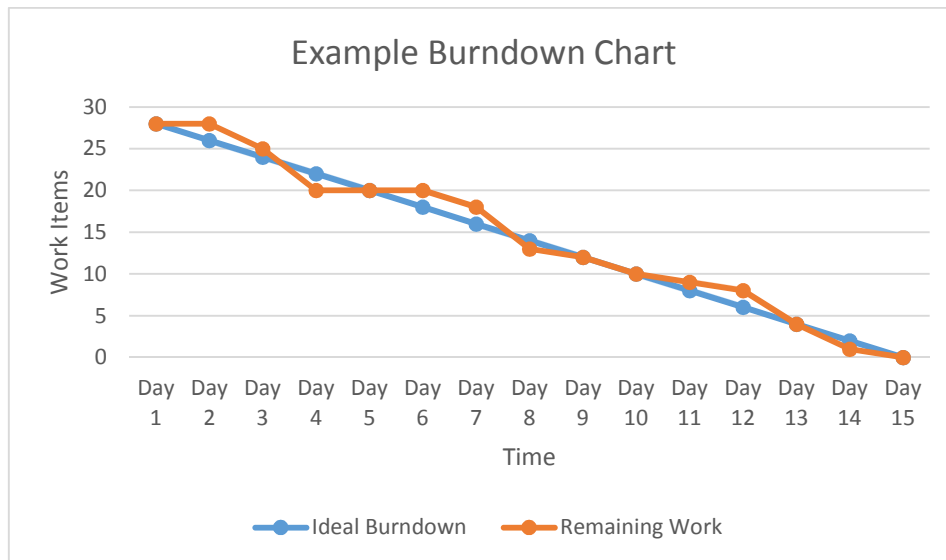


Figure 12: Exemplary Burndown Chart - a central SCRUM tool that appears in many tools which support Scrum (e.g. Visual Studio 2013 (Microsoft Corporation 2013))

At the end of each sprint there is a **Review Meeting** which reflects on all the work that has been achieved during the sprint. The current system build is typically demonstrated to all involved stakeholders as well. At this point, significant changes can be made.

Where the review meeting focused on the product, the **Retrospective Meeting** focuses on the team. It is used to identify potential for improvement with regard to the team and the development process.

The final core principle is the **Daily Scrum** which gives the development team 15 minutes at the beginning of each day to discuss what has been accomplished during the day before, which problems arose and what is planned for the current day. This again is a tool for quickly identifying problems within the development process and also serves as a means for building trust within the programming team.

As with all agile development methods, the short development cycle allows for increasing accuracy in estimating the effort that has to be committed to the project. This, along with tools such as the Burndown Chart, which shows the project's progress over time, typically increases moral and motivation within the team and builds trust on the side of the customer.

Based on (Schwaber, Sutherland 2013)

### 2.7.2 Natural Emergence of Requirements in Agile Software Development Teams

A study conducted by Lan and Ramesh (2008) explored the natural emergence of requirements and the utilization of Agile Requirements Engineering techniques in projects and companies who embrace Agile Software Development.

During the course of the study, they identified seven Agile Requirements Engineering practices (Lan, Ramesh 2008):

- Face-to-face communication over written specifications
- Iterative Requirements Engineering
- Requirement Prioritization goes extreme
- Managing requirements change through constant planning
- Prototyping
- Test-Driven Development
- User review meetings and acceptance tests

Ultimately, their findings were that the Agile Requirements Engineering used in these companies had a highly iterative form, contrary to the traditional approach which aims to plan the solution in its entirety before the development process starts. In addition to this, exhaustive documentation has been perceived as less important than the close interaction between developers and customers. Perhaps not surprisingly, these approaches emerged in environments where the creation of traditional requirements specifications is inappropriate or even infeasible and therefore a more agile approach is necessary. This, for example, is the case with small projects and teams where knowledge can be passed around quickly from employee to employee via personal interaction and where a high amount of flexibility is needed in the development process. (Haberfellner, de Weck et al. 2012)

However, they do also state that Agile Requirements Engineering cannot replace traditional Requirements Engineering completely as for every benefit found for the seven practices listed above, disadvantages (or challenges, as they are called in the study) have been identified. Thus, they advocate careful consideration before choosing agile approaches over the traditional ones, as they can yield a significant benefit but might also hamper the development process when used for the wrong project (Lan, Ramesh 2008).

### 2.7.3 Introducing Agile Aspects to Requirements Engineering

The previous chapter focused on the use of Requirements Engineering techniques in already established Agile Software Development processes. Waldmann (2011) tackles the problem from a different perspective: Can agile aspects be introduced into established Requirements Engineering processes?

The reasoning behind this is the very same one that initially led to the development of agile approaches: The environment changes too fast and typically there are not enough resources to solve the problem in its entirety, which then leads to prioritization and similar techniques. The same is, of course, true as well for Requirements Engineering. Especially very complex projects are hard to plan ahead completely, Requirements Engineers are scarce and late changes during the process are to be avoided but do occur.

Waldmann (2011) then introduces an interesting concept: Focusing on those requirements which are the most valuable from a customer perspective. In addition to this, requirements are categorized according to a few specific key factors such as the implementation of similar requirements in previous versions of the system. These key factors are then used to determine if the requirement would benefit from detailed Requirements Engineering or not. Another example for this would be domain specific knowledge which is available in form of experience in the development team (these requirements would not need to be analyzed in great detail as the developers would be very experienced in the domain and therefore would be very likely to make few mistakes). (Waldmann 2011) This process of sorting requirements according to their customer value closely resembles the prioritization process that is very common to Agile Software Development approaches (see section 2.7.1).

Another very interesting idea is the introduction of frequent releases. This is another central aspect of agile development techniques like Scrum (see section 2.7.1.2) and similar approaches. There are typically multiple full releases within one year with running development builds which are even more frequent. Waldmann (2011) proposes a separation in at least two different releases, six and ten months after the start of the Requirements Engineering process. They focus on basic, elementary requirements, which would be extremely costly if done too late, and then subsequent *updates* to clarify further, where additional clarification is needed. He also states that the attempt of providing one exhaustive requirements document which serves all stakeholders is merely a suboptimal solution and

therefore specialized documents for each and every different recipient should be preferred. However, it is also stated that it is vital to inform the customer of any new planning approach, as different stakeholders, especially those who are experienced with traditional development processes, may be alienated by the changes introduced. (Waldmann 2011)

#### 2.7.4 Traditional (linear) Requirements Engineering and Agile Software Development – It depends on the Project

As can be seen in the sections above, there is actually an area between the two extremes, which are exhaustive planning at the beginning of the system development and short cycles with small goals in agile approaches. As with all the tools involved in the software development process, it strongly depends on the project. When specific factors, which will be described in the following, can be determined, traditional (linear) methods are superior, but there are of course cases as well, where the high amount of flexibility in agile methods is key for project success.

Haberfellner et al. (2012) provide a list of five criteria for deciding if agile or traditional methods should be used (Haberfellner, de Weck et al. 2012):

- **Size of the project:** Small projects should prefer an agile approach, whereas big projects may benefit more from traditional methods.
- **Criticality:** When there are strong legal constraints, or otherwise very strongly limiting factors, traditional methods should be preferred.
- **Dynamics of the environment:** Highly dynamic environments are one of the reasons why agile methods were developed. They provide significant advantages in such cases because of their high amount of flexibility.
- **Personnel:** Highly qualified personnel is only necessary during the planning phase, the implementation can be done by lesser qualified personnel in traditional approaches. Agile methods require highly qualified personnel to be available during the entire development process.
- **Culture:** Agile methods work better in environments where people prefer freedom and responsibility with regard to their tasks. Traditional methods provide more security and are preferred in some cases.

Software is not developed using a strict set of rules. The survey mentioned in section 2.7.2 above clearly shows that in certain projects – even though the overall method utilized is an agile one – the need for a more systematic planning approach can be perceived (Lan, Ramesh 2008). In such cases often an incremental approach to Requirements Engineering can be used. This process does not require an exhaustive specification and documentation of all requirements before the system is in development, but allows for the emergence of requirements during the development.

The other perspective is possible as well, as can be seen in section 2.7.3 above. Companies and project teams that adopt traditional (linear) software development approaches may find the need for added flexibility. An example for this would be the introduction of elaborate prioritization techniques with regard to the requirements in order to be able to react better to deadlines and changing customer requirements.

In conclusion it can be said, that traditional software development methodologies, including traditional (linear) Requirements Engineering, and Agile Software Development are fundamentally different approaches, yet both have very clear advantages and disadvantages depending on the project that is being developed. However, in some projects, the dogmatic implementation of one single method may not produce as good results as a mixture of different approaches would. In these cases, such a mixture can be considered in order to achieve improved overall results.

### 3 REQUIREMENTS ENGINEERING: APPLICATION

This thesis is embedded into a research project (Vorraber, Vössner et al. 2014), aiming at improving medical processes through the use of Near-Eye Display Devices. Instead of implementing various ideas and prototypes, the very methodological approach of Requirements Engineering has been utilized to identify use case scenarios where the development of a human-centered information service could lead to improvements in terms of process performance, efficiency and / or safety. This process will be described in detail in the following sections.

It is worth mentioning that the process used in this section of the thesis is based upon the SOPHIST-Approach (Rupp, SOPHISTen 2009) to Requirements Engineering as outlined above (see section 2.5).

This thesis focuses mainly on medical processes such as interventions and surgeries. As can be seen in Figure 13, five general “Use Case Classes” with specific real-world occurrences (“Use Case Instances”) have been identified in a meeting with various stakeholders directly involved in the project (Vorraber, Neubacher 2014).

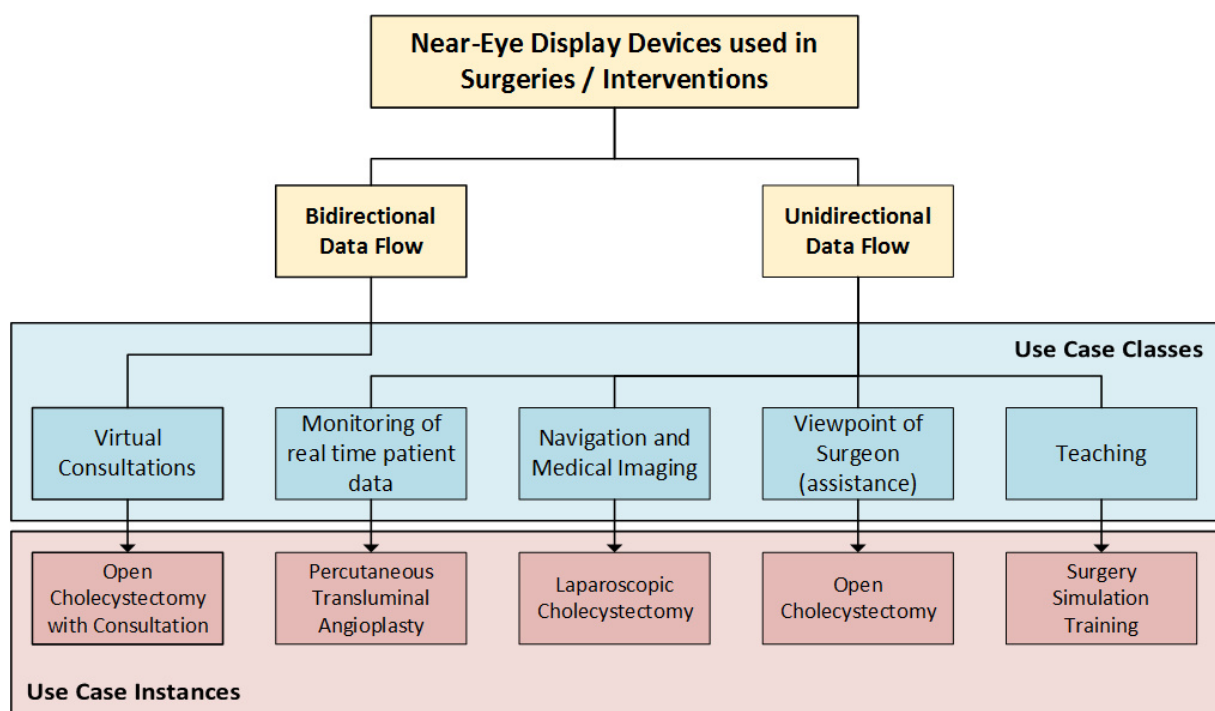


Figure 13: The five Use Case Classes along with their real-world counterparts this thesis focuses on.



These are first classified using a classification scheme that is explained in section 3.1 below, then briefly explained and then analyzed as they are currently implemented. The aforementioned analysis is then used to identify potential for improvements, which in turn serves as the basis for a detailed use case, which will be used for the design of an information system (see section 4 below). In order to improve readability, only one use case will be described at a time and explanation, analysis, identification of potential and use case formulation will be described as a whole for each use case.

### 3.1 CATEGORIZATION OF USE CASES

To be able to systematically distinguish between the different use cases, a categorization is necessary. Vorraber et al. (2014) have proposed the following scheme which will be used throughout this Master's Thesis for categorization (see Table 1):

Factor	Characteristics			
# users per case	1		N	
Data dynamics	Static		Dynamic	
Collaboration	Yes		No	
Information flow	Unidirectional		Bidirectional	
Mode of operation	Passive		Active	
Frequency of use	High – daily	Medium – weekly	Low - monthly	
Network coverage	Local	Regional	National	International

Table 1: Use Case classification scheme proposed by Vorraber et al. (2014).

The categories of the classification scheme outlined above will now be explained briefly, ordered by their impact with regard to the classification of the use cases outlined in Figure 13.

1. **Information flow:** Especially the distinction between unidirectional and bidirectional data flow is a key criterion for the classification of the use cases, as can be seen in Figure 13.
2. **Data dynamics:** Another extremely important factor in the classification scheme above is the distinction between static and dynamic data. The use cases described in this thesis focus only on dynamic data (e.g. data that changes frequently over time). It would, however, be interesting to explore static data as a source for Near-Eye Display Devices in the future as well (see 7.2 below).

3. **Frequency of use:** The frequency of use should be fairly self-explanatory. The significance of this criterion is its expression of the importance of the classified use case (processes and use cases that occur daily are clearly more important than the ones that are only performed on a monthly basis).
4. **Network coverage:** The network coverage provides information with regard to the technical difficulties that have to be overcome. Consulting with a surgeon who performs a surgery in another country poses a completely new set of problems (e.g. latency, bandwidth, etc.).
5. **# users per case:** The number of users should be fairly self-explanatory. It denotes the number of users that directly interact with the information service at any given point in time during the use case.
6. **Collaboration:** Collaboration provides information with regard to the type of use case and its (directly involved) users: Are two (or more) users directly collaborating using the system?
7. **Mode of operation:** the mode of operation denotes if the wearer is actively interacting with the system (e.g. through touch gestures), or not.

Based on (Vorraber, Vössner et al. 2014)

## 3.2 PROJECT GOALS

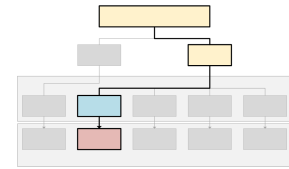
As described in section 2.5.1.3 above, it is part of the AS-IS Analysis to devise a general list of goals that can be classified as requirements of specification level 0 (see section 2.5.2 above). The fundamental goals for this project are based on early discussions with central stakeholders.

- 1) Improvement of well-established processes in medical environments by introducing Near-Eye Display Devices.
- 2) Provision of a way to conduct consultations without the need for the consulting party to be locally present.
- 3) Provision of real-time patient data (e.g. vital signs) through a Near-Eye Display Device.
- 4) Provision of a possibility to allow other parties to include the wearer's field of vision into their routine (e.g. for improve assistance during surgeries).
- 5) Provision of the field of vision of the wearer for evaluation and teaching purposes.

### 3.3 MONITORING OF REAL TIME PATIENT DATA

One of the most interesting use cases is the monitoring of real time patient data in the operating room during surgeries. This is especially

interesting with regard to minimally invasive surgeries, where an anesthesiologist usually is not present and therefore the task of monitoring has to be done by the surgeon himself.



#### 3.3.1 Use Case Instance Explanation – Percutaneous Transluminal Angioplasty

As a specific use case instance for the **Monitoring of Real Time Patient Data**, the **Percutaneous Transluminal Angioplasty** has been chosen. The process behind this surgical intervention (see Figure 14) will now be explained briefly: A Percutaneous Transluminal Angioplasty (in short “PTA”) is a minimally invasive procedure which seeks to widen occluded (or narrow) arteries by inserting a balloon catheter which is then inflated. The procedure is typically conducted in a minimally invasive way by inserting the catheter through a single small incision into a larger artery (e.g. in the area of the thighs) and then navigating to the necessary spot using medical imaging techniques (typically an angiogram).



Figure 14: Percutaneous Transluminal Angioplasty performed with Google Glass. The surgeon receives real time patient data onto his Near-Eye Display Device (Vorraber, Vössner et al. 2014).

### 3.3.2 Classification and Explanation

In this section, the Percutaneous Transluminal Angioplasty will be classified (see Table 2) utilizing the classification scheme outlined in section 3.1 above.

Factor	Characteristics			
# users per case	1		N	
Data dynamics	Static		Dynamic	
Collaboration	Yes		No	
Information flow	Unidirectional		Bidirectional	
Mode of operation	Passive		Active	
Frequency of use	High – daily	Medium – weekly	Low - monthly	
Network coverage	Local	Regional	National	International

Table 2: Classification of the Percutaneous Transluminal Angioplasty use case using the classification scheme proposed by Vorraber et al. (2014).

### 3.3.3 AS-IS-Analysis and Identification of Potential for Improvement

The Percutaneous Transluminal Angioplasty is a surgical procedure that does not require full narcosis of the patient and therefore does not have the need for an anesthesiologist to be present at any given point in time. This, however, requires the surgeon to constantly monitor the patients' vital signs himself.

The process has been modeled using BPMN (see Figure 15) building on information gathered through interviews with the involved stakeholders. Activities which could potentially benefit from the introduction of Near-Eye Display Devices are highlighted.

The following part of the full BPMN diagram shows the activities which could benefit from improvement by the introduction of Near-Eye Display Devices:

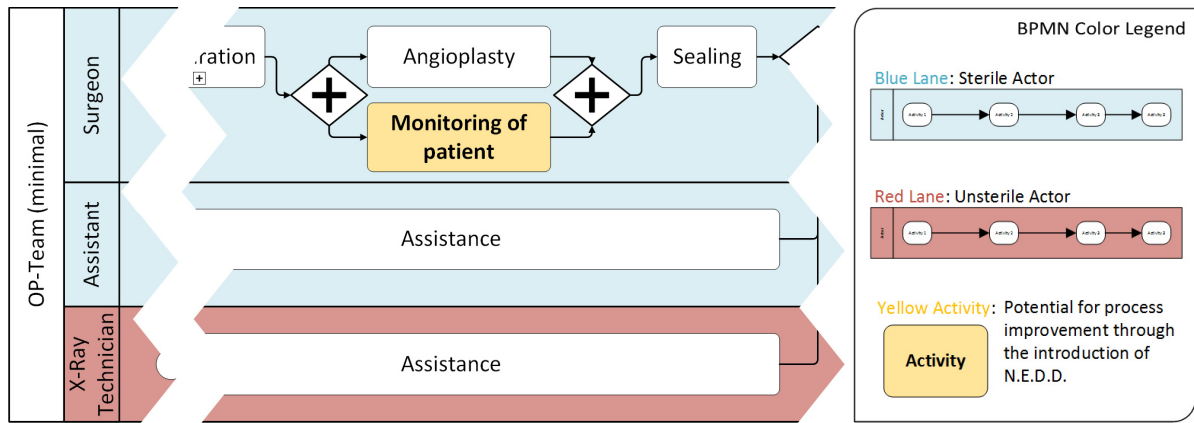


Figure 15: Percutaneous Transluminal Angioplasty AS-IS-Process Analysis (BPMN excerpt)

Figure 16: BPMN Color Legend.

- **Monitoring of patient:**

As has already been mentioned above, in this particular kind of medical intervention, the surgeon needs to monitor the patients’ vital signs on his own. However, there are certain situations, where the surgeon loses the line-of-sight to the monitoring screen (see Figure 14). Providing the patients’ data in a form where it is constantly visible to the surgeon – no matter what he is doing at the moment – could improve efficiency as well as safety.

The full BPMN diagram with all activities, actors and the overall process can be found in section 11.1.2 below.

### 3.3.4 Description of Resulting Use Case

In the following Table 3, the use case as it would appear with Google Glass as a new information channel, is explained. This systematic form of specification (proposed by Rupp et al. (2009)) has been chosen because it provides a very structured way to describe all necessary information related to the process in a compact form.

<b>Name</b>	Percutaneous Transluminal Angioplasty (Monitoring of Real Time Patient Data)
<b>Short Description</b>	This use case describes how Near-Eye Display Devices can be used to display real time patient data (such as heart rate, blood pressure or oxygen levels) at a fixed location in the field of vision of the surgeon without the need of specifically having to locate a monitoring screen positioned somewhere in the room.

<b>Actors</b>	Patient, Surgeon
<b>Prerequisites</b>	Local Network, Near-Eye Display Device (such as Google Glass), server (PC running inside of the network with started server application → Thin Client Architecture, see section 4.2.3 below), medical data via standardized interfaces (such as VGA, DICOM (DICOM Standards Committee 2011) or similar)
<b>Trigger</b>	Start of the operation (where a constant observation of specific patient data adds value, efficiency and / or safety to the process).
<b>Typical Process</b>	<ul style="list-style-type: none"> <li>• Operating room is prepared. <ul style="list-style-type: none"> <li>○ Network is prepared.</li> <li>○ Near-Eye Display Device is started and connected to the network.</li> <li>○ Server-PC is started, connected to the network and server-application is launched.</li> <li>○ Data-Feed (e.g. a VGA-signal) is connected to the server.</li> </ul> </li> <li>• Surgeon wears Near-Eye Display Device and is able to see live data inside the display during surgery.</li> <li>• After the surgery, the server application and the Near-Eye Display Device are disconnected and may be switched off until further application.</li> </ul>
<b>Remarks</b>	The use of a video signal like VGA yields the added benefit that the received data looks exactly as it does on the monitoring screen and therefore minimizes familiarization time for the surgeon.

Table 3: Specific Percutaneous Transluminal Angioplasty use case, using the scheme proposed by Rupp et al. (2009).

Especially interventions and surgeries that do not require the presence of an anaesthesiologist can benefit from the provision of the patient's vital signs in the field of vision of the surgeon. The use case outlined in Table 3 describes a surgical intervention where the interventionist wears the Near-Eye Display Device and receives the data of the medical appliance that monitors the patient's vital signs. Therefore, no matter where the interventionist looks or what he focuses on, the data is directly in front of him (see Figure 17, right). This allows him to acquire the information he needs by simply refocusing instead of physically locating the monitoring device in the room and actively having to focus on it.

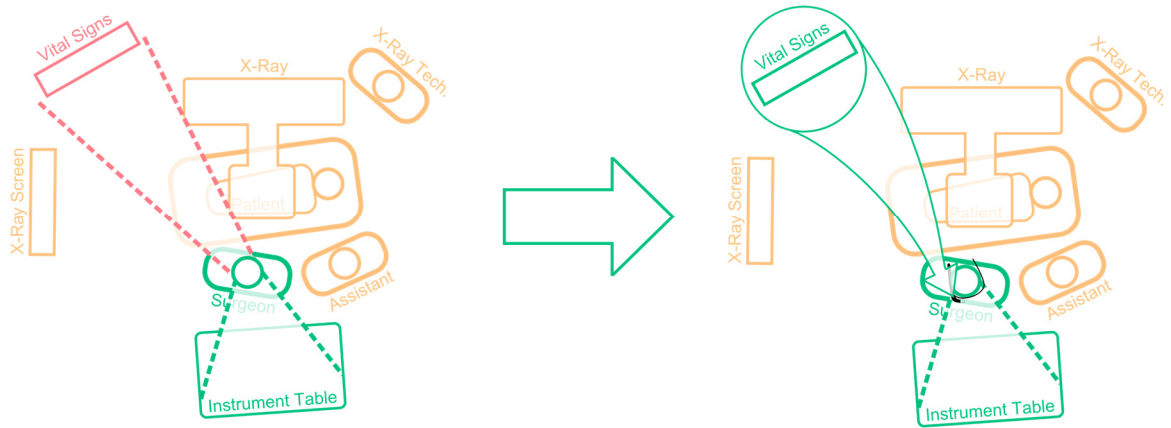
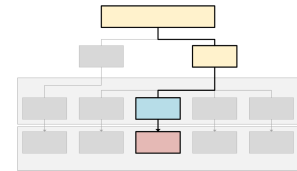


Figure 17: Through the introduction of Near-Eye Display Devices, the interventionist cannot lose line of sight to the patient's vital signs anymore (right).

### 3.4 NAVIGATION AND MEDICAL IMAGING

There are many different surgeries that rely on medical imaging techniques (such as roentgen imaging, angiograms, MRIs, etc.). A special sub-category of these use cases are surgeries involving surgical-navigational-instruments (which allow the surgeon to monitor the position of his instruments on a predefined medical image, such as a roentgen image, in real time).



#### 3.4.1 Use Case Instance Explanation – Laparoscopic Cholecystectomy

As a specific use case instance for **Navigation and Medical Imaging**, the **Laparoscopic Cholecystectomy** has been chosen. The surgical process behind this surgery (see Figure 18)

will now be explained briefly: A Laparoscopic Cholecystectomy is a minimally invasive surgery, which aims to remove the gallbladder. When the operation starts, the stomach region is first inflated using an inert gas (typically CO<sub>2</sub>), then pierced by a number of so-called trocars, which provide a way for other medical instruments to be inserted into the stomach. One of these instruments is an endoscope (which provides a visual image of the inside of the stomach region). Using the endoscope and various medical instruments, the gallbladder is separated from its surroundings (using an angiogram to determine which connections can safely be severed) and then extracted through one of the trocars.



Figure 18: Surgeon and assisting personnel focusing on the endoscopic image during a Laparoscopic Cholecystectomy (Sachs 2014a).

#### 3.4.2 Classification and Explanation

In this section, the Laparoscopic Cholecystectomy will be classified (see Table 4) utilizing the classification scheme outlined in section 3.1 above.

Factor	Characteristics	
# users per case	1	N
Data dynamics	Static	Dynamic
Collaboration	Yes	No



Information flow	Unidirectional		Bidirectional	
Mode of operation	Passive		Active	
Frequency of use	High – daily	Medium – weekly	Low - monthly	
Network coverage	Local	Regional	National	International

Table 4: Classification of the Laparoscopic Cholecystectomy use case using the classification scheme proposed by Vorraber et al. (2014).

### 3.4.3 AS-IS-Analysis and Identification of Potential for Improvement

Due to the way the operating room at the Hospital of Elisabethinen is built (especially with regard to oxygen outlets in the walls), the patient has to be positioned in a very specific way for this kind of surgery. This leads to a sub-optimal positioning of medical equipment, where the screen showing the angiogram is behind the surgeon and the screen showing the endoscopic image is in front of him (see Figure 19, marked in red and Figure 20, screens 1 and 2. Screen 3 is used for surgery documentation in the Hospital Information System - HIS).

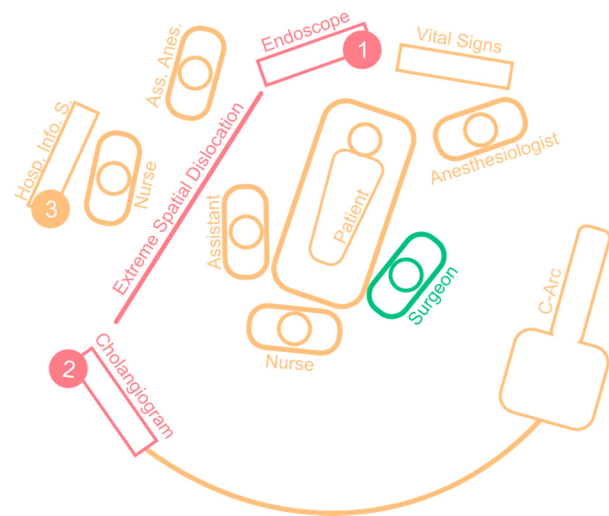


Figure 19: Schematic overview of the operating room during a Laparoscopic Cholecystectomy.

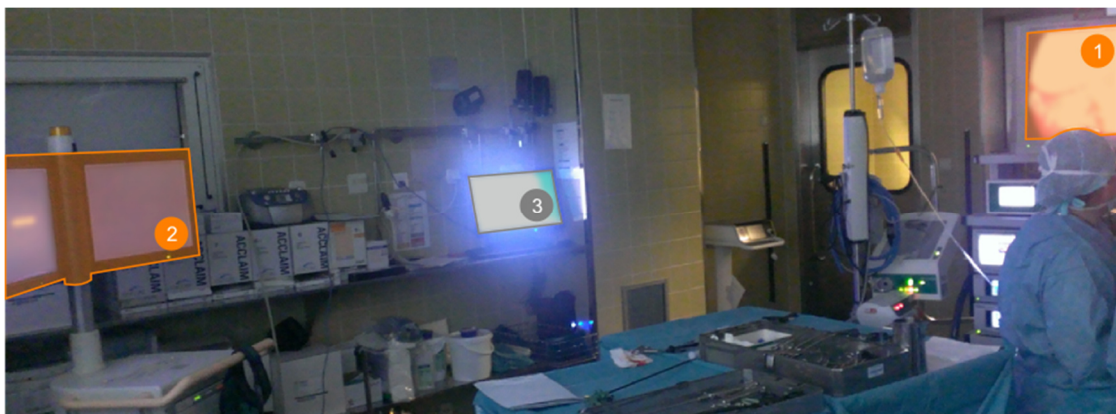


Figure 20: The screen positioning in the operating room during a Laparoscopic Cholecystectomy at the Hospital of Elisabethinen (Sachs 2014b).

Again, the process has been modeled using BPMN (see Figure 21) building on information gathered through interviews with involved stakeholders. Activities which could potentially

benefit from the introduction of Near-Eye Display Devices are highlighted. For a full legend, please refer to 11.1.1 below.

The following activities might benefit from the introduction of Near-Eye Display Devices:

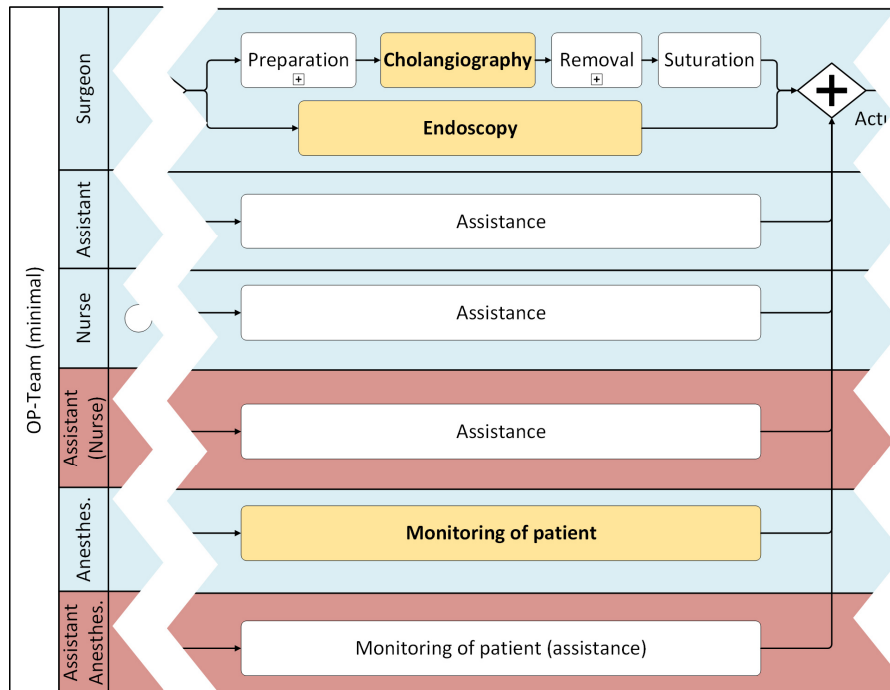


Figure 21: Laparoscopic Cholecystectomy AS-IS Process Analysis (BPMN excerpt)

- **Endoscopy or Cholangiography:**

Because of the placement of screens in the operating room, one is constantly outside the field of vision of the surgeon (see Figure 19 and Figure 20). Displaying the contents of one of these screens inside the Near-Eye Display Device could lead to an improvement with regard to efficiency.

- **Monitoring of patient:**

This particular activity might benefit not as much as the other, because the anesthesiologist monitors the vital signs, leaving the surgeon occupied with the actual surgery.

The full BPMN diagram with all activities, actors and the overall process can be found in section 11.1.3 below.

### 3.4.4 Description of Resulting Use Case

In Table 5, the use case as it would appear with a Near-Eye Display Device as a new information channel, is explained.

<b>Name</b>	Laparoscopic Cholecystectomy (Navigation and Medical Imaging)
<b>Short Description</b>	This use case describes how Near-Eye Display Devices may be utilized to facilitate alternating between different medical images during surgeries where multiple screens are used to visualize different aspects of the operation.
<b>Actors</b>	Patient, Surgeon
<b>Prerequisites</b>	Local Network, Near-Eye Display Device (such as Google Glass), server (PC running inside of the network with started server application), medical imaging data via standardized interfaces (such as VGA, DICOM (DICOM Standards Committee 2011) or similar)
<b>Trigger</b>	Start of the operation (where multiple medical monitoring screens are utilized).
<b>Typical Process</b>	<ul style="list-style-type: none"> <li>• Operating room is prepared. <ul style="list-style-type: none"> <li>○ Network is prepared.</li> <li>○ Near-Eye Display Device is started and connected to the network.</li> <li>○ Server-PC is started, connected to the network and server-application is launched.</li> <li>○ Medical Appliances (e.g. navigational or medical imaging devices) are connected to the server via standardized interfaces (e.g. VGA).</li> </ul> </li> <li>• Surgeon wears Near-Eye Display Device and therefore has constant access to medical imaging data without the need of physically locating the monitoring screen in the room.</li> <li>• After the operation, the server application and the Near-Eye Display Device are disconnected and may be switched off until further application.</li> </ul>
<b>Remarks</b>	The aforementioned medical appliances are often located at various different locations in the operating room, making it difficult for the surgeon to switch between them in a quick fashion. Providing one screen directly in front of the surgeon could improve efficiency and / or safety.

*Table 5: Specific Laparoscopic Cholecystectomy use case, using the scheme proposed by Rupp et al. (2009).*

With the problems outlined in section 3.4.3 above, the provision of the angiogram in the Near-Eye Display Device has been identified as a potential opportunity for process improvement. Instead of having to alternate between the two central screens the surgeon has to rely on (see Figure 22, left), he can focus specifically on the screen showing the transmission of the

endoscope (see Figure 22, right), while having the angiogram visible in the corner of his eye through the display of his Near-Eye Display Device. This allows the surgeon to avoid having to physically locate the angiogram's screen and to alternate between it and the image of the endoscope during the critical part of the surgery.

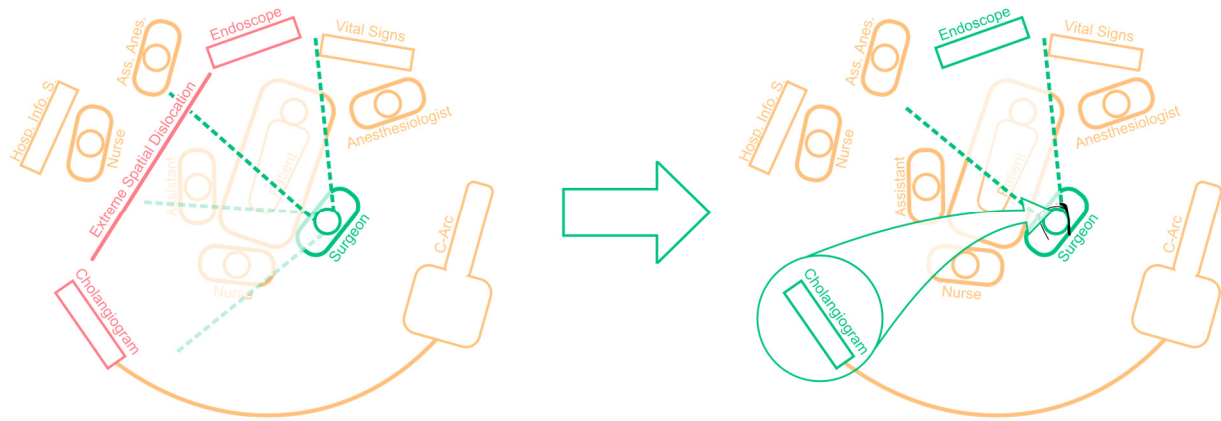


Figure 22: The introduction of a Near-Eye Display Device to the process of a Laparoscopic Cholecystectomy reduces the number of necessary shifts with regard to the surgeon's point of view.

### 3.5 VIEWPOINT OF SURGEON FOR IMPROVED ASSISTANCE

For aesthetic reasons, the incision used in most surgeries is kept as small as possible. This is especially a problem in the area of the head (ear, nose and throat), where the part of the body that is operated on (and therefore is cut open) is typically extremely narrow. In an interview with stakeholders, it became apparent, that in these cases, assisting in the operating room is extremely difficult and assisting personnel often has to rely strongly on experience and non-verbal communication. Having the point of view of the surgeon displayed on a screen during the operation could decrease the amount of communication needed and therefore increase efficiency and possibly even safety.

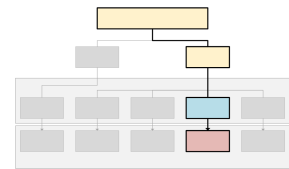
As a specific use case instance for the use case class *Viewpoint of Surgeon for Improved Assistance*, the **Open Cholecystectomy** has been chosen.

#### 3.5.1 Use Case Instance Explanation – Open Cholecystectomy

The process behind the **Open Cholecystectomy** will now be explained briefly: The Laparoscopic Cholecystectomy has already been outlined above (see section 3.4) but this kind of minimally invasive procedure is not always a possibility, in which case a traditional, “Open” Cholecystectomy has to be performed. In this case a significantly bigger cut is made through which the surgeon performs the surgery. However, the incision is kept as small as possible (cf. Figure 23) which provides only a very narrow window, offering just the surgeon a good point of view of the field of operation. Then, similar to the Laparoscopic Cholecystectomy (see section 3.4 above), the connection to the gallbladder is severed and it is extracted.



Figure 23: Open surgeries, especially with minimal incisions are often too narrow to be clearly visible (especially) to the assisting personnel. (Weldi 2014)



As can be seen in the sketch provided in Figure 24, the assistant (marked with “2”) who needs to hold the wound open for the surgeon (marked with “1”) to be able to operate properly, has to stand on the opposite side of the table. This provides him only with a very limited view of the surgery itself. Therefore he typically has to rely strongly on experience and non-verbal communication for the surgery to forgo smoothly.

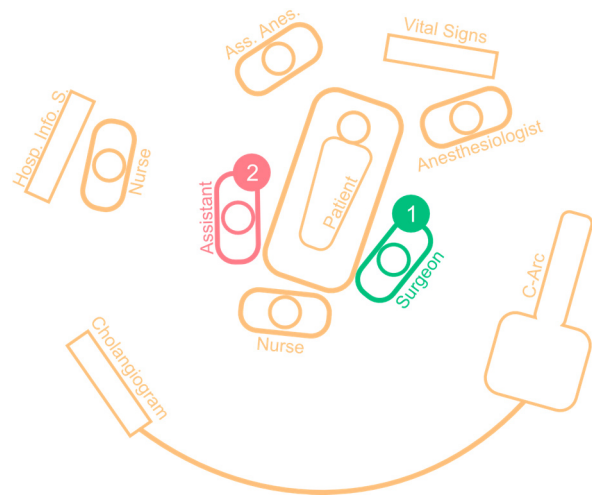


Figure 24: Sketch of the operating room during an Open Cholecystectomy.

### 3.5.2 Classification and Explanation

In this section, the Open Cholecystectomy will be classified (see Table 6) utilizing the classification scheme outlined in section 3.1 above.

Factor	Characteristics			
# users per case	1		N	
Data dynamics	Static		Dynamic	
Collaboration	Yes		No	
Information flow	Unidirectional		Bidirectional	
Mode of operation	Passive		Active	
Frequency of use	High – daily	Medium – weekly	Low - monthly	
Network coverage	Local	Regional	National	International

Table 6: Classification of the Open Cholecystectomy use case using the classification scheme proposed by Vorraber et al. (2014).

### 3.5.3 AS-IS-Analysis and Identification of Potential for Improvement

Similar to the two use cases that have already been analyzed in the sections above, this process too has been modeled using BPMN. Again, activities which could potentially benefit from the introduction of Near-Eye Display Devices have been highlighted. For a full legend, please refer to section 11.1.1 below.

The following activities have been identified as the ones which could potentially benefit from the introduction of Near-Eye Display Devices:

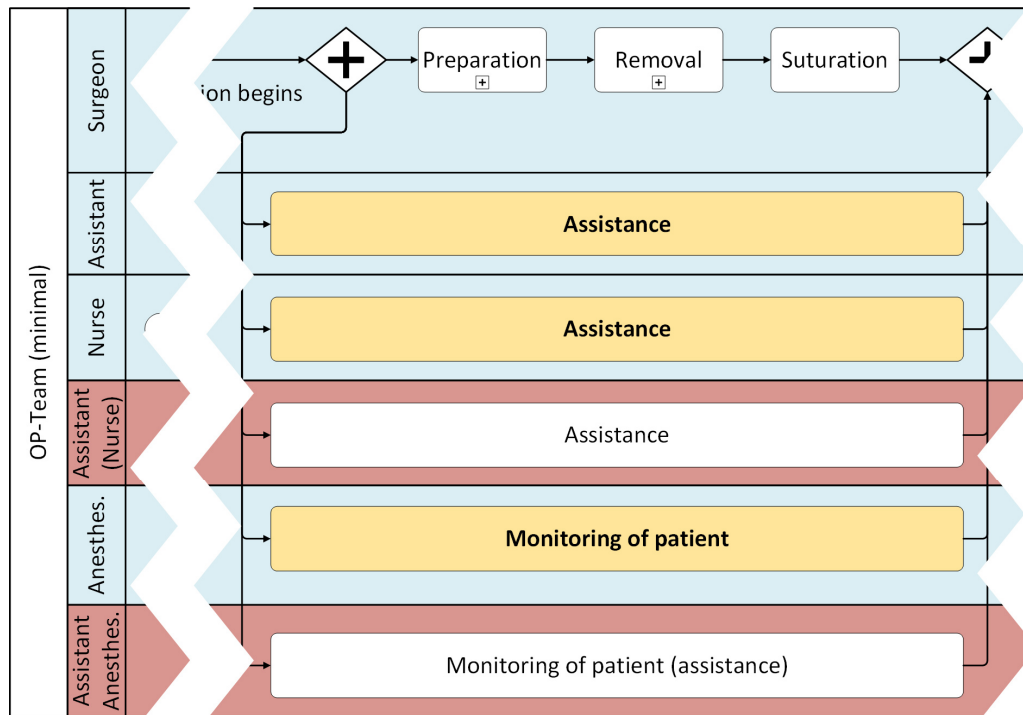


Figure 25: Open Cholecystectomy AS-IS-Process Analysis (BPMN excerpt)

- **Assistance:**

Because of the small size of the incision, even the surgeon has to position himself in a cramped way to be able to see everything important. The assistant has almost no chance of an adequate viewpoint and therefore has to rely strongly on experience and intuition. The viewpoint of the surgeon, displayed on a screen inside the operating room could be of great help in this particular scenario, as stakeholders have mentioned in an interview.

- **Monitoring of patient:**

Again, this particular activity might benefit not as much as the other, because the anesthesiologist monitors the vital signs, leaving the surgeon occupied with the actual surgery.

### 3.5.4 Description of Resulting Use Case

Similar to the previous sections, the use case as it would appear with a Near-Eye Display Device as a new information channel, is explained below (see Table 7).

<b>Name</b>	Open Cholecystectomy (Viewpoint of Surgeon for Improved Assistance)
<b>Short Description</b>	This use case describes how a head-mounted camera, which is often integrated in Near-Eye Display Devices such as Google Glass, may be used during surgery to facilitate tasks which are conducted by personnel other than the surgeon (e.g. the nurse, who may benefit from a more direct perspective of the operating area during processes such as suction).
<b>Actors</b>	Patient, Surgeon, a third-party who benefits from the surgeons viewpoint (e.g. Nurse)
<b>Prerequisites</b>	Local Network, head-mounted camera (such as the one which is integrated in Google Glass), server (PC running inside of the network with started server application), Screen to display the viewpoint of the surgeon (either via direct output from the server, or by utilizing another thin-client in communication with the server)
<b>Trigger</b>	Start of the operation (where the viewpoint of the surgeon is of value for the surrounding personnel e.g. in surgeries with very narrow operating areas).
<b>Typical Process</b>	<ul style="list-style-type: none"> <li>• Operating room is prepared. <ul style="list-style-type: none"> <li>○ Network is prepared.</li> <li>○ Head-mounted camera (such as the camera integrated in Google Glass) is started and connected to the network.</li> <li>○ Server-PC is started, connected to the network and server-application is launched.</li> <li>○ Screen is started and connected to the server (either directly, or over the network).</li> </ul> </li> <li>• Surgeon wears head-mounted camera and therefore provides his viewpoint to the server.</li> <li>• Personnel other than the surgeon may utilize this unique viewpoint during critical actions.</li> <li>• After the operation, the server application and the head-mounted camera are disconnected and may be switched off until further application.</li> </ul>
<b>Remarks</b>	None.

*Table 7: Specific Open Cholecystectomy use case, using the scheme proposed by Rupp et al. (2009).*

With the introduction of Near-Eye Display Devices into the process of an Open Cholecystectomy, it is possible to make the viewpoint of the surgeon available to the assisting



personnel. Especially with regard to surgeries in narrow areas, this could improve the process by providing a clear view of the operating area to all involved parties. The Near-Eye Display Device is used as a camera and as a transmitter, which will then display the field of vision of the surgeon to a screen for the assisting personnel (see Figure 26, right). This allows them to utilize the unique point of view of the surgeon for the improvement of their assistance during the surgery, which is normally hampered by their limited perspective on the operating field. This could increase not only efficiency and effectiveness, by reducing the amount of verbal and non-verbal communication needed during the surgery, but could also increase safety for the patient.

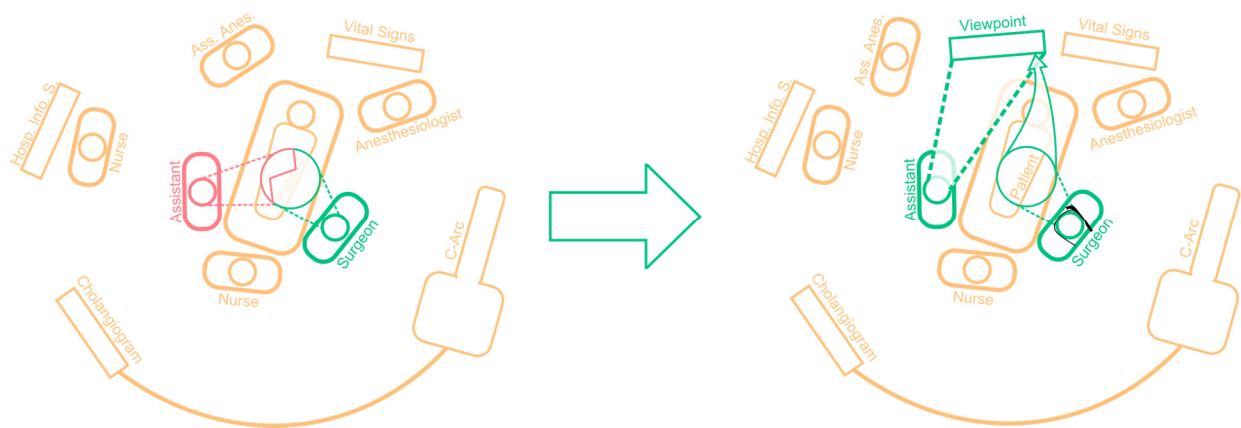
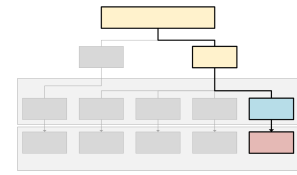


Figure 26: The assisting personnel can improve their assistance by utilizing the unique point of view of the surgeon. This allows them to avoid having to rely solely on their limited perspective of the operating field.

### 3.6 RECORDING OF MEDICAL PROCESSES FOR POST-EVALUATION AND TEACHING PURPOSES

Training is vital for the processes in the operating room to run smoothly, especially when it comes to medical emergencies where timing and collaboration are essential for the overall success of the surgery.



There are already systems which aim to record simulations in an operating room and provide them for post-evaluation such as SIMStation (Studiokonzzept Medientechnik GmbH 2012) and similar solutions. These systems, however, only provide one or multiple different camera-perspectives of the operating room and the medical appliances in them. They typically do not provide a mobile perspective (such as the point of view of the surgeon).

As a specific use case instance for the *Recording of Medical Processes for Post-Evaluation and Teaching Purposes*, the simulation of a surgery has been chosen.

#### 3.6.1 Use Case Instance Explanation – Surgery Simulation Training

The point of view of the surgeon may prove to be a helpful tool for teaching and post-evaluation. According to medical students, one big problem when it comes to the observation of surgeries is that there is typically at least one obstacle which is

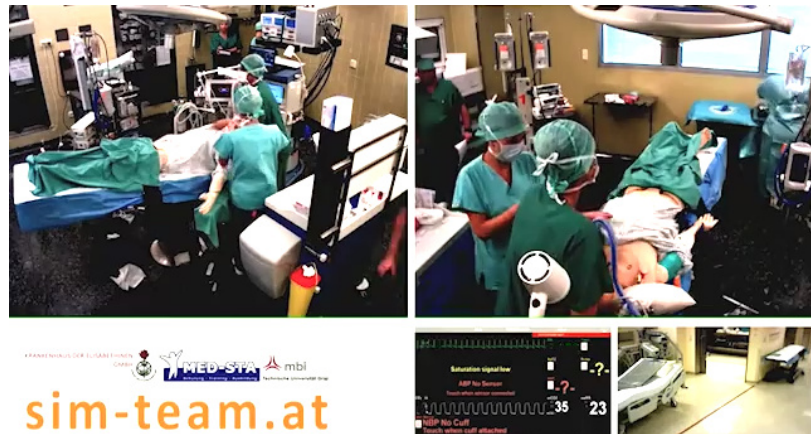


Figure 27: Current recording setups often include only stationary cameras (Krankenhaus der Elisabethinen GmbH 2013).

blocking the view: The surgeon himself. Therefore, the ability to provide recordings of operations (with, or without the student in the operating room) could prove helpful with regard to teaching purposes. In addition to this, typical recording systems only provide fixed perspectives of the operating room, rather than moving point-of-view ones (see Figure 27). These moving perspectives could also provide helpful insights for students, as they can observe what an experienced surgeon focuses on during the process. This becomes even more interesting, when the Near-Eye Display Device has the capability of tracking the eye

movement of the wearer, which Google Glass is not capable of at this point in time (see section 7.2 below).

Similar recordings could be also used for post-surgery-evaluation with the operating team after the surgery.

### 3.6.2 Classification and Explanation

The Surgery Simulation Training Use Case will be classified in Table 8 utilizing the classification scheme outlined in section 3.1 above.

Factor	Characteristics			
# users per case	1		N	
Data dynamics	Static		Dynamic	
Collaboration	Yes		No	
Information flow	Unidirectional		Bidirectional	
Mode of operation	Passive		Active	
Frequency of use	High – daily	Medium – weekly	Low - monthly	
Network coverage	Local	Regional	National	International

Table 8: Classification of the Surgery Simulation Training use case using the classification scheme proposed by Vorraber et al. (2014).

### 3.6.3 AS-IS-Analysis and Identification of Potential for Improvement

The traditional process of recording surgeries for training and simulation processes has been analyzed and potentially interesting activities have been highlighted (see Figure 28). An excerpt is provided below (the full BPMN diagram with all activities, actors and the overall process can be found in section 11.1.5 below), for a legend of the colors used in the BPMN diagram, please refer to section 11.1.1 below:

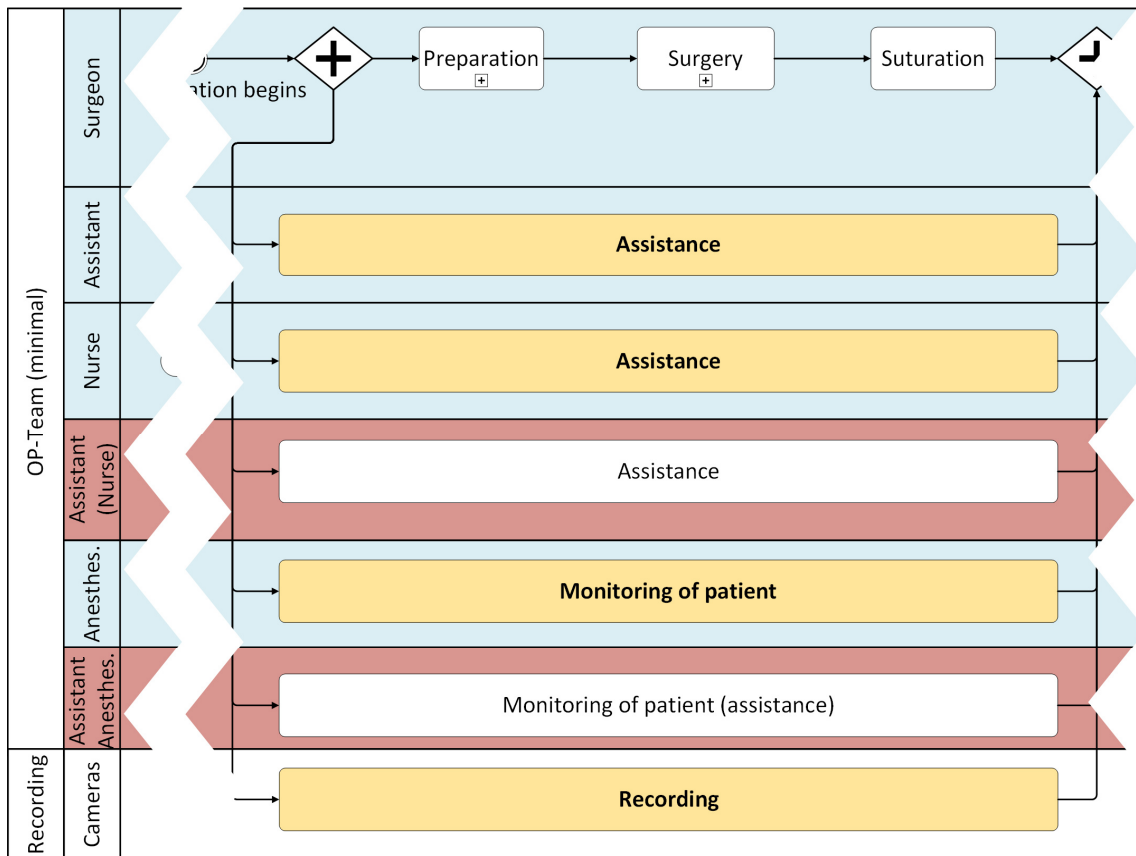


Figure 28: Surgery Simulation Training AS-IS-Process Analysis (BPMN excerpt).

- **Recording:**

The recording system that is currently in use at the Krankenhaus der Elisabethinen Graz GmbH provides multiple stationary HD cameras. The introduction of Near-Eye Display Devices would allow for a completely new perspective: The Staff. For example, it could be vital for post-evaluation (or even teaching purposes) to be able to follow the movements of the surgeon with regard to the center of his attention. Similar to this, it could be interesting to observe the head-movements of an anesthesiologist who faces a medical emergency during an operation (e.g. “Cannot Intubate, Cannot Ventilate”). These new recordings could then be used for improved post-evaluation and process improvement or the teaching of medical students.

- **Monitoring of patient:**

Similar to the use cases outlined in the previous sections, it could be interesting to provide the anesthesiologist with a portable way to monitor the dummy’s vital signs.

### 3.6.4 Description of Resulting Use Case

Again, similar to the previous sections, the use case as it would appear with a Near-Eye Display Device is explained below (see Table 9).

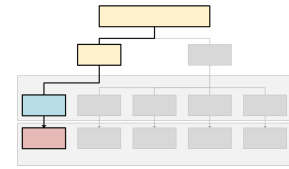
<b>Name</b>	Surgery Simulation Training (Recording of Medical Processes for Post-Evaluation and Teaching Purposes)
<b>Short Description</b>	This use case describes how a head-mounted camera (such as the one integrated in Google Glass) may be used during a simulated surgery to record the surgical process with the aim of analysis and post-evaluation.
<b>Actors</b>	Dummy, Surgeon, OP-Staff
<b>Prerequisites</b>	Local Network, head-mounted camera (such as found in Google Glass), server (PC running inside of the network with started server application)
<b>Trigger</b>	The start of the surgery simulation.
<b>Typical Process</b>	<ul style="list-style-type: none"> <li>• Operating room is prepared. <ul style="list-style-type: none"> <li>○ Network is prepared.</li> <li>○ Head-mounted camera is started and connected to the network.</li> <li>○ Server-PC is started, connected to the network and server-application is launched.</li> <li>○ Recording is started.</li> </ul> </li> <li>• Surgeon wears head-mounted camera and therefore provides his viewpoint to the server.</li> <li>• After the simulation, the server application and the head-mounted camera are disconnected and may be switched off until further application.</li> <li>• The recorded data may then be used at any given point in time to analyze the conducted surgery or provide real-world-data for post-evaluation meetings.</li> </ul>
<b>Remarks</b>	None.

Table 9: Specific Surgery Simulation Training use case, using the scheme proposed by Rupp et al. (2009).

The Near-Eye Display Device is worn by the surgeon (or any other party of the OP-Staff) and the whole process is recorded in either visual, or audiovisual form. The resulting recordings can then be used either for the education of medical students or for post-evaluation after simulation trainings.

### 3.7 VIRTUAL CONSULTATIONS

Although rather rare, consultations do happen during surgeries, especially with young and still inexperienced surgeons that do not



want to judge specific situations completely by themselves. The problem with consultations in their current form is the time that is needed for the consulting surgeon to prepare himself before entering the operating room (especially the process of cleaning is extremely time consuming). However, there are situations where just a second opinion and a quick look could suffice for the operating surgeon to continue the surgery.

As a specific use case instance for Virtual Consultations, the **Open Cholecystectomy with Consultation** has been chosen.

#### 3.7.1 Use Case Instance Explanation – Open Cholecystectomy with Consultation

The process behind the **Open Cholecystectomy with Consultation** will now be explained briefly: Especially young and inexperienced surgeons might find themselves confronted with a situation they are not completely comfortable judging on their own. In this case, a bidirectional audiovisual link to a more experienced surgeon might provide both a relief for the acting surgeon as well as a safer surgery for the patient.

#### 3.7.2 Classification and Explanation

The Open Cholecystectomy with Consultation will be classified in Table 10 utilizing the classification scheme outlined in section 3.1 above.

Factor	Characteristics			
# users per case	1		N	
Data dynamics	Static		Dynamic	
Collaboration	Yes		No	
Information flow	Unidirectional		Bidirectional	
Mode of operation	Passive		Active	
Frequency of use	High – daily	Medium – weekly	Low - monthly	
Network coverage	Local	Regional	National	International

Table 10: Classification of the Open Cholecystectomy with Consultation use case using the classification scheme proposed by Vorraber et al. (2014).

### 3.7.3 AS-IS-Analysis and Identification of Potential for Improvement

Again, the traditional process with regard to a consultation during the surgery has been analyzed and potentially interesting activities have been highlighted (see Figure 29). An excerpt is provided below (the full BPMN diagram with all activities, actors and the overall process can be found in section 11.1.6 below), for a legend of the colors used in the BPMN diagram, please refer to section 11.1.1 below:

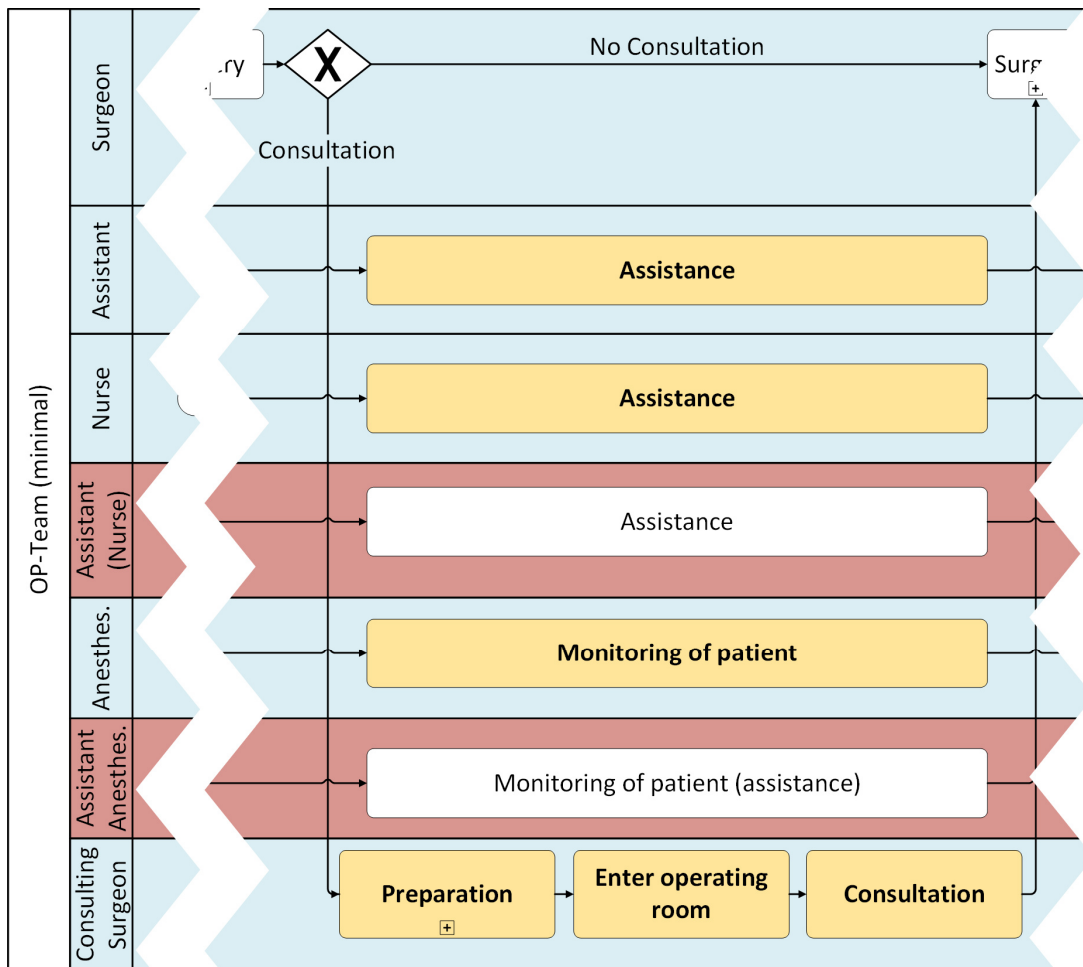


Figure 29: Open Cholecystectomy with Consultation AS-IS Process Analysis (BPMN excerpt)

- **Assistance, Monitoring of patient**

The base use case for this scenario was the Open Cholecystectomy. Therefore, Assistance and Monitoring are identical to the processes outlined in section 3.5.3 above.

- **Preparation, Enter operating room, Consultation**

Especially the process of preparation takes a significant amount of time (first and foremost the process of cleaning and sterilization before entering the operating room). The introduction of a two-way data feed to provide the viewpoint, high-resolution images and maybe even an audio line to a consulting surgeon might reduce the time needed for consultation significantly and also, by minimizing the time spent by the consulting surgeon, would increase efficiency (as the surgeon can use this time for other activities).

The big advantage of a head mounted device in this case would be the possibility for the consulting surgeon to have the exact viewpoint of the surgeon instead of one fixed perspective, which he would have with a normal camera and being able to alter this viewpoint by giving instructions to the surgeon performing the surgery via an audio line (e.g. “look over there”). This in turn provides the consulting surgeon with a view of the operating room which yields the added benefit that he does not have to rely on a description of the situation (e.g. the oxygen levels of a patient), but can look at them himself and is therefore able to use his natural pattern recognition abilities, which are part of his experience as a surgeon. Especially the latter part, the ability to judge situations by recognizing patterns on monitoring devices or on the patient have been mentioned by stakeholders in several interviews.

### 3.7.4 Description of Resulting Use Case

The use case that has been identified using the basis provided in section 3.7.3 above will now be described in Table 11.

<b>Name</b>	Virtual consultations
<b>Short Description</b>	This use case describes how a Near-Eye Display Device along with a built-in camera, microphone and speaker may be used to perform real time consultations without the need for the consulting person to be locally present (and therefore being able to perform such consultations more quickly and efficiently).
<b>Actors</b>	Patient, Surgeon, Consulting Surgeon
<b>Prerequisites</b>	Network, head-mounted device (such as Google Glass), server (PC running inside of the network with started server application), Thin-Client for the consulting surgeon (providing video and audio in- and output)



<b>Trigger</b>	The surgeon sees himself confronted with a situation which he does not want to risk judging solely by himself. Therefore a consultation is conducted with a fellow surgeon (ideally with special experience on the specific matter).
<b>Typical Process</b>	<ul style="list-style-type: none"> <li>• Operating room is prepared. <ul style="list-style-type: none"> <li>○ Network is prepared.</li> <li>○ Near-Eye Display Device is started and connected to the network.</li> <li>○ Server-PC is started, connected to the network and server-application is launched.</li> <li>○ Screen (or monitoring thin client device) is started and connected to the server (either directly, or over the network).</li> </ul> </li> <li>• Surgeon wears Near-Eye Display Device and therefore provides his viewpoint along with additional data (sound / speech) to the server via the built-in sensors.</li> <li>• The Surgeon may trigger a consultation to any person outfitted with either a thin client that is connected to the server as well or the possibility to watch the process on the server application itself.</li> <li>• After the operation, the server application and the Near-Eye Display Device are disconnected and may be switched off until further application.</li> </ul>
<b>Remarks</b>	For consultations it should be possible to provide at least video-out, high-quality photo-out and audio-in / -out. Video- and Photo-In are of added benefit but may have to be used sparsely because of data transfer limitations (the more data is transferred simultaneously, the bigger is the challenge of providing adequate performance).

Table 11: Specific Virtual Consultations use case, using the scheme proposed by Rupp et al. (2009).

By allowing the surgeon to trigger a consultation at any given point in time during the surgery, thus giving him the possibility to establish a two-way audiovisual link to one of his colleagues, the time needed for consultations can, in specific cases, be limited to an absolute minimum. The necessity for the consulting surgeon to enter the operating room (which includes extensive cleaning beforehand) is not given anymore. The consulting surgeon is connected to the system as well (either by directly using the server application or by using another client device), which allows him to communicate with and to give his advice to the operating surgeon.

### 3.8 ELICITATION AND IDENTIFICATION

The information gathered in this section was obtained by using two techniques: **observations** and **open interviews**. Specifically the interviews provided helpful insight in what especially the medical staff expects out of the resulting system. The participating parties in these interviews were part of the hospital's medical staff and therefore are on specification level 2 and below (see section 2.5.2 above).

In the following section, the collected requirements will be mentioned only briefly in the way they have been mentioned in the interviews. Documentation, removal of conflicts and formal specification will be provided in section 3.9 below.

It quickly became apparent that the aspects described in the following sections are regarded as core criteria for the system with the aforementioned use cases in mind (see section 3 above).

#### 3.8.1 Hardware Requirements

In addition to software requirements, a small number of hardware requirements have been mentioned as well. They will be described in this section.

- The system has to be physically robust. Although medical appliances are handled with care, physical impacts can occur (e.g. dropping the device). This should not render the device unusable immediately.
- In addition to physical robustness, the system has to be cleanable. Sterility is another issue, but it has to be possible to clean the device thoroughly (if, for example, drops of blood reach it during a surgery).
- The device has to be able to be worn by different people (including surgeons who are already wearing glasses).
- If the trigger for certain functionalities (e.g. taking high resolution pictures) is realized as hardware rather than software, it should too be cleanable and robust.

### 3.8.2 Functional Requirements

What follows in this section are functional software and service requirements that have been mentioned during the interviews and observations in the elicitation phase.

- A clear need for the transmission of video became apparent during the interviews. Similar wishes have been mentioned for audio transmission as well.
- Similar requests have also been made for high-resolution photos.
- The system should ideally interface (or extend) the already established Hospital Information System. For this, the example of automatically adding recorded pictures to a patient's health record has been brought up by stakeholders.
- Every action (be it video transmission, audio transmission, etc.) has to be cancelable at any given point in time.
- If transmission of video and audio is possible, recording should be too.
- The system should be able to interface with existing medical appliances using standardized interfaces.
- The system should in all cases be able to at least gather visual data from medical appliances.
- The system should be able to provide video (and possibly audio) data to another (consulting) surgeon.
- The aforementioned consultations should ideally work over the internet as well.
- The system should be able to be controlled from a PC by the surgeon's unsterile assistance (who has to interact with the PC in any case for live documentation of the surgery).

### 3.8.3 Non-Functional Requirements

In this section, the non-functional requirements that have been brought up during the elicitation-phase will be listed and described.

- The video quality has to be "good enough" (various resolutions have been tested and for motion picture, an absolute minimum of 640x480 pixels has been identified).
- The system should be quick and responsive.
- The system should start quickly (OP-time is extremely expensive).

- The system should be simple (few clicks / menus for every action).
- The system has to ensure data security and privacy at all times.
- The system should be able to sustain video / audio transmission for a while (at least 2-3 hours).

Of course, the requirements in these sections are very vague, as they are directly taken out of the interview- and observation protocols. Because of this reason, they will be specified, analyzed and documented in greater detail in the following section (see 3.9 below).

### 3.9 ANALYSIS AND SPECIFICATION

In this part of the thesis, the aforementioned requirements will be analyzed, any conflicts between them will be removed, and finally they will be formally specified.

However, before the requirements can be specified in a systematic way, they have to be sanitized. To achieve this, the SOPHIST-REgelwerk method (see section 2.5.3 above) has been utilized. Documentation is then performed using the requirement specification method discussed in the theoretical part of this thesis (see 2.5.4 above).

#### 3.9.1 Indexing Format

As can be seen in sections 3.9.3 - 3.9.5, the requirements have been indexed using a specific format. This schema will be explained in the following:

The index looks like this: FR-OBL-4. As can be seen clearly, the format consists of three distinct parts (A-B-C). These will now be explained along with their possible values:

- A) A shortened form of the type of requirement that is defined (e.g. Functional Requirement). Possible values are: **FR** (Functional Requirement), **NFR** (Non-Functional Requirement) and **HR** (Hardware Requirement)
- B) A shortened indicator showing the severity of the requirement (e.g. implementation is obligatory). Possible values are: **OBL** (obligatory), **OPT** (optional), **FUT** (obligatory in a future version)

- C) A unique global identifier to allow for identification of requirements, even if they change during Requirements Engineering or system development.

### 3.9.2 Definitions

This section provides an overview (see Table 12) over the terminology that was used in the following sections.

Expression	Definition
System	The system is regarded as “the system” as soon as the connection between client and server has been established.
OP-Staff	All personnel that is present during surgeries, including sterile and non-sterile actors.
Core system components	Glass Server and Glass Client (see Figure 32).
Non-authorized personnel	Everyone who is not permitted to view the data without the patients express permission.
System data	All data that is either generated, received, transmitted by or somehow associated with the system.
Medical cleaning procedures	Standard cleaning procedures for medical appliances that do not directly come in contact with the patient.
Hardware part	Any hardware part that is part of the system. The PC, the Near-Eye Display Device and possibly additional remote controls.
At any given point in time	Every point in time between the establishment and termination of the connection between the system’s components.

*Table 12: Definitions of the terminology used in sections 3.9.3 - 3.9.5.*

### 3.9.3 Functional Requirements

What follows in this section of the thesis are the functional requirements briefly mentioned in section 3.8.2 above, specified using the pattern proposed by Rupp et al. (Rupp, SOPHISTen 2009) and the indexing scheme outlined in section 3.9.1 above.

**Obligatory present and future Functional Requirements:**

- FR-OBL-1 At any given point in time the system shall provide the OP-Staff with the ability to transmit video to a screen inside the operating room.
- FR-OBL-2 At any given point in time the system shall provide the OP-Staff with the ability to acquire high-resolution images.
- FR-OBL-3 At any given point in time the system shall provide the OP-Staff with the ability to end whichever transmissions are currently active (Video / Audio, In / Out).
- FR-OBL-4 At any given point in time the system shall provide the surgeon with the ability to receive visual data from medical appliances inside the Near-Eye Display Device.
- FR-OBL-5 At any given point in time the system shall provide the surgeon's unsterile assistance with the ability to fully control all functionality of the system.
- FR-OBL-6 At any given point in time the system shall provide the surgeon with the ability to receive audio from another party who is connected to the system.
- FR-OBL-7 At any given point in time the system shall provide the OP-Staff with the ability to transmit high-resolution images to another party who is connected to the system.
- FR-OBL-8 At any given point in time the system shall provide the OP-Staff with the ability to transmit video to another party who is connected to the system.
- FR-OBL-9 At any given point in time the system shall provide the OP-Staff with the ability to transmit audio to another party who is connected to the system.
- FR-OBL-10 At any given point in time the system shall provide the surgeon with the ability to receive high-resolution images from another party who is connected to the system.
- FR-OBL-11 At any given point in time the system shall provide the surgeon with the ability to receive video from another party who is connected to the system.
- FR-OBL-32 During video transmission the system shall provide the surgeon's unsterile assistance with the ability to choose between resolutions and compression levels of the video image (see FR-OBL-1, FR-OBL-8 and FR-OBL-11).

- FR-OBL-33 At any given point in time the system shall provide the surgeon's unsterile assistance with the ability to choose between resolutions and compression levels for the pictures (see FR-OBL-2 and FR-OBL-7) that can transmitted.
  
- FR-FUT-12 The system will be able to record the video stream sent by the Near-Eye Display Device.
- FR-FUT-13 The system will be able to record the audio stream sent by the Near-Eye Display Device.
- FR-FUT-14 The system will be able to record the video stream sent to the Near-Eye Display Device.
- FR-FUT-15 The system will be able to record the audio stream sent to the Near-Eye Display Device.

**Optional Functional Requirements:**

- FR-OPT-16 At any given point in time the system should be able to provide the OP-Staff with the ability to transmit video to a screen outside the hospital.
- FR-OPT-17 The system should be able to store recorded images inside the Hospital Information System.

### 3.9.4 Non-Functional Requirements

What follows in this section are the non-functional requirements briefly mentioned in section 3.8.3 above, classified formally.

**Obligatory present and future Non-Functional Requirements:**

- NFR-OBL-18 At any given point in time the system shall react visibly on any single interaction within 0.5 seconds.
- NFR-OBL-19 When the user starts a functionality the system shall provide this functionality within 2 seconds.

- NFR-OBL-20 When started, the system shall be responsive within 0.5 seconds.
- NFR-OBL-21 When initiated by the user, the core system components shall establish connection within 5 seconds.
- NFR-OBL-22 At any given point in time the system shall ensure non-authorized personnel cannot access system data.
- NFR-OBL-23 The system shall be able to sustain continuous video transmission for a minimum of 2 hours.
- NFR-OBL-24 The system shall be able to sustain continuous audio transmission for a minimum of 2 hours.
- NFR-OBL-25 The system shall be able to provide video streaming with a resolution of a minimum of 640x480 pixels.
- NFR-OBL-26 The system shall be able to provide high-resolution photos with a resolution of a minimum of 1920x1080 pixels.
- NFR-OBL-27 The system shall be able to provide video streaming with 25 frames per second.
- NFR-OBL-28 The user shall be able to invoke every single functionality of the system within 5 seconds.

### 3.9.5 Hardware Requirements

What follows in this section of the thesis are the hardware requirements briefly mentioned in section 3.8.1 above, classified formally.

#### **Obligatory present and future hardware Requirements:**

- HR-OBL-29 The OP-Staff shall be able to clean any hardware part of the system using medical cleaning procedures.
- HR-OBL-30 If dropped from a height of 1 meter onto solid ground, the system shall continue functioning.
- HR-OBL-31 Different users, including those who wear glasses, shall be able to wear the Near-Eye Display Device.



### 3.9.6 Conflicts

In this section, potential conflicts between the requirements that have been outlined in sections 3.9.3 - 3.9.5 above will be discussed and solutions for them will be provided.

With the current set of requirements, one potential conflict has been identified:

- NFR-OBL-25 - The system shall be able to provide video streaming with a resolution of a minimum of 640x480 pixels.
- NFR-OBL-27 - The system shall be able to provide video streaming with 25 frames per second.

With the current hardware platform (Google Glass), this is not possible because of the strong (inverse) relation of resolution and frame rate with regard to the video transmission. The limiting factors are the built-in CPU and Wi-Fi module (see section 4.6.1 below). Therefore an additional requirement has been devised:

- FR-OBL-32 - During video transmission the system shall provide the surgeon's unsterile assistance with the ability to choose between resolutions and compression levels of the video image (see FR-OBL-1, FR-OBL-8 and FR-OBL-11).

## 3.10 APPLIED REQUIREMENTS ENGINEERING – CONCLUSION

The sections above discussed the processes as they are currently carried out, the specific use cases that are regarded as central for the system development and the requirements as they appeared in interviews with stakeholders and during observation of processes in daily routines of the medical staff. What follows now is the modeling and design of the system that will satisfy these requirements (see section 4 below).

## 4 DESIGN AND ARCHITECTURE

This part of the Master's Thesis discusses the design of the resulting system. Based on the use cases outlined above and the requirements that have subsequently been identified, the system has been designed. The choices that have been made during the conceptual phase will be explained in detail in the following sections.

### 4.1 THE PLATFORM – GOOGLE GLASS EXPLORER EDITION

The platform for the system that will be developed is Google Glass, specifically the Explorer Edition, as has already been mentioned in section 1.1.2 above.



Figure 30: Exploded Google Glass Explorer Edition, CC BY-NC-SA 3.0 (Torborg, Simpson 2014). Feature outlines have been added.

Output	Input	Sensors	Connectivity	Control
Prism (1)	Camera (2)	Light (3)	USB	Button (5)
Speaker (6)	Microphone (3)	Motion	Bluetooth	Touch Pad (4)
	Touch Pad (4)	Magnetic Field	Wi-Fi	Voice
	Button (5)	Gravity		

Table 13: Google Glass Explorer Edition Features, classified.

The features of Google Glass (see Figure 30 and Table 13) will now be explained briefly along with their functionality, both in general and in medical context. The explanations of the visible

features have been sorted to concur with the order in Figure 30 in which they have been outlined from front to back.

1. **Prism:** The prism serves as a half-transparent screen which is used to display generic visual data.

**Context:** With regard to medical environments, this screen can be used for various purposes, e.g. displaying static or dynamic medical data such as patient records or vital signs (see section 3.3 above).
2. **Camera and Microphone (3):** Both, camera and microphone, can be used to acquire audiovisual data from the device's environment.

**Context:** In medical environments the possibility to transmit live audiovisual data can be used for improved assistance by providing the wearer's point of view (see section 3.5 above) or to enable virtual consultations (see section 3.7 above).
3. *(Not visible)* **proximity and ambient light sensor:** These sensors can be used to react to blinking and to determine if the device is currently being worn.

**Context:** In a medical context, blinking could be used as a trigger for acquiring pictures from the camera.
4. **Touch Pad:** The touchpad can be used to interact with the system via touch gestures (e.g. swiping or tapping).

**Context:** Due to the high standards with regard to hygiene, the touch pad cannot be used in medical environments.
5. **Camera button:** A generic Button which is used for taking pictures during normal operation.

**Context:** Similar to point 4, the camera button cannot be used reasonably in medical scenarios.
6. **Speaker:** The speakers can be used as a means to play audio data (e.g. for calls or consultations).

**Context:** The speaker can be used in medical environments to either gather the attention of the wearer (e.g. the surgeon) or to allow for communication during consultations (see section 3.7 above).

What follows now are sensors and important functionality that is not visible in Figure 30:

- **Various movement and location related sensors** (e.g. Accelerometer, Gyroscope or Magnetometer). (Google Inc. 2013b)

**Context:** In medical environments, gestures like nodding could be used for taking pictures or triggering other functionalities.

- **Voice Recognition:** Voice control is used during normal operation to interact with the system.

**Context:** Unfortunately this functionality is not easily usable outside the immediate Operating System (e.g. inside an application), because of the lack of Trigger-Phrases.

Finally, Glass can interface with other electronic hardware by utilizing **Wi-Fi**, **Bluetooth** and **USB**. The latter is also used for charging the device. (Google Inc. 2014a)

## 4.2 GENERAL STRUCTURE

In this first sub-section of the explanation of the system design, the overall architecture will be discussed.

### 4.2.1 Requirements for the system with regard to its general architecture

The following overall requirements have been identified as key factors for the resulting system. They will be used to devise a global system architecture that will support these conditions.

#### 1. Usability

The system must be easy to use. No extensive training should be needed to teach the OP-Staff how to work with it. This includes, for example, the ability to remote control the Near-Eye Display Device, as the device's interfaces for interaction (e.g. the touchpad) are not sophisticated enough to be used as a means for system interaction with regard to medical processes.

Corresponding Requirements: NFR-OBL-18, NFR-OBL-19 and NFR-OBL-28.

## 2. Performance

The system must be highly performant. It must be able to provide high-quality video, audio and image data, while responding to any and every action by the user in a timely manner.

Corresponding Requirements: NFR-OBL-18, NFR-OBL-19, NFR-OBL-20, NFR-OBL-21, NFR-OBL-25 and NFR-OBL-26.

## 3. Compatibility

The system should be compatible with pre-established medical systems. Interfacing via standardized interfaces (visual link, VGA or, in the future, HL7 / DICOM direct interfaces).

Corresponding Requirements: FR-OBL-4, FR-OPT-17

## 4. Runtime

The system has to be able to sustain its core functionality (e.g. video streaming) for a specific amount of time (e.g. 2-3h).

Corresponding Requirements: NFR-OBL-23, NFR-OBL-24

## 5. Data Security

The data that is processed and / or generated by the system must not be obtainable by unauthorized personnel.

Corresponding Requirements: NFR-OBL-22

### 4.2.2 Possible solutions

With regard to the overall system architecture, there are three possibilities for system design that were explored. They will now be listed and explained briefly.

After the explanation, the requirements outlined in section 4.2.1 above will be used to choose one approach (see Table 14) which will then be further explained in section 4.2.3 below. What follows is the list of possible system architectures along with brief explanations.

- **Fat-Client architecture:** The traditional “fat-client” architecture describes a system that is fully self-contained. In this particular case, the system would be implemented

in its entirety on Google Glass as a platform (as the Near-Eye Display Device is the central part of the system).

However, especially due to the limited capabilities of Google Glass, or Near-Eye Display Devices in general, which are of course always subject to the size of the built-in hardware with regard to their performance, out-sourcing a part of the necessary calculations can be considered. This can either be onto a local server, or onto the “cloud” that gained significance over the last few years.

- **Cloud-computing:** A thin-client architecture with all computationally intensive calculations done on dislocated servers would solve the problem of limited hardware capabilities of Near-Eye Display Devices. However, data security is an extremely delicate topic, especially with regard to electronic health records, where there are highly complex legal obligations to fulfill.
- **Traditional thin-client architecture:** Instead of interfacing with the “Cloud”, a system architecture can be designed that relies on a local server instead of hardware that might be positioned at a distant computing center. This yields the added benefit of being inherently more secure than the cloud-approach but still offers significantly more performance than Google Glass itself.

Architect. Criterion	Fat-Client	Cloud	Thin-Client
Usability	◐	◐	●
Performance	◐	●	●
Compatibility	◐	◐	●
Runtime	◐	●	●
Data Security	●	◐	●

Table 14: Comparison of different system architecture alternatives.

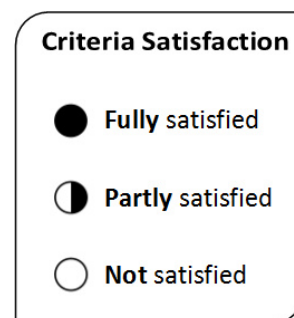


Figure 31: Criteria Satisfaction Legend.

The solutions discussed above have been listed in Table 14 with the degree to which they satisfy the criteria that will now be explained briefly:

- **Usability:** The traditional thin-client architecture has the need for a server to be running on hardware in the same network. This server can have a graphical interface thus providing advanced interaction patterns involving mouse and keyboard which are much more versatile than the small touchpad of Google Glass that would have to be

used both with the Fat-Client and with the Cloud approach (assuming no third client is developed solely for interaction with the cloud server).

- **Performance:** As the Fat-Client would have to do all computation on the hardware of the Near-Eye Display Device, the other two alternatives are inherently superior.
- **Compatibility:** Running the server-application directly on a PC inside of the operating room (which is already present as a Terminal for the Hospital Information System) provides an easy way to interface with medical appliances, as they can be connected over interfaces to the PC.
- **Runtime:** The less calculations are done on the Near-Eye Display Device, the less energy is consumed, which in turn results in prolonged battery life.
- **Data Security:** The need for an Internet-connection with the Cloud-Approach makes it inherently less secure.

#### 4.2.3 The choice behind the Thin-Client-Architecture

A traditional “Fat-Client” design pattern does not need any additional hardware except for the machine the system runs on. However, this means that all features must be implemented directly on the system which is inherently more taxing on the (client-system’s) hardware than only a minimal implementation.

If, however, some of the computational work can be done on a different piece of hardware, ideally a more powerful one, often an overall improvement in terms of performance can be achieved.

As Google Glass is, compared to traditional X86 (or X64) hardware, fairly limited in its computational capabilities, a Thin-Client model has been chosen. This allows the system to be split into two fundamental parts: **Glass Server** (running on the more powerful hardware) and **Glass Client** (running directly on Glass). Both applications communicate over the network, utilizing well established protocols such as TCP and UDP.

This system architecture leads to a minimal overall system consisting of the following elements (see Figure 32):

- A PC with the server application (Glass Server). All computationally intensive tasks are done on this hardware. In addition to this, most of the system interaction will be done using the interfaces provided by Glass Server as well.
- Google Glass with the client application (Glass Client). A minimal client implementation, which serves as mere *extension* of the server (providing a means of transmitting and receiving audiovisual data).
- A wireless network for TCP and UDP-Communication with at least two ports open.

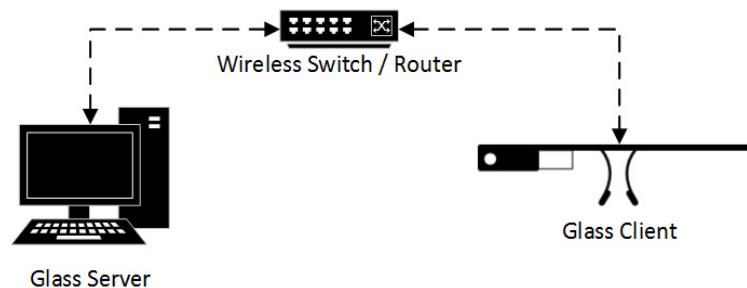


Figure 32: Schematic overview over the (minimal) overall System.

### Why is this design superior in this particular case?

- Demanding tasks are done on hardware that is far more powerful. This leaves less work for the client which runs on the limited platform (Google Glass).
- The fact that less computational work is performed on Google Glass reduces the heat generated by the device. There are no fans in Google Glass. If the device runs too hot, the integrated parts reduce clock speed to avoid overheating, which then results in an overall decrease of performance.
- Google Glass is fairly limited with regard to user interaction. Very few buttons, a touchpad and the lack of a way to use “trigger-words” (specific word combinations that trigger voice recognition and control) inside of an application make it extremely hard to design a functional, yet versatile user interface. The Thin-Client Design allows complex tasks to be remotely controlled by the server application (Glass Server) using well-established interaction-paradigms (based on Mouse and Keyboard).



### 4.3 SYSTEM, CONTEXT AND ENVIRONMENT

Based upon the classification by Rupp et al. (2009), the system and its environment has been divided into the following three fundamental parts: System, Context and Environment (see Figure 33). These categories, along with their elements with regard to this particular system will be explained in greater detail below.

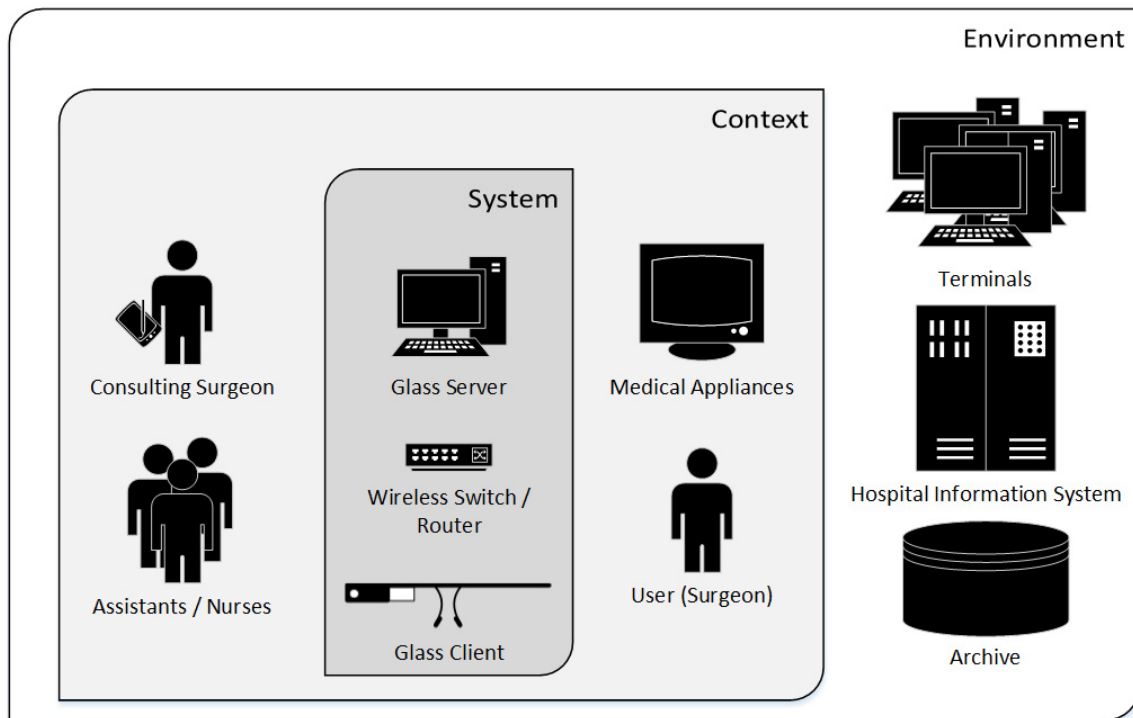


Figure 33: Schematic overview over the elements of System, Context and Environment, based on the classification scheme by Rupp et al. (2009)

- **System:** The system itself and all parts of it.
  - Glass Server (Local Server) holds the full implementation of the system allowing the client to be minimal.
  - Glass Client (Google Glass) is an extension of the server. Providing the wearer with audiovisual data from the system and the server with audiovisual data from the wearer's point of view.
  - Wireless Switch (or wireless router) is used to establish the network connection between Glass Server and Glass Client.
  - In some specific cases (e.g. consultations), another client might become also part of the system. This then again would be a thin client with a minimal implementation leaving the computationally intensive tasks to Glass Server.

- **Context:** Everything that comes in direct contact with the system.
  - Medical Appliances, which provide data over standardized interfaces (e.g. a heart-rate monitor, see section 3.3 above)
  - Assisting staff who uses data that is generated by the system (e.g. for improved assistance, see section 3.5 above)
  - The surgeon's (unsterile) assistance who controls the system by interacting with the server.
  - Consulting surgeons who may benefit from data generated by the system as well (see section 3.7 above).
- **Environment:** Everything that can be considered important, but does not come in direct contact with the system (yet).
  - Most notable in this category is the Hospital Information System. With regard to information flow, this system is the most important part of the hospital and would therefore be very interesting to interface with (see section 7.4 below).

## 4.4 GLASS CLIENT

The following section will describe the client that runs on the Google Glass platform. As has already been mentioned above (see section 4.2 above), Glass Client is merely a thin client. The choices behind its design and remarks with regard to the actual development for the platform will be provided in the following sections.

### 4.4.1 Requirements for the Client Implementation

With Google Glass as the platform, the choices for programming language and APIs are extremely limited. One requirement, which will already suffice for the choice made in section 4.4.2 below, is the transmission of real-time video and photo data. Other requirements will not be listed in this paragraph, as the criterion mentioned before (video transmission) can only be achieved with a single development alternative.

#### 4.4.2 Possible solutions

There are three fundamental ways to develop for Google Glass as a platform at this point in time:

- **The Glass Development Kit (GDK):** Java-based Development Platform very similar to the Android Development Kit (ADK). In fact, GDK is an extension of the ADK.
- **The Mirror API:** RESTful API based development on top of typical web development paradigms.
- **Hybrid Glassware:** MirrorAPI applications that can invoke GDK-counterparts.

The requirement mentioned in section 4.4.1 above, the transmission of real-time video and photo data, is absolutely impossible to achieve with the MirrorAPI, as it does not allow real time communication with the camera and microphone modules (cf. (Google Inc. 2014b)). Therefore, the **GDK approach** had to be used.

#### 4.4.3 The choice behind the GDK and the Immersion

The client has been designed with the “**Immersion**”-Pattern (Google Inc. 2014c) in mind. This requires the GDK as a basis (with Java as the development language) and provides a way to completely block out all other interfaces thus being able to take complete advantage of the screen and the built-in sensors and devices.

The GDK is the only development platform on Google Glass that allows for real-time low-level API access (for interfacing with the camera and the microphone device). This is a fundamental condition for the continuous transmission of video and audio data.

### 4.5 GLASS SERVER

As with the Glass Client in the sections above, this section will describe the Server-part of the system that runs on a PC. The design choices behind the architecture, the programming platform and the GUI-Choice will be explained in the sections that follow.

For the Glass Server application, there are no restrictions with regard to the programming platform.

The server has been designed with **C#** and the current version of **.NET** (4.5.1) in mind. However, no techniques were taken into consideration, which could not be implemented using other languages as well. C# was the primary choice because of its ability to rapidly develop applications with graphical user interfaces. These user interfaces are necessary for various parts of the system, including user interaction (cf. FR-OBL-5) and video display (cf. FR-OBL-1). With regard to performance, the Proof-of-Concept (see section 5 below) has been tested on a number of different devices, including Desktop PCs and Laptops and the performance has been deemed sufficient, even though C# is no native programming language and therefore has inherent performance disadvantages.

The other obvious choice would have been **Java**, especially since the client is already implemented in Java with the GDK as the underlying platform. However, since the project partner (Krankenhaus der Elisabethinen Graz GmbH) and all other people involved in the project primarily use Windows as an operating system, platform-independence was not an issue. Therefore C# has been chosen, as it provides a number of advantages with regard to system API-Access (because of .NET) and the development of graphical user interfaces (Windows Presentation Foundation - WPF). **WPF** has been chosen as the front-end development platform, as it is regarded as the successor to WinForms and thus is the standardized way to develop user interfaces for C# applications on Windows.

As an overall development strategy, the Model/View/ViewModel (or **MVVM**) (Gossman 2005) pattern has been utilized. MVVM describes a very elegant way to separate graphical user interfaces from application logic and is widely used with regard to C# WPF applications.

The Model/View/ViewModel design pattern consists of three distinct parts: **Model**, **View** and **ViewModel** (see Figure 34). (Gossman 2005)

- **View:** In clean MVVM, the View does not contain any logic at all.
- **Model:** Everything that is not directly related to the user interface (e.g. data layer and application logic).
- **ViewModel:** Combines View and Model. The View “binds” itself to the ViewModel, which instantiates and uses the classes provided by the Model.

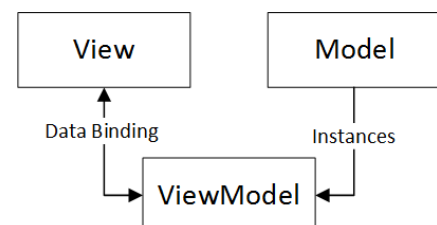


Figure 34: Schematic overview over the relation between View, Model and ViewModel in the MVVM-Pattern as it is described by Gossman (2005).

The concept of **Data Binding**, which essentially, through the usage of an Observer-Pattern, enables the user interface to update itself, allows for the View to be completely without background-logic. This facilitates the development of rich graphical user interfaces, which in the case of WPF are written using a language called XAML. This is especially useful, when applications are developed for a number of devices (e.g. Smart Phones, Tablets and Desktop PCs), as the actual logic of the application can (ideally) be absolutely identical, yet look completely different on every platform.

## 4.6 INTERFACES

It immediately becomes clear that the design outlined in section 4.2.3 above requires the definition of interfaces for communication not only internally – between Glass Client and Glass Server – but also externally, with medical appliances and similar equipment. These interfaces will be explained in greater detail in the following two sections.

### 4.6.1 Internal Interface – Glass Messaging Protocol

As has already been stated, internal communication between Glass Server and Glass Client is vital for the overall functionality of the system. The chosen platform – Google Glass – does only support the IEEE 802.11b/g specification for wireless data transfer at this given point in time (Google Inc. 2014a) and thus can only perform within the tight limits of the respective standards which are a maximum of 11 Mbit/s (1.375 MiB/s, IEEE 802.11b) and 54 Mbit/s (6.75 MiB, IEEE 802.11g). (LAN/MAN Standards Committee of the IEEE Computer Society 2000, LAN/MAN Standards Committee of the IEEE Computer Society 2003)

However, it should be noted, that at the time of writing, only an early prototype (the so-called “Explorer Edition”) was available (Google Inc. 2014d). Future versions, including the actual consumer version, could and likely will improve regarding connectivity.

Precisely because of this reason, a custom binary messaging protocol has been created: The **Glass Messaging Protocol (GMP)**. The advantage of such a custom binary protocol is the ability to avoid virtually any overhead that is imposed by other structures (like XML / JSON based messaging formats). The details of the communication protocol will be discussed in the sections below.

#### 4.6.1.1 Handshake

In order to be able to establish the initial connection, the two devices, **Glass Client** and **Glass Server**, must find each other in the network. Unfortunately, because of the limitations regarding usability in Near-Eye Display Devices, this is not a trivial task.

##### 4.6.1.1.1 Possible Solutions

With regard to the location of the devices in the network, various solutions have been explored and will be briefly explained below. A set of requirements has been devised and the concepts will be evaluated using them (see Table 15). What follows is a list, including brief explanations, of the explored solutions:

- **Fixed IPs:** It would suffice, if one of the two devices has the IP of the other which then can be used to establish a TCP-connection as a basis for all further communication. The trivial way to do this would be a hard-coded IP-Address in one of the devices.
- **QR-Codes:** Similar to the procedure that is used to add new Wi-Fi connections to Google Glass, a QR-Code could be used to obtain the IP of the server. The server application would open a Socket and would then encode the information of the Socket (IP and Port) in form of a QR-Code and display it. The user then scans the QR code after initially launching the Glass Client to obtain the IP and Port of the server which then can be used to establish a TCP-connection for further communication.
- **Local DNS:** A local domain name system along with “fake domains” could be used to mask the dynamically generated IPs with a static domain name.
- **Local UDP-Beacons:** The server could use periodic local (xxx.xxx.xxx.255) message broadcasts to provide its position (IP and Port) in the network. The client listens for these broadcasts and thus is able to establish a connection to the open socket of the server.

Solutions Criteria	Fixed IPs	QR-Codes	Local DNS	Local UDP- Beacons
Efficiency	●	○	◐	●
Usability	●	◐	●	●
Simplicity	●	◐	●	◐
Maintainability	●	●	○	●
Portability	○	●	●	●

Table 15: Possible handshake concepts along with their advantages and disadvantages.

Table 15 gives a quick overview over the various solutions discussed above and their degree of satisfaction with regard to the criteria that will be explained in the following:

- **Efficiency:** The connection must be established quickly and efficiently. All automatic methods have an inherent advantage over methods that require user interaction with regard to the time needed until the connection can be fully established.
- **Usability:** Similar to the *Efficiency* requirement, the key factor behind this is too the interaction with the user. The less interaction is needed, the simpler the process for the user.
- **Simplicity:** All approaches are fairly manageable with regard to their implementation. However, especially QR-Code-Scanning is significantly more time consuming to implement, as Google seems to be unwilling to provide their QR-Code API which is used for adding new Wi-Fi networks at this time of writing.
- **Maintainability:** The less the solution relies on the necessity of additional hardware and software, the easier it is to maintain.
- **Portability:** The system has to function on many different networks. Hard-Coded IPs are impossible to use on all networks, as policies do not always allow for specific hardware to consistently use the same IP in the network.

#### 4.6.1.1.2 The choice behind the UDP Message Beacon

The solution that finally went into the system design is a little more complicated than fixed IPs but has the advantage of being completely automatic, thus requiring no user interaction at all. As has already been briefly explained in the overview provided in section 4.6.1.1.1 above, the general concept behind this approach can be summarized as follows:

- The Server provides a continuous messaging beacon in form of a local broadcast packet that is sent repeatedly on a pre-shared port (e.g. 9095). It contains the socket information (IP and Port) of the server.
- The client can listen for this beacon, extract the IP and Port of the server and use this information for the establishment of a proper two-way connection.

The process will now be explained in a more detailed manner in the remaining part of this section.

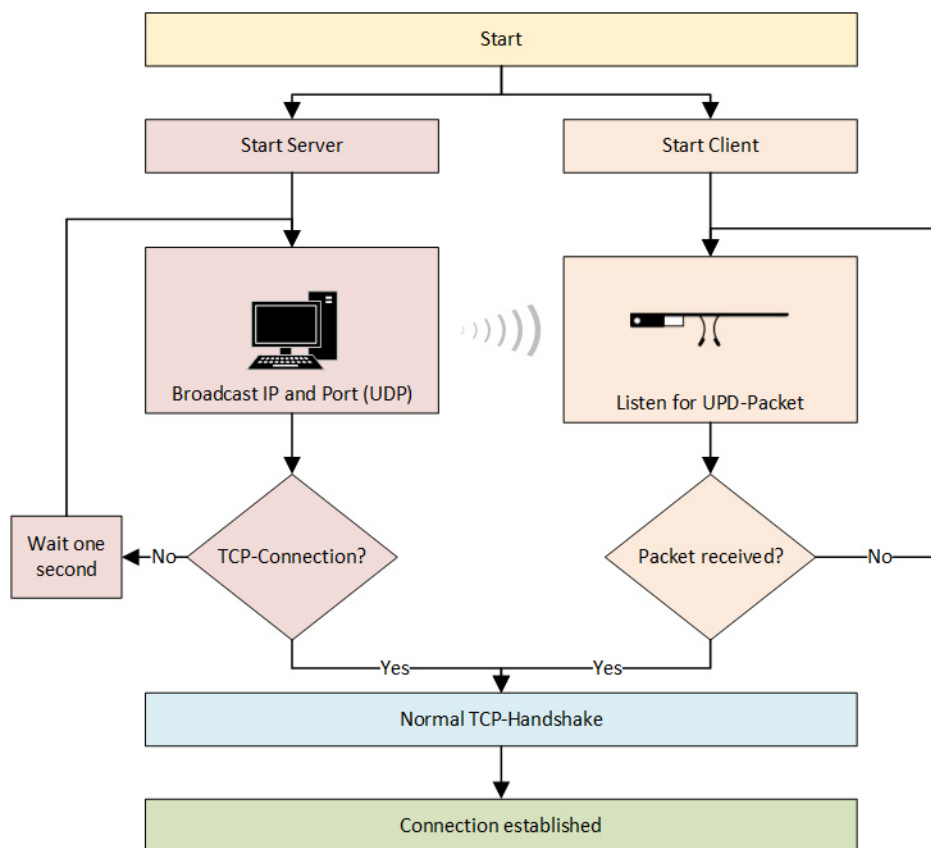


Figure 35: Schematic overview over the Glass Messaging Protocol connection establishment.

As outlined in Figure 35 above, both applications need to be started to find each other. Upon system start both immediately start a loop in which they reside until the TCP-connection has been established.

The Server-Loop contains the continuous transmission of a simple UDP-Packet containing its IP (the IP of the Network Interface Card that is connected to the local wireless network) and a Port (which defaults to 9090). The structure of the packet is very similar to the one of the



messages outlined in the Glass Messaging Protocol (see section 4.6.1.2 below). It consists of a Marker-field for identification and the IP and Port (see Figure 36).

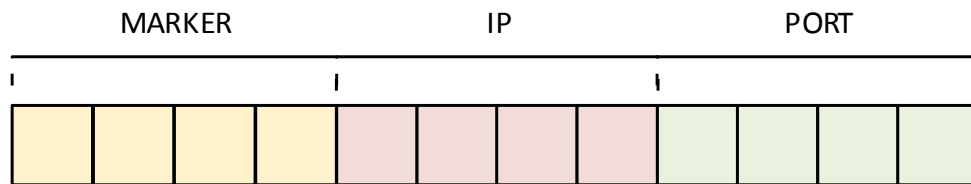


Figure 36: Contents of the UDP-Beacon Packet

The included IP and Port allow the Glass Client to establish a proper TCP-Connection with Glass Server. To be able to receive this packet, a UDP-Socket is opened on the client. This socket then waits for the correct packet to arrive. As soon as the packet arrives, the IP and Port can be used to establish the TCP-Connection. When the TCP-Connection has been established, the UDP-Beacon is disabled and arbitrary messages can be sent back and forth. The handshake is now complete and general communication can begin.

#### 4.6.1.2 General Communication

For the reasons outlined in section 4.6.1 above, a custom communications protocol, the **Glass Messaging Protocol** (GMP) has been developed. This way, communication overhead can be reduced to a minimum. As the protocol is built on top of already established protocols on the transport layer (OSI Layer 4 (Zimmermann 1980)), it was necessary to introduce a message-based form of communication, as the underlying protocol (TCP) is stream-based. To achieve this, a general message consists of several fields containing information about the contained data. The following table provides an overview over the fields (see Table 16):

Field name	Field content
Marker	A specific <i>magic number</i> which indicates the start of a message (4 bytes).
Type	A single byte which indicates the type of the message.
Size	An unsigned integer (4 bytes) which contains the length of the payload in bytes.
Payload	The binary data of the message (containing text, image, audio, etc.).

Table 16: The different fields in a GMP-message and their content.

These fields are then encoded into a binary data stream and sent to the Glass Client or Glass Server at the other end of the connection. Thus, a message can be visualized the following way (see Figure 37):

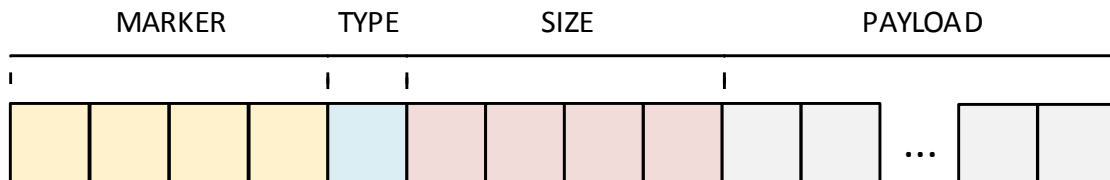


Figure 37: Schematic representation of the messages in the Glass Messaging Protocol.

Both clients scan the incoming traffic for the Marker until it has been fully identified as a continuous sequence in the network stream. As soon as this happens and thus the beginning of a message has been identified, further parts of the message can be read into pre-specified buffers. The size-field is used to determine the end of the message.

This has one significant advantage: Because of the transmission of the actual message size, internal network buffers can be used to their maximum and therefore performance does not need to be limited by reading single bytes at a time during the transfer of a message. If, however, the system cannot identify a message correctly, it automatically can fall back to scanning for the Marker (which does, in fact, read one byte at a time). During normal operation and by using a reliable protocol like TCP, the scanning should only ever occur at the beginning of each message, thus maximizing the performance by switching to large buffers whenever possible.

Of particular interest with regard to the Glass Messaging Protocol is the Type-Parameter along with the Payload-field. In contrast to the Marker-field in the beginning, which is merely a sequence of four bytes, and the Size-field, which is a 32-Bit Integer, the Type-field actually carries information not only about the nature of the message but also about the way the data is encoded in the Payload-field.

The following table (see Table 16) gives an overlook over the different types of messages and their payloads.

Message-Type	Message-Payload
Text	Text encoded as UTF-8 byte stream.
Image / Video	The image data encoded as JPEG byte stream. As the video transmission is Motion JPEG (MJPEG), the byte stream in a video message is JPEG as well. The built-in APIs for video and audio streaming have proven to be extremely unreliable with regard to quality and consistency of transmission. This is further corroborated by the fact that a recent update for Google Glass removed the native video call functionality, as it did not satisfy Google's quality standards (Google Glass Team 2014). Therefore MJPEG has been devised as a custom solution. The implementation of an advanced video codec like MP4 could increase efficiency even more (see section 7.3 below)
Request	Contains just a single byte containing the nature of the request. For further information, please refer to the code documentation.
ImageQuality	The first byte contains the target (Video 0, Photo 1), an integer containing the width follows (4 byte), then another integer containing the height (4 bytes). An additional integer containing the compression quality (0 – 100; 4 bytes) follows and finally another integer which contains the frame limiting rate (0 – 30; 4 bytes).
ImageResolutions	A continuous stream of width / height pairs (8 byte each, 4 byte per width / height).
Audio	A raw pulse-code modulation byte stream (8000 Hz, 16 bit, mono).

Table 17: The different message types and their payload.

## 4.6.2 External Interfaces

This section describes the various ways how data can be transmitted into the system and how it is then used throughout the service.

### 4.6.2.1 Glass Client

For the sake of completeness, the interfaces that are used to obtain and emit data will now be discussed briefly in the points below:

- **Camera**

To be able to provide video transmission and high-resolution still-frames, it is necessary to obtain the data on the client side. For this purpose the built-in camera of Glass is used. By utilizing the provided low-level APIs (Google Inc. 2014e), real time binary image data can be obtained from the camera module.

- **Microphone**

Similar to video and photo outlined above, sound can be transmitted from the client to the server as well. In order to do this, the built-in microphone in Glass can be utilized. The provided APIs (Google Inc. 2014f), again low-level binary data interfaces, are used to obtain pulse-code modulation data.

- **Display and Speaker**

Any data that is received has to be made available to the user. The built-in prism and speaker are used for video / image and audio rendering.

#### 4.6.2.2 Glass Server

Apart from the client itself, which is able to provide video and audio data, the server has to be able to provide data obtained from its surroundings as well. These interfaces are outlined below:

- **Generic video interface**

Perhaps most important for the system, especially in its early stages, where users are not yet accustomed to the service, is the generic video interface. The interface is constructed in such a way that it is able to obtain images, either still or in motion, from a common generic source plugged into the PC which is connected to the system (e.g. the PC running Glass Server). This allows for the system to provide visual data, in the way the surgeons are used to seeing it, directly from the medical appliances that produce them by either filming the screen via, for example, an USB-Camera or by directly interfacing with the image signal (e.g. VGA or similar standards) using hardware interfaces such as VGA Interface Cards.

- **Direct access to medical data in the Hospital Information System**

There are different proprietary and non-proprietary standards (e.g. HL7 (Health Level Seven International 2013)) which provide a way to interface directly with the hospitals information system. With the use cases mentioned above (see section 3 above) as the basis for the system's design, these standards were not part of the focus of this thesis. However, as these protocols might become part of the system in the future, it has been

designed in a way which makes it easily extendable (e.g. the generic way in which the communication has been implemented, see 5.1.3 below).

## 5 IMPLEMENTATION – PROOF-OF-CONCEPT

---

The system that was described in sections 3 and 4 above has been implemented in its fundamentals to proof the viability of the underlying concepts (e.g. proof that video, audio or general data transmission is possible using a local network only). This implementation will now be discussed briefly in the following sections.

As has already been described in section 4.2 above, the system has been divided into two fundamental parts: Glass Server (see Figure 38) and Glass Client.

The goal behind this Proof-of-Concept implementation was to prove the viability of the solutions that have been devised during the Requirements Engineering in section 3 above. Therefore the central motivation is that all the technical aspects, which have to be available for the use cases (see Figure 13) to be possible, have been implemented at least fundamentally.

The absolutely vital functions, based on the requirements outlined in section 3.9.3 above, are therefore:

- Video and Audio In and Out (see FR-OBL-1, FR-OBL-6, FR-OBL-8, FR-OBL-9, FR-OBL-11, FR-OPT-16, NFR-OBL-23, NFR-OBL-24, NFR-OBL-25 and NFR-OBL-27)
- Remote Control (see FR-OBL-3, FR-OBL-5, FR-OBL-32 and FR-OBL-33)
- Displaying arbitrary data in the Near-Eye Display Device (see FR-OBL-4)
- High-Resolution Still Frames (see FR-OBL-2, FR-OBL-7, FR-OBL-10 and NFR-OBL-26)
- Local LAN without Internet and without Google Services (see NFR-OBL-22)

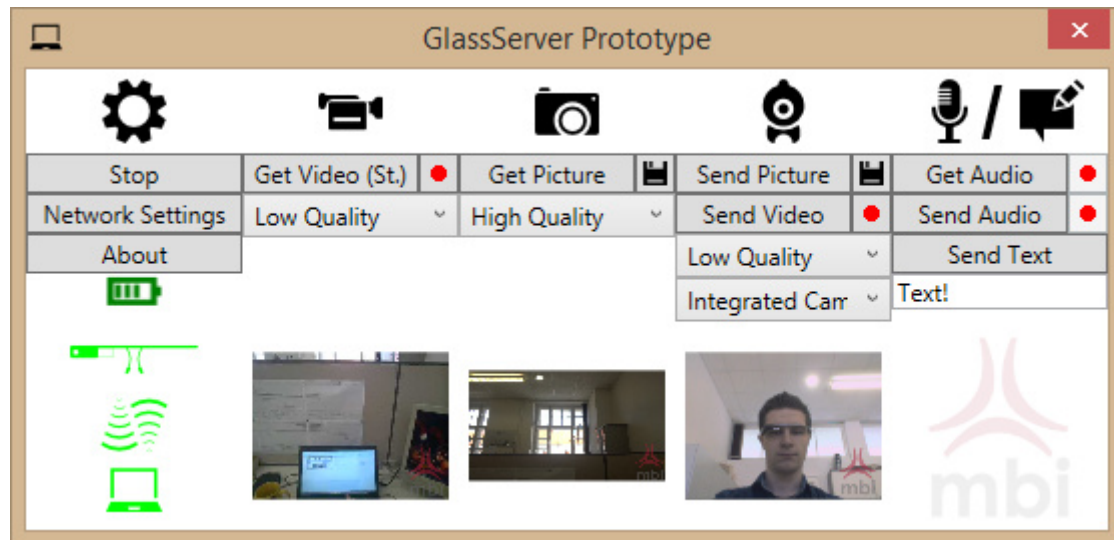


Figure 38: Screenshot of the Glass Server Proof-of-Concept.

## 5.1 GENERAL CHOICES

In this section, general development choices behind Glass Server and Glass Client will be explained. For a more detailed explanation of the system and the code behind it, please refer to the code documentation.

### 5.1.1 Glass Server

As has already been outlined in section 4.5 above, the server has been developed using C# with the current version of .NET (4.5.1), Visual Studio 2013 Ultimate and Windows 8.1.

The reason for this choice is that this system is primarily a Proof-of-Concept, so **Rapid Application Development (RAD)** was a factor that went into consideration. C# has very strong support for RAD.

The Proof-of-Concept has been tested on a number of different devices for various durations of time and no problems with regard to performance have been noted. With C# as the language of choice, WPF has been chosen as the front-end development platform. The application has been developed using the MVVM pattern (see section 4.5 above). In addition to the already mentioned platforms and frameworks, two Third-Party-Libraries have been utilized for Camera and Microphone access (see section 5.1.3.1 below).

### 5.1.2 Glass Client

It was already mentioned in section 4.4.3 above that the GDK is necessary as the platform for development of the client application. This results largely out of the inability to access low-level hardware (e.g. Camera and Microphone) with MirrorAPI as the target platform.

Internally, the system is kept to a minimum with regard to complexity, as it is merely a Thin-Client (see section 4.2.3 above). All hardware access is handled through the provided Google-APIs (e.g. (Google Inc. 2014e) and (Google Inc. 2014f)) and the communication is handled in separate classes which provide Callback-Events for the GUI to be able to react to incoming messages (e.g. Video).

### 5.1.3 Communication

The server’s and the client’s communication implementation are very similar and completely encapsulated in their own classes. For reasons of simplicity, only the server implementation will be explained in the following. However, the client implementation is very similar and can be found fully explained in the code documentation.

The message protocol that provides the basis for the communication described in this section has been outlined in great detail in section 4.6.1.2 above.

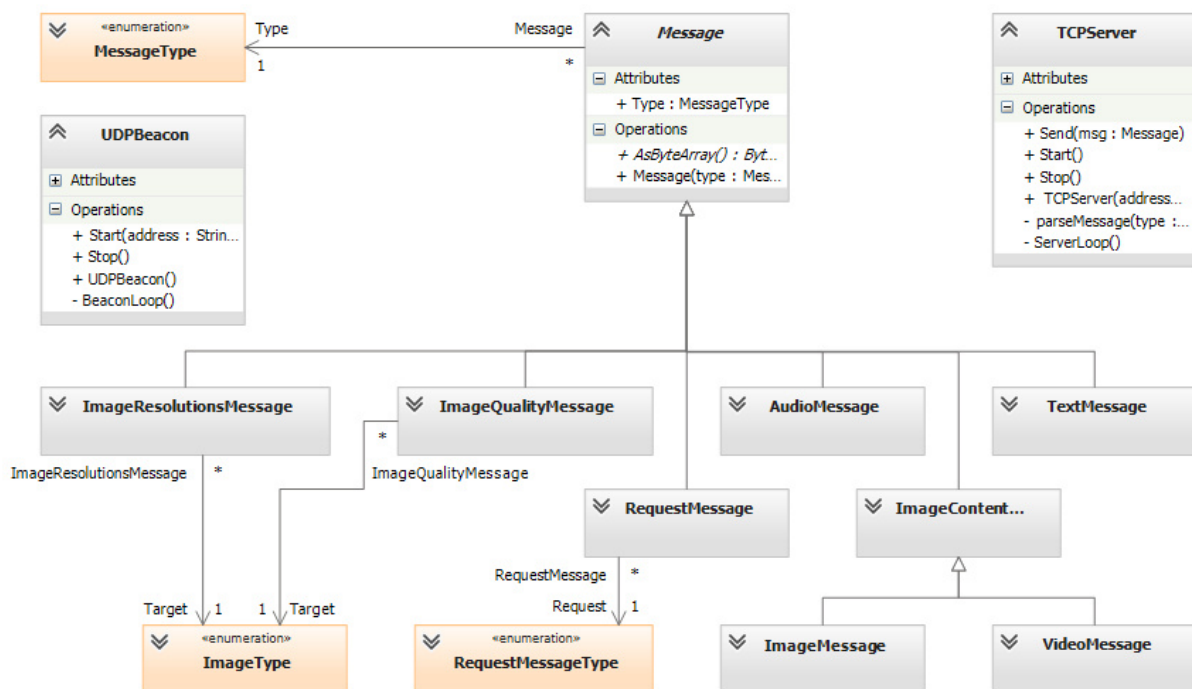


Figure 39: An UML Class Diagram providing an overview over the TCP / UDP-Server-Classes and the Messaging architecture.



Figure 39 provides a compact overview over the classes involved in the communication. The most interesting part is the **Message** class. Its most notable member would be the `asByteArray` virtual method, along with all the constructors of the inheriting child classes, which accept byte arrays as well. This allows for the server to implement a `Send(Message msg)` method that internally can extract the data of a Message by using its capability to export itself to a byte array. This enables the server to handle new messages, without having to implement various different Send-Methods.

Another notable method is the `ServerLoop()`-Method in **TCPServer**. As soon as the connection is established, the server listens in an endless loop on the TCP-Channel in order to receive TCP-Packets containing Message-Data (which is then assembled inside of the loop and passed through a Callback-Method to the UI), until `Stop()` is called.

In addition to this, the **UDPBeacon**-Class can be seen. It is used in the beginning of the `ServerLoop()`-Method of the TCP-Server to broadcast its position as outlined in section 4.6.1.1 above.

#### 5.1.3.1 Third Party Libraries

For the development of the Glass Client application no Third-Party-Libraries were utilized. Glass Server does rely on two external libraries which will be listed below along with their licenses and purpose:

- **AForge.NET**

Used for Camera access and handling on the PC-side. It is licensed under LGPL v3 (Free Software Foundation 2007) and can be found at: <http://www.aforgenet.com/> [last accessed: 2014-04-27]

- **NAudio**

Used for Microphone access and handling on the PC-side. It is licensed under Ms-PL (Microsoft Corporation 2007) and can be found at <http://naudio.codeplex.com/> [last accessed: 2014-04-27]

## 5.2 CAPABILITIES

This section provides an overview over what the Proof-of-Concept currently is able to provide in terms of functionality. In this section the features of the overall system will be described as a whole, without specific differentiation between its server and client part.

### 5.2.1 Glass to Server Video Transmission

The Proof-of-Concept is able to provide a constant video stream from the device (Google Glass) to the server application. The stream can be altered in its quality either by using pre-defined settings like “Low Quality” (320x240x50, the last parameter denotes the image quality), “Medium Quality” (640x480x75) and “High Quality” (1280x720x75), or by using virtually any resolution that is supported by the client device along with dedicated sliders for Framerate-Limiting and Compression Quality (see Figure 40). In addition to this, at any point in time the incoming stream can be recorded using the corresponding button (the recorded file is encoded using MPEG4 compression).

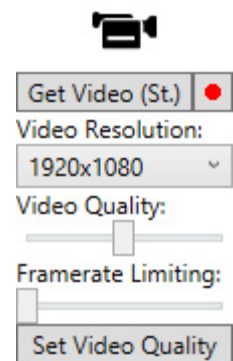


Figure 40: "Expert"-Settings for video transmission.

### 5.2.2 Glass to Server Picture Transmission

On top of the video transmission outlined in section 5.2.1 above, the Proof-of-Concept is also capable of providing high-resolution still frames either during video transmission or during normal operation, obtained through the camera integrated in Google Glass.

Similar to the video-transmission, the picture quality too can be altered using either pre-defined settings like “Low Quality” (640x480x50), “Medium Quality” (1280x720x75), “High Quality” (1920x1080x75) and “Very High Quality” (2592x1944x75), or by using very similar sliders as the ones outlined in section 5.2.1 above, along with the choice of every picture-resolution supported by the camera in the device (see Figure 41). In addition to this, the received picture can be saved using the corresponding button (in which case it will be saved as lossless PNG).

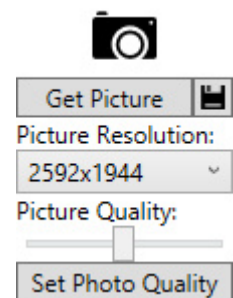


Figure 41: "Expert"-Settings for picture transmission.

### 5.2.3 Glass to Server Audio Transmission

Especially for Virtual Consultations (see section 3.7 above), audio transmission is, in addition to video and picture, as well a key aspect.

The Proof-of-Concept, in its current form, provides the possibility to stream audio from Glass to the server in 8000Hz, 16 bit Mono quality. Using the corresponding button, it is possible to record this audio stream into a 16bit WAV file.

### 5.2.4 Server to Glass Video and Still Frame Transmission

Similar to the outbound video and picture transmission, it is also possible to send a continuous video stream or single frames from the server to the Google Glass Device. The different quality levels provided for both actions are limited to “Low Quality” (160x90x50), “Medium Quality” (320x180x75) and “High Quality” (640x360x75).

Higher resolutions are not supported since they cannot be displayed on the prism integrated in Google Glass, which supports only a maximum of 640x360 pixels (Google Inc. 2014c). By using the “Expert”-View, Framerate-Limiting and Compression Quality can be controlled as already described in section 5.2.1 above.

In addition to this, the server application supports any generic webcam device. If the PC running the application has multiple devices connected (e.g. in case of a laptop: an inbuilt camera and an external USB-Webcam and possibly even an USB-VGA-Interface), a menu is provided to switch between these devices (see Figure 42).

Of course, both, the outgoing video stream, as well as any sent pictures, can be saved to the hard drive using the corresponding buttons (formats are MPEG4 and PNG).

### 5.2.5 Server to Glass Audio Transmission

Similar to the functionality outlined in section 5.2.3 above, it is also possible to transmit audio from the server to the client.

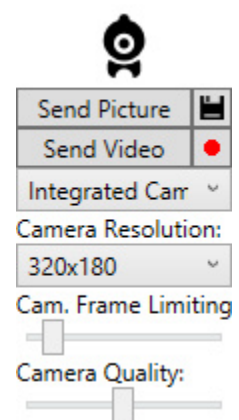



Figure 42: "Expert"-Settings for inbound video and photo transmission.

The Proof-of-Concept provides the possibility to stream audio from the server to Glass in 8000Hz, 16 bit Mono quality. Using the corresponding button, it is again possible to record this audio stream into a 16bit WAV file.

## 5.2.6 Miscellaneous Features

This section contains all features, which are too insignificant to be mentioned in their own section.

- **Indication of current Battery level**

The server application is able provide information regarding the current battery status of the client device. This is indicated by symbols such as  and 🔋.

- **Dedicated Windows**

Clicking on the images on the bottom of the server application opens up the image or video-feed in a dedicated window for closer inspection.

- **Zooming-functionality**

The windows that can be opened by clicking on the images provide the possibility to zoom into the picture or video feed.

- **Server to Glass Text Messages**

As a proof that the system can send arbitrary data between the devices, the possibility to send text messages has been implemented as well.

## 6 EVALUATION

In this part, the fundamental goal of the thesis will be evaluated: The improvement of medical processes by introducing Near-Eye Display Devices. The Proof-of-Concept will serve as the basis for this evaluation. The findings that are described in the following are based on observations of surgeries conducted with the Proof-of-Concept as a supporting tool and on interviews (and questionnaires) with the involved stakeholders before and after the surgeries.

As a test case, a Percutaneous Transluminal Angioplasty (see section 3.3 above) has been performed and has also been discussed by Vorraber et al. (2014).

The interventionist used the Proof-of-Concept to monitor the vital signs of the patient (see Figure 43).

During the observation and conversations with the interventionist before and after the surgery, the following aspects were noted

(and are also, in part, described in the publication by Vorraber et al. (2014)):



*Figure 43: Percutaneous Transluminal Angioplasty. The interventionist faces situations where he has no direct view to the patient or his vital signs. Near-Eye Display Devices allow for continuous monitoring. (Vössner 2014) Circles have been added.*

- The interventionist relied fully on the Proof-of-Concept and did not use the monitoring device in the operating room, which was provided as a backup (marked in Figure 43 with a red circle in the background).
- One particular fact that has been mentioned multiple times by the interventionist is that there are situations, where he has to physically turn away from the monitoring device (see Figure 43). This causes him to lose the line of sight to the patient's vital signs. The Near-Eye Display Device solves this problem by providing the necessary data in a constant spot within the wearer's field of view.
- Another interesting aspect was that the interventionist, who had already finished the process and went to a different room to fill out a questionnaire, suddenly noticed a different pattern within the vital signs which he still was monitoring in the corner of his eye and re-entered the operating room to check on his patient.

- The pattern that has been mentioned before has also been described as a very important factor. The medical staff, across the board, stressed the importance of patterns and pattern recognition. Rather than constantly reading values, surgeons seem to rely strongly on patterns on medical devices. A change is immediately noticed and reacted upon. This advocates the use of interfaces that do not change the way surgeons are used to seeing their data.

In conclusion, this test case showed that Near-Eye Display Devices do, in fact, have the potential to increase efficiency and safety. However, the data cannot diverge significantly from its usual visual appearance as the surgeons rely strongly on this presentation. Thus, with regard to providing visual information, it seems that only the channel may be changed, not its content or its visual representation.

## 7 FUTURE WORK AND CONCLUSION

---

In the short confines of a Master's Thesis it is difficult to completely grasp the scope of a big project, especially one that is to be used in medical environments. Below, a number of suggestions, current shortcomings and future ideas will be discussed.

### 7.1 DATA SECURITY AND STANDARDS CONFORMITY

Whereas technical aspects have been discussed extensively throughout the course of this thesis, other factors such as security and standards-conformity have been only touched upon briefly.

In most countries there are strict standards with regard to medical appliances and software, especially concerning data security and privacy. The latter was already an implicit part of this project through the isolated custom network communication, which does not have the need for the data to leave the network of the hospital. In use cases like remote consultation, well-established techniques like VPN-Networks can be used to ensure data security and integrity by establishing a secure network over the internet. Especially when the system is extended via interfaces to the Hospital Information System, laws and standards must be followed to the letter and strong encryption might become necessary to protect patient data.

### 7.2 FURTHER USE CASES AND FUNCTIONALITY

This thesis mainly focused on five use cases. However, there is tremendous potential for Near-Eye Display Devices. A few ideas that have come up during the creation of this thesis in discussions and research include:

- Triggering functionality with hardware controls: A surgeon could take a picture with a Bluetooth-Footpedal or a remote control.
- Providing static patient data (e.g. Electronic Health Records) to the medical staff for use in ward rounds. The Beth Isreal Deaconess Medical Center in Boston is currently exploring such possibilities. (Halamka 2014)
- Utilizing eye tracking to indicate where the wearer was looking on picture and video streams. This feature is not yet integrated in Google Glass. However, it seems

reasonable to assume that such a feature will eventually be implemented (the corresponding patent has already been filed (Raffle, Wong et al. 2012)).

- Similar to the static data outlined in the Electronic Health Records use case above, planning-pictures might be interesting as well, especially with regard to aesthetic surgery.
- Another interesting aspect would be the observation of students in late phases of their training. The possibility to observe and intervene via an audio link might provide a useful method to judge the students' abilities without alienating him by the physical presence of the examiner, as currently explored by Vallurupalli et al. (2013).

### 7.3 EXTENSION OF PROOF-OF-CONCEPT

The Proof-of-Concept that has been developed during the course of this Master's Thesis is not a complete system. The primary goal behind the development of the proof was to show the feasibility of the ideas behind the system and the use cases.

In addition to the full development of the system, the proof can also be improved in terms of efficiency. Opportunities for such improvement would, for example, be the communication stack and image compression.

Currently, almost all (with the UDP-Beacon and therefore the initial connection-handshake as an exception) communication is done over TCP. However, especially the data intensive parts (images, video and audio) could be transmitted over a secondary UDP-based channel. TCP ensures that every bit arrives at the corresponding device, but it does it at a cost of efficiency. UDP does not have this feature and is therefore generally better suited to transmit video and audio data. One could even go as far as implementing a complete video streaming protocol like RTP / RTSP either directly for the transmission between Glass Client and Glass Server (and therefore open up the possibility for non-system clients to receive the stream as well) or for transmission to other clients with Glass Server as the source (thus keeping the connection between Glass Client and Glass Server exclusive).

Video is currently encoded as Motion JPEG (MJPEG), which means that images are only compressed with regard to their content but not with regard to how the content changes over time. Implementing a different compression standard (e.g. MPEG4) would almost certainly



lead to an improvement in terms of efficiency and overall performance, provided that this compression does not pose too much stress on the integrated hardware in the Near-Eye Display Device. With that in mind, even a two-factor compression would be a possibility, using simple compression algorithms for the transmission of data between Glass Client and Glass Server and advanced compression (and even encryption) algorithms for transmission to other clients, e.g. over the Internet.

#### 7.4 INTERFACING WITH ALREADY ESTABLISHED SYSTEMS

Virtually all hospitals have some kind Hospital Information System which builds the backbone of the information flow between all processes and appliances utilized in the hospital.

In case of the Krankenhaus der Elisabethinen GmbH Graz, this system is called KIS (short for “Krankenhausinformationssystem”). During interviews with various stakeholders, it became apparent that interfacing with this system would be a welcomed feature. One stakeholder went even as far as proposing a complete encapsulation of the resulting system inside this Hospital Information System. This might, however, prove to be very complicated or even impossible to do. However, interfacing with the system itself should in theory be possible. One example would be the transmission of high-resolution photos the surgeon takes during a surgery to the system which then stores them alongside other patient data. Various medical appliances (e.g. the endoscope) currently in use during surgeries offer similar functionalities, hinting at the possibility that there are standards and protocols that could be used to support such a feature.

#### 7.5 CONCLUSION

Requirements Engineering is a very central part of this thesis. One may argue that the traditional (linear) approaches of Requirements Engineering are not used any more in modern software development projects. However, this seems untrue, as there are project-types, especially big ones, which still benefit from thorough planning in the beginning of, and even more so from iterative planning during the development process, as has been outlined in section 2.7.4. Agile Development Methods do, to some extent, provide better tools for specific projects. In addition to this, there seems to be even a grey area, where superiority of agile and

traditional approaches cannot be identified clearly. In these cases, a mixture of both might be advisable (see sections 2.7.2 and 2.7.3).

With the tools provided by (incremental) Requirements Engineering, a system has been designed and a prototype has been developed to prove its viability. At its center, five specific use cases have been identified which were developed by working closely with the people involved in medical processes. The Proof-of-Concept has then been tested with a subset of these use cases (see section 6 above) to validate the requirements that have been previously negotiated with all involved stakeholders (see section 3.9 above). The platform behind the system and the Proof-of-Concept is Google Glass.

Google Glass as a platform, or even similar devices by other manufacturers, hold enormous potential for process improvement in various different environments. The limited set of use cases and the system designed in this thesis are just a small part of all the ideas that are potentially interesting.

With Google as the technology giant behind the development of such Near-Eye Display Devices, it seems reasonable to assume that there will be a lot of development in the coming months and years. Even if Google Glass does not succeed (or does so only in specific branches), there almost certainly will be other devices because of the push Google is making at the moment. The system, processes and use cases discussed in this thesis can be implemented for various different devices. Merely the technical implementation of the Proof-of-Concept has been developed specifically for Google Glass, but is portable to other platforms with moderate time and effort as well.

The central idea behind this kind of device is to simplify well established routines and there is definitely potential for such improvement with regard to medical processes as well, as has become apparent in the various interviews and observations with medical personnel throughout the course of this thesis.

In conclusion, it can be said that Near-Eye Display Devices do, in fact, yield potential for process improvement, including medical processes. There is still a lot of work to do until a final system can be implemented and brought into daily medical routine, as there are certain aspects which have to be brought into consideration (especially legal aspects, security aspects, proprietary interfaces, etc.). The aim of this thesis was to explore possibilities for such

improvement and to prove that a solution with Google Glass as its platform is a viable possibility. However, since there is such a huge potential behind the platform and the processes involved, there are many other use cases that are worth exploring (see section 7.2 above) in the future. It is my personal opinion that Near-Eye Display Devices such as Google Glass and its successors will have a similar impact on existing processes as the Smart Phone did, when it first gathered significance around the year 2007.

## 8 REFERENCES

---

ALFOR, M.W. and LAWSON, J.T., 1979. *Software Requirements Engineering Methodology (Development)*.

BALZERT, H., BALZERT, H., KOSCHKE, R., LÄMMEL, U., LIGGESMEYER, P. and QUANTE, J., 2009. *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. 3rd ed. Spektrum Akademischer Verlag.

BECK, K., 1999. *Extreme Programming Explained*. Addison-Wesley.

BECK, K., BEEDLE, M., BENNEKUM, A.V., COCKBURN, A., CUNNINGHAM, W., FOWLER, M., GRENNING, J., HIGHSMITH, J., HUNT, A., JEFFRIES, R., KERN, J., MARICK, B., MARTIN, R.C., MELLOR, S., SCHWABER, K., SUTHERLAND, J. and THOMAS, D., 2001. *Manifesto for Agile Software Development*. Available from: <<http://www.agilemanifesto.org/>>. [05 March 2014].

BOEHM, B.W., 1981. *Software Engineering Economics*. Prentice-Hall.

BROOKS, F.P., Jr., 1987. No Silver Bullet Essence and Accidents of Software Engineering. *Computer*, **20**(4), pp. 10-19.

CARRILLO DE GEA, J.M., NICOLAS, J., ALEMAN, J.L.F., TOVAL, A., EBERT, C. and VIZCAINO, A., 2011. Requirements Engineering Tools. *IEEE Software*, **28**(4), pp. 86-91.

COPELAND, L., 2001. *Extreme Programming*. [online] Available at: <[http://www.computerworld.com/s/article/66192/Extreme\\_Programming](http://www.computerworld.com/s/article/66192/Extreme_Programming)> [Accessed 22 April 2014].

DAVIS, A.M., 2010. *Requirements Bibliography*. [online] Available at: <<http://www.reqbib.com/index.htm>> [Accessed 12 May 2014].

DICOM STANDARDS COMMITTEE, 2011. *Digital Imaging and Communications in Medicine (DICOM)*. National Electrical Manufacturers Association. [online] Available at: <<http://medical.nema.org/standard.html>> [Accessed 10 May 2014]

DORFMAN, M., BYRNE, E.R., GARCIA, S.M., HARWELL, R.M., MILLER, L., SABOR, B., SWEENEY, T.P. and WHITE, S., 1994. Requirements engineering standardization, *Proceedings of the First International Conference on Requirements Engineering 1994*, pp. 57-63.

DORFMAN, M. and THAYER, R.H., 1990. *Standards, guidelines, and examples on system and software requirements engineering*. IEEE Computer Society Press.

FREE SOFTWARE FOUNDATION, 2007. *GNU Lesser General Public License (Version 3)*. [online] Available at: <<http://www.gnu.org/licenses/lgpl.html>> [Accessed 27 April 2014].

- GLINZ, M., 2007. On Non-Functional Requirements, *15th IEEE International Requirements Engineering Conference 2007*, pp. 21-26.
- GLINZ, M. and WIERINGA, R.J., 2007. Guest Editors' Introduction: Stakeholders in Requirements Engineering. *IEEE Software*, **24**(2), pp. 18-20.
- GOOGLE GLASS TEAM, 2014. #GlassUpdates are back with KitKat for Glass, photo bundles & more. [online] Available at: <<https://plus.google.com/+GoogleGlass/posts/gNS3JvHEdvV>> [Accessed 03 May 2014].
- GOOGLE INC., 2014a. Tech specs. [online] Available at: <<https://support.google.com/glass/answer/3064128>>. [Accessed 01 April 2014].
- GOOGLE INC., 2014b. Add a cat to that. [online] Available at: <[https://developers.google.com/glass/develop/mirror/stories#add\\_a\\_cat\\_to\\_that](https://developers.google.com/glass/develop/mirror/stories#add_a_cat_to_that)>. [Accessed 27 April 2014].
- GOOGLE INC., 2014c. Immersions. [online] Available at: <<https://developers.google.com/glass/develop/gdk/immersions>>. [Accessed 27 April 2014].
- GOOGLE INC., 2014d. The Glass Explorer Program. [online] Available at: <<http://www.google.com/glass/start/how-to-get-one/>>. [Accessed 01 April 2014].
- GOOGLE INC., 2014e. ADK Camera API. [online] Available at: <<http://developer.android.com/reference/android/hardware/Camera.html>>. [Accessed 23 April 2014].
- GOOGLE INC., 2014f. ADK AudioRecord API. [online] Available at: <<http://developer.android.com/reference/android/media/AudioRecord.html>>. [Accessed 23 April 2014].
- GOOGLE INC., 2013a. Glass Development Kit. [online] Available at: <<https://developers.google.com/glass/develop/gdk/index>>. [Accessed 18 April 2014].
- GOOGLE INC., 2013b. Location and Sensors. [online] Available at: <<https://developers.google.com/glass/develop/gdk/location-sensors>>. [Accessed 18 April 2014].
- GOSSMAN, J., 2005. Introduction to Model/View/ViewModel pattern for building WPF apps. [online] Available at: <<http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx>>. [Accessed 29 April 2014].
- HABERFELLNER, R., DE WECK, O.L., FRICKE, E. and VÖSSNER, S., 2012. *Systems Engineering: Grundlagen und Anwendung*. 12th ed. Orell Füssli.

- HALAMKA, J., 2014. *Wearable Computing at BIDMC*. [online] Available at: <[http://geekdoctor.blogspot.co.at/2014/03/wearable-computing-at-bidmc\\_12.html](http://geekdoctor.blogspot.co.at/2014/03/wearable-computing-at-bidmc_12.html)>. [Accessed 22 April 2014].
- HEALTH LEVEL SEVEN INTERNATIONAL, 2013. *Health Level Seven International*. [online] Available at: <<http://www.hl7.org/>> [Accessed 04 May 2014]
- IEEE STANDARDS BOARD, 1990. *IEEE Standard Glossary of Software Engineering Terminology*. Institute of Electrical and Electronics Engineers.
- IREB E.V., n.d.. *Mission*. [online] Available at: <<http://www.ireb.org/mission.html>> [Accessed 12 May 2014].
- KANO, N., SERAKU, N., TAKAHASHI, F. and TSUJI, S., 1984. Attractive Quality and Must-Be Quality. *Journal of the Japanese Society for Quality Control*, **14**(2), pp. 39-44.
- KRANKENHAUS DER ELISABETHINEN GMBH, 2013. *Videoassistierte Simulation von Notfällen im OP*. [video online] Available at: <<http://www.elisabethinen.at/ger/Filme/Filmbeitrag-SALUS-Preisverleihung-2013>> [Accessed 04 May 2014].
- LAN, C. and RAMESH, B., 2008. Agile Requirements Engineering Practices: An Empirical Study. *IEEE Software*, **25**(1), pp. 60-67.
- LAN/MAN STANDARDS COMMITTEE OF THE IEEE COMPUTER SOCIETY, 2003. *IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band*. Institute of Electrical and Electronics Engineers.
- LAN/MAN STANDARDS COMMITTEE OF THE IEEE COMPUTER SOCIETY, 2000. *Supplement to IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band*. Institute of Electrical and Electronics Engineers.
- MICROSOFT CORPORATION, 2013. *Work in sprints*. [online] Available at: <<http://msdn.microsoft.com/en-us/library/vstudio/ee191595.aspx>> [Accessed 28 April 2014].
- MICROSOFT CORPORATION, 2007. *Microsoft Public License (Ms-PL)*. [online] Available at: <<http://www.microsoft.com/en-us/openness/licenses.aspx>> [Accessed 27 April 2014].
- NICELY, T.R., 1994. *Bug in the Pentium FPU*. [online] Available at: <<http://www.trnicely.net/pentbug/bugmail1.html>> [Accessed 04 May 2014].
- PAETSCH, F., EBERLEIN, A. and MAURER, F., 2003. Requirements engineering and agile software development, *Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises 2003*, pp. 308-313.

- PARTSCH, H.A., 2010. *Requirements-Engineering Systematisch*. Springer Berlin Heidelberg.
- POHL, K. and RUPP, C., 2011. *Requirements Engineering Fundamentals*. Rocky Nook.
- RAFFLE, H.S., WONG, A. and GEISS, R., GOOGLE INC. 2012. *Unlocking a screen using eye tracking information*. U.S. Pat. 8,235,529.
- RALPH, P., 2013. The illusion of requirements in software development. *Requirements Engineering*, **18**(3), pp. 293-296.
- RASIEL, E.M., 1999. *The McKinsey Way*. McGraw-Hill.
- ROBERTSON, S. and ROBERTSON, J., 2006. *Mastering the Requirements Process*. 2nd ed. Addison-Wesley Professional.
- RUPP, C. and SOPHISTEN, 2009. *Requirements-Engineering und -Management: Professionelle, Iterative Anforderungsanalyse für die Praxis*. 5th ed. Carl Hanser Verlag München Wien.
- SACHS, M. 2014a. *Laparoscopic Cholecystectomy*. [photograph] (Internal Collection)
- SACHS, M. 2014b. *Laparoscopic Cholecystectomy, Room*. [photograph] (Internal Collection)
- SCHWABER, K. and SUTHERLAND, J., 2013. *The Scrum Guide*. [pdf] Available at: <https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf> [Accessed 04 May 2014].
- SCHWABER, K., 1997. SCRUM Development Process. In: J. SUTHERLAND, C. CASANAVE, J. MILLER, P. PATEL and G. HOLLOWELL, eds. 1997. *Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings*: Springer London, pp. 117-134.
- STARNER, T., 2013. Project Glass: An Extension of the Self. *IEEE Pervasive Computing*, **12**(2), pp. 14-16.
- STECKLEIN, J.M., DABNEY, J., DICK, B., HASKINS, B., LOVELL, R. and MORONEY, G., 2004. *Error Cost Escalation Through the Project Life Cycle*. [pdf] NASA Technical Reports Server (NTRS). Available at: <http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100036670.pdf> [Accessed 07 May 2014].
- STUDIOKONZEPT MEDIEN-TECHNIK GMBH, 2012. *SIMStation Product Sheet*. [pdf] Available at: <http://www.simstation.eu/media/SIMStation.pdf> [Accessed 23 April 2014].
- THOMAS, B.H. and SANDOR, C., 2009. What Wearable Augmented Reality Can Do for You. *IEEE Pervasive Computing*, **8**(2), pp. 8-11.
- TORBORG, S. and SIMPSON, S., 2014. *What's Inside Google Glass?*. [online] Available at: <http://www.catwig.com/google-glass-teardown/> [Accessed 28 April 2014].

VALLURUPALLI, S., PAYDAK, H., AGARWAL, S.K., AGRAWAL, M. and ASSAD-KOTTNER, C., 2013. Wearable technology to improve education and patient outcomes in a cardiology fellowship program - a feasibility study. *Health and Technology*, **3**(4), pp. 267-270.

VERSIONONE INC., 2014. State of Agile Survey.

VORRABER, W. and NEUBACHER, D., 2014. *Besprechungsprotokoll: Abstimmungsbesprechung User Group (Internal Report)*.

VORRABER, W., VÖSSNER, S., STARK, G., NEUBACHER, D., DEMELLO, S. and BAIR, A., 2014. Towards human-centered medical information services using near-eye display devices such as Google Glass. Submitted to: *Journal of Medical Systems*.

WELDI, M., 2014. *Surgery*. [photograph] Internal Collection.

WALDMANN, B., 2011. There's never enough time: Doing requirements under resource constraints, and what requirements engineering can learn from agile development, *19th IEEE International Requirements Engineering Conference 2011*, pp. 301-305.

VÖSSNER, S., 2014. *Percutaneous Transluminal Angioplasty*. [photograph] Internal Collection.

ZIMMERMANN, H., 1980. OSI Reference Model-The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, **28**(4), pp. 425-432.



## 9 LIST OF FIGURES

---

Figure 1: The relative cost to fix an error during different project development phases (figure modified from (Boehm 1981)).	2
Figure 2: Google Glass Explorer Edition. The device which serves as the mobile platform for development during this thesis.	3
Figure 3: The three fundamental parts of this thesis, along with a more detailed sub-division into five essential sections.	5
Figure 4: Requirements Engineering publications per year (figure modified from (Partsch 2010) which has been built utilizing the data provided by (Davis 2010)).	9
Figure 5: KANO-Model (figure modified from (Pohl, Rupp 2011)).	11
Figure 6: System, Context and Environment (figure modified from (Pohl, Rupp 2011)).	14
Figure 7: Typical MECE-Tree. The term at the top is split into an exhaustive list of non-overlapping sub-elements.	21
Figure 8: Requirements formula used in this thesis (figure modified from (Rupp, SOPHISTen 2009)).	22
Figure 9: Typical sequence diagram as used for Requirements Engineering by Partsch (2010).	22
Figure 10: The inter-relationship between the elements of XP (figure modified from (Beck 1999)).	31
Figure 11: Scrum (and its variants) are the most adopted agile development approaches according to the State of Agile Survey (figure modified from (VersionOne Inc. 2014)).	32
Figure 12: Exemplary Burndown Chart - a central SCRUM tool that appears in many tools which support Scrum (e.g. Visual Studio 2013 (Microsoft Corporation 2013)).	34
Figure 13: The five Use Case Classes along with their real-world counterparts this thesis focuses on.	39
Figure 14: Percutaneous Transluminal Angioplasty performed with Google Glass. The surgeon receives real time patient data onto his Near-Eye Display Device (Vorraber, Vössner et al. 2014).	42
Figure 15: Percutaneous Transluminal Angioplasty AS-IS-Process Analysis (BPMN excerpt).	44
Figure 16: BPMN Color Legend.	44

Figure 17: Through the introduction of Near-Eye Display Devices, the interventionist cannot lose line of sight to the patient's vital signs anymore (right).....	46
Figure 18: Surgeon and assisting personnel focusing on the endoscopic image during a Laparoscopic Cholecystectomy (Sachs 2014a).....	47
Figure 19: Schematic overview of the operating room during a Laparoscopic Cholecystectomy. ....	48
Figure 20: The screen positioning in the operating room during a Laparoscopic Cholecystectomy at the Hospital of Elisabethinen (Sachs 2014b).....	48
Figure 21: Laparoscopic Cholecystectomy AS-IS Process Analysis (BPMN excerpt).....	49
Figure 22: The introduction of a Near-Eye Display Device to the process of a Laparoscopic Cholecystectomy reduces the number of necessary shifts with regard to the surgeon's point of view. ....	51
Figure 23: Open surgeries, especially with minimal incisions are often too narrow to be clearly visible (especially) to the assisting personnel. (Weldi 2014) .....	52
Figure 24: Sketch of the operating room during an Open Cholecystectomy. ....	53
Figure 25: Open Cholecystectomy AS-IS-Process Analysis (BPMN excerpt).....	54
Figure 26: The assisting personnel can improve their assistance by utilizing the unique point of view of the surgeon. This allows them to avoid having to rely solely on their limited perspective of the operating field.....	56
Figure 27: Current recording setups often include only stationary cameras (Krankenhaus der Elisabethinen GmbH 2013).....	57
Figure 28: Surgery Simulation Training AS-IS-Process Analysis (BPMN excerpt).....	59
Figure 29: Open Cholecystectomy with Consultation AS-IS Process Analysis (BPMN excerpt) .....	62
Figure 30: Exploded Google Glass Explorer Edition, CC BY-NC-SA 3.0 (Torborg, Simpson 2014). Feature outlines have been added. ....	73
Figure 31: Criteria Satisfaction Legend. ....	77
Figure 32: Schematic overview over the (minimal) overall System.....	79
Figure 33: Schematic overview over the elements of System, Context and Environment, based on the classification scheme by Rupp et al. (2009).....	80
Figure 34: Schematic overview over the relation between View, Model and ViewModel in the MVVM-Pattern as it is described by Gossman (2005). ....	83

Figure 35: Schematic overview over the Glass Messaging Protocol connection establishment. .....	87
Figure 36: Contents of the UDP-Beacon Packet.....	88
Figure 37: Schematic representation of the messages in the Glass Messaging Protocol. ....	89
Figure 38: Screenshot of the Glass Server Proof-of-Concept. ....	94
Figure 39: An UML Class Diagram providing an overview over the TCP / UDP-Server-Classes and the Messaging architecture.....	95
Figure 40: "Expert"-Settings for video transmission.....	97
Figure 41: "Expert"-Settings for picture transmission. ....	97
Figure 42: "Expert"-Settings for inbound video and photo transmission.....	98
Figure 43: Percutaneous Transluminal Angioplasty. The interventionist faces situations where he has no direct view to the patient or his vital signs. Near-Eye Display Devices allow for continous monitoring. (Vössner 2014) Circles have been added. ....	100

## 10 LIST OF TABLES

---

Table 1: Use Case classification scheme proposed by Vorraber et al. (2014). .....	40
Table 2: Classification of the Percutaneous Transluminal Angioplasty use case using the classification scheme proposed by Vorraber et al. (2014).....	43
Table 3: Specific Percutaneous Transluminal Angioplasty use case, using the scheme proposed by Rupp et al. (2009). .....	45
Table 4: Classification of the Laparoscopic Cholecystectomy use case using the classification scheme proposed by Vorraber et al. (2014). .....	48
Table 5: Specific Laparoscopic Cholecystectomy use case, using the scheme proposed by Rupp et al. (2009). .....	50
Table 6: Classification of the Open Cholecystectomy use case using the classification scheme proposed by Vorraber et al. (2014).....	53
Table 7: Specific Open Cholecystectomy use case, using the scheme proposed by Rupp et al. (2009). .....	55
Table 8: Classification of the Surgery Simulation Training use case using the classification scheme proposed by Vorraber et al. (2014). .....	58
Table 9: Specific Surgery Simulation Training use case, using the scheme proposed by Rupp et al. (2009).....	60
Table 10: Classification of the Open Cholecystectomy with Consultation use case using the classification scheme proposed by Vorraber et al. (2014).....	61
Table 11: Specific Virtual Consultations use case, using the scheme proposed by Rupp et al. (2009). .....	64
Table 12: Definitions of the terminology used in sections 3.9.3 - 3.9.5. ....	68
Table 13: Google Glass Explorer Edition Features, classified.....	73
Table 14: Comparison of different system architecture alternatives.....	77
Table 15: Possible handshake concepts along with their advantages and disadvantages.....	86
Table 16: The different fields in a GMP-message and their content. ....	88
Table 17: The different message types and their payload.....	90

# 11 APPENDIX

---

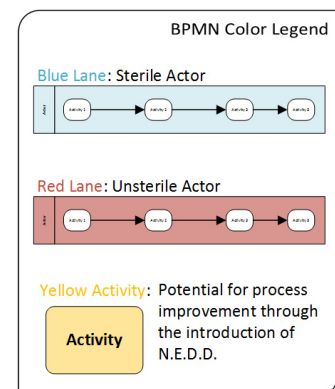
## 11.1 BPMN DIAGRAMS

In this section a legend for the BPMN diagrams used in the Master's Thesis will be provided. In addition to this, the full diagrams to the excerpts used throughout the document will be included.

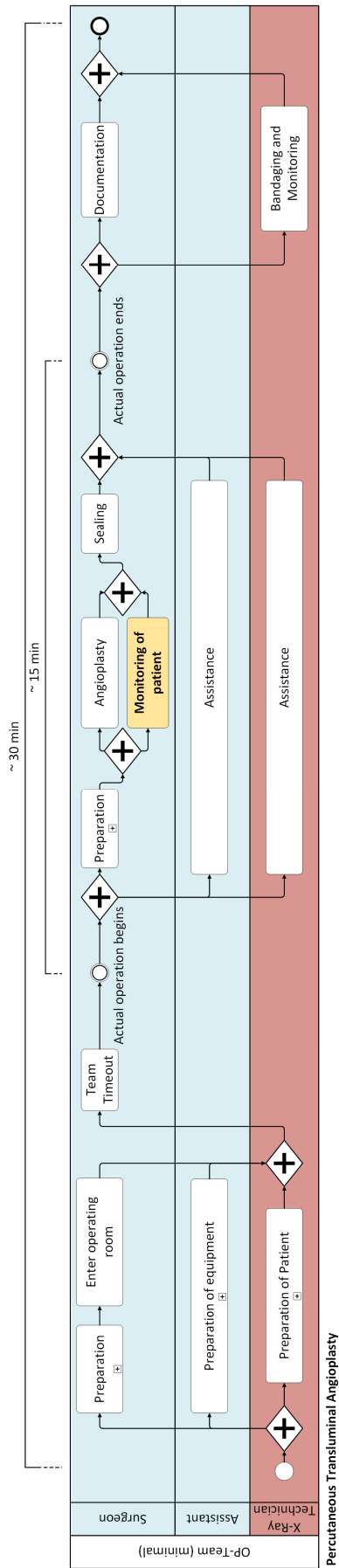
### 11.1.1 BPMN Legend

For the BPMN diagrams, the standard elements have been used. However, in order to convey additional information, specific colors have been used. These will now be explained:

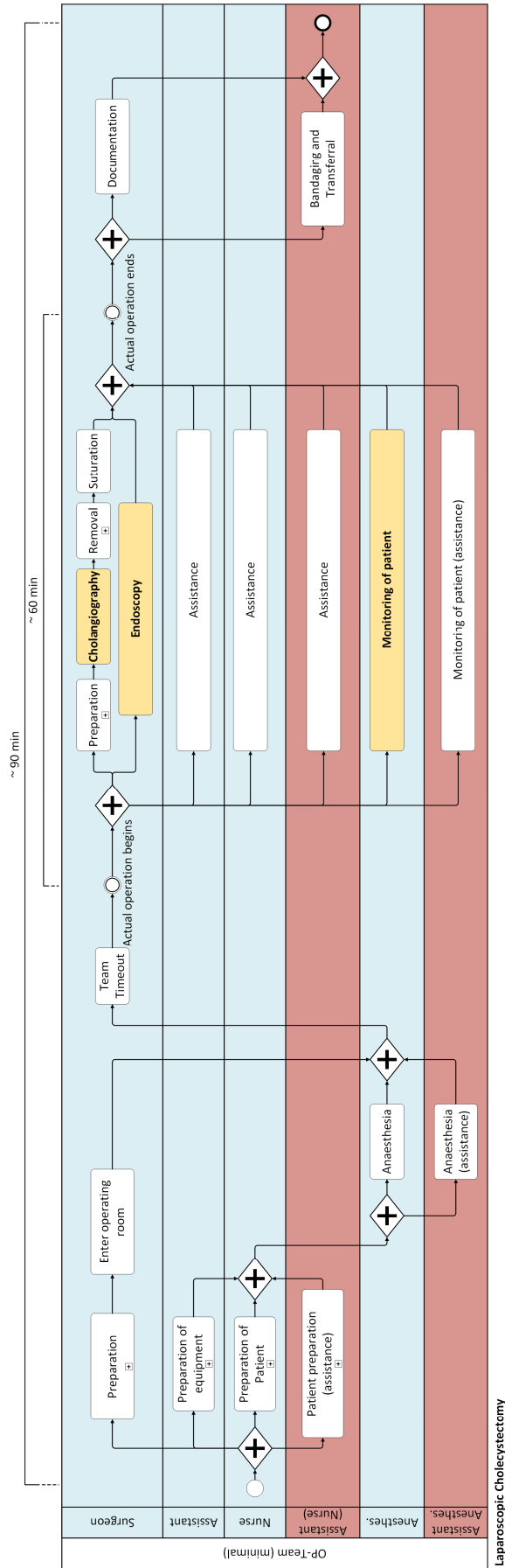
- Swim Lanes occur in two distinct colors:
  - Red indicates unsterile actors
  - Blue indicates sterile actors
- Actions may occur in yellow, indicating potential for process improvement through the introduction of Near-Eye Display Devices.



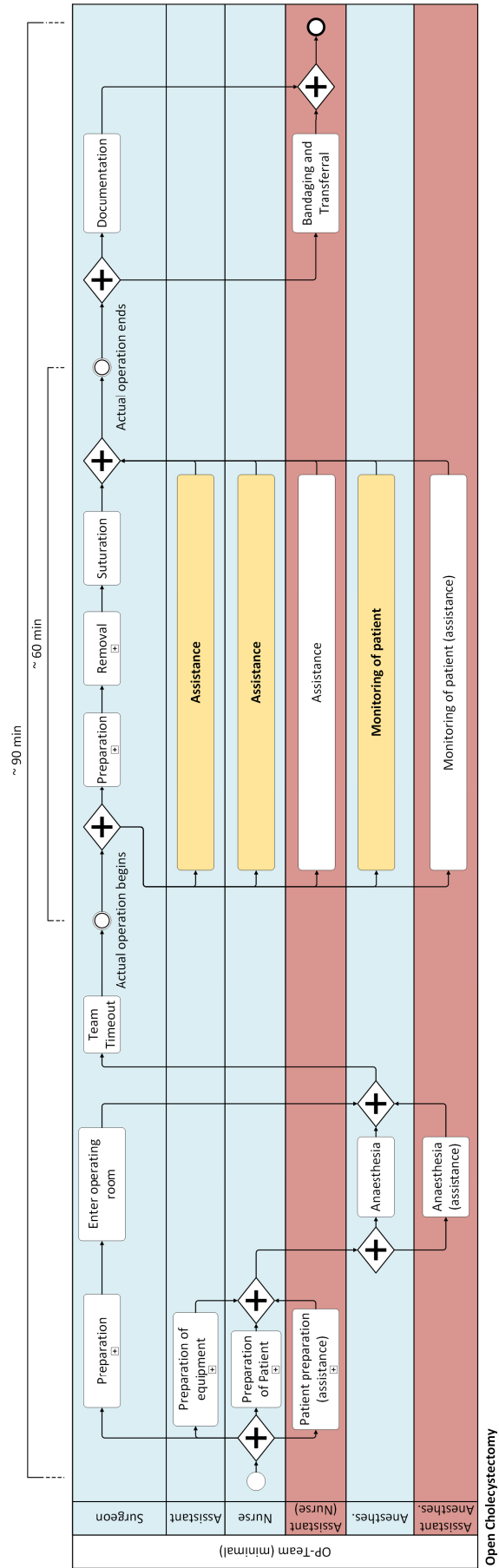
### 11.1.2 Percutaneous Transluminal Angioplasty



### 11.1.3 Laparoscopic Cholecystectomy

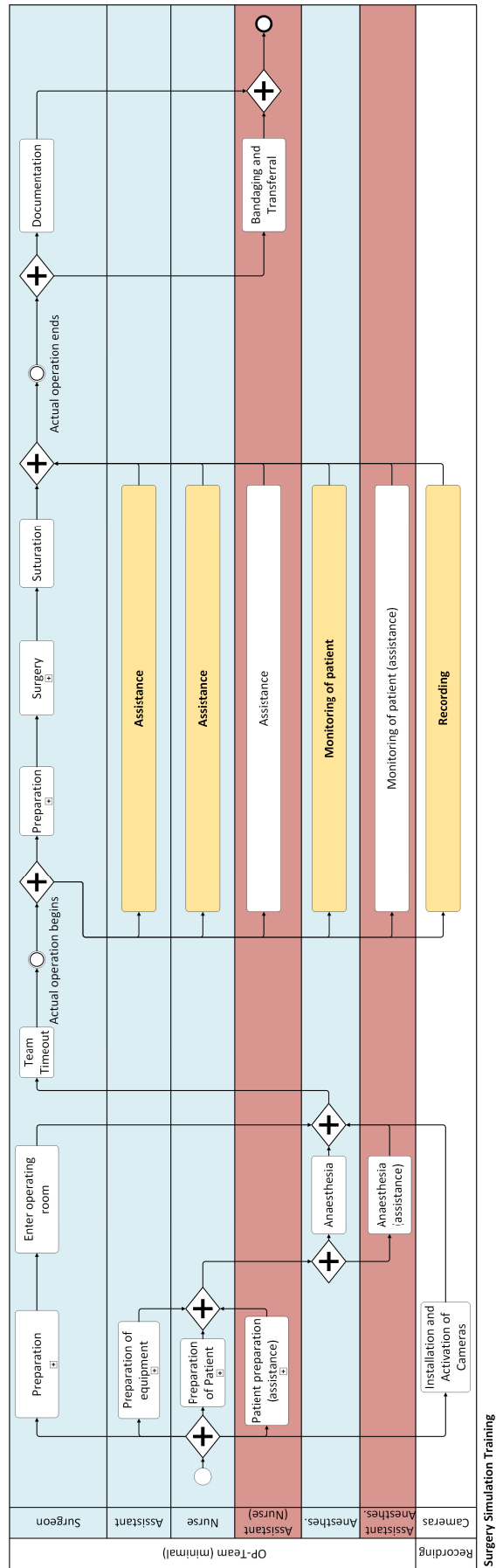


### 11.1.4 Open Cholecystectomy





### 11.1.5 Surgery Simulation Training



### 11.1.6 Open Cholecystectomy with Consultation

