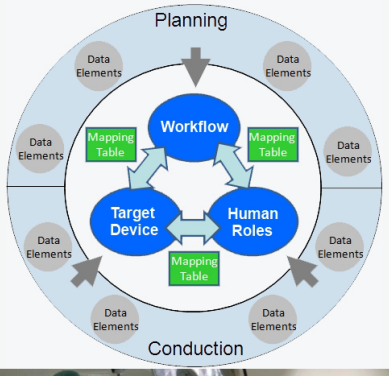
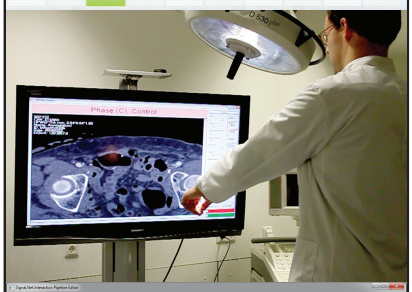
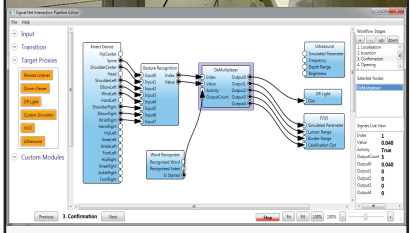
Computer Aided Medical Procedures
(CAMP)
Prof. Dr. Nassir Navab







Dissertation

Operating Room Specific Domain Model for Usability Evaluations and HCI Design

Ali Bigdelou

Fakultät für Informatik
Technische Universität München

TECHNISCHE UNIVERSITÄT MÜNCHEN

Chair for Computer-Aided Medical Procedures & Augmented Reality

Operating Room Specific Domain Model for Usability Evaluations and HCI Design

Ali Bigdelou

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. F. Matthes

Prüfer der Dissertation:

1. Univ.-Prof. Dr. N. Navab
2. Prof. P. Jannin, Ph.D.
   Universite de Rennes 1 / Frankreich

Die Dissertation wurde am 11.05.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 10.11.2012 angenommen.

Abstract

Advanced computerized solutions play a vital role in modern Operating Rooms (OR), however, they can also be challenging for the OR crew while operating them. It has been shown that such a lack of usability adversely affects patient safety and patient outcomes and can be rooted in the complex nature of the OR domain. The OR is a collaborative environment, where multiple users interact with distributed platforms of intra-operative devices, following a strict set of regulations defined as a surgical workflow. Moreover, sterility regulations limit the usage of classical touch-based interfaces. Creation and evaluation of computerized solutions for such an environment is not an easy endeavor. As a solution, in this work a specific domain model has been proposed. This model consists of surgical workflows, human roles and intra-operative devices, where their correlations are preserved in mapping tables. In next steps, this conceptual model has been exploited in development of two supportive frameworks specific to OR domain:

The first framework is OR-Use, which is developed for conducting multicenter usability evaluations for intra-operative devices. This framework facilitates the acquisition and management of usability data as required in different phases of usability testing such as planning and conducting. For the analysis phase, additionally a data visualization technique has been introduced, using the OR domain model and an off-the-shelf visualization toolkit.

The second framework is Signal.NET, a flexible framework that enables to define customizable, context-aware and human-centered interfaces for intra-operative solutions. This framework fuses the interfaces of all target systems in a unified interface, based on the modeled requirements of a given surgery. The behavior of such an interface can be adapted, utilizing a specialized visual editor. Furthermore, several categorical and spatio-temporal gesture recognition methods using different types of input modalities e.g. Kinect, Inertial Sensor, and machine learning techniques e.g. PCA, Manifold have been developed within the proposed framework.

Additionally, two methodological approaches are introduced to bring the proposed frameworks into practice. All the introduced concepts in this work have been investigated in several quantitative and qualitative studies, which approved their applicability in practice. These frameworks can be considered as the first in their type specifically designed and developed for the OR domain and can be a basis for many other multi-disciplinary collaborations and studies in the future.

Keywords: Operating Room, Usability Evaluations, User Interface, Human-Computer Interactions, Gesture.

Zusammenfassung

Fortgeschrittene, rechnergestützte Systeme spielen eine entscheidende Rolle in modernen Operationssälen (OP), stellen aber auch eine Herausforderung für das OP Personal dar, das sie benutzen muss. Es ist erwiesen, dass solch eine mangelhafte Benutzerfreundlichkeit, die sich durch das komplexe Umfeld der OP-Domäne begründet, die Patientensicherheit und das Behandlungsergebnis negativ beeinflussen. Der OP ist ein multidisziplinäres Umfeld, in dem viele Beteiligte mit unterschiedlichsten intra-operativen Systemen interagieren. Dabei folgen sie einem strengen Ablauf an Behandlungsschritten, die in einem chirurgischen Workflow definiert sind. Das sterile Umfeld und die damit verbundenen Auflagen schränken den Einsatz von klassischen, berührungsempfindlichen Eingabegeräten zusätzlich ein. Die Entwicklung und Evaluation von rechnergestützten Lösungen für solch eine Umgebung ist ein anspruchsvolles Vorhaben. Als Lösung wurde in dieser Arbeit ein spezifisches Domänenmodell vorgestellt. Dieses Modell besteht aus chirurgischen Workflows, menschlichen Faktoren und intra-operativen Werkzeugen, deren Wechselwirkungen in Zuordnungstabellen gespeichert werden. In den nächsten Schritten wurde dieses konzeptionelle Modell zur Entwicklung von zwei unterstützenden Frameworks für diese Domäne eingesetzt.

Das erste Framework nennt sich OR-Use und wurde entwickelt, um Multi-Center Auswertungen zur Benutzerfreundlichkeit von intra-operativen Werkzeugen durchzuführen. Dieses Framework erleichtert die Akquirierung und Verwaltung von Benutzerfreundlichkeits-Daten, welche in verschiedenen Phasen der Benutzerfreundlichkeits-Evaluation, wie der Planung und Ausführung, benötigt werden. Für die Analyse-Phase wurde zusätzlich eine Daten-Visualisierungstechnik eingeführt, welche auf dem OP-Domänenmodell und einem handelsüblichen Visualisierungs-Toolkit aufbaut.

Mit Signal.NET wird ein flexibles, zweites Framework vorgestellt, welches die Erstellung von anpassbaren, kontextabhängigen und auf den Menschen fokussierten Benutzeroberflächen für intra-operative Anwendungen ermöglicht. Dieses Framework vereint die Benutzeroberflächen von allen beteiligten Systemen in einer einheitlichen Benutzeroberfläche, basierend auf den modellierten Anforderungen einer gewünschten chirurgischen Anwendung. Das Verhalten einer solchen Benutzeroberfläche kann durch einen spezialisierten grafischen Editor angepasst werden. Im Rahmen des vorgestellten Frameworks wurden verschiedene strikte und räumlich-zeitliche Gestenerkennungsmethoden entwickelt, welche auf verschiedenen Arten von Eingabemodalitäten wie Kinect, Beschleunigungssensoren und Machine Learning Techniken (PCA, Manifold) basieren.

Zusätzlich werden zwei methodologische Ansätze eingeführt, um die vorgestellten Frameworks in der Praxis einzusetzen. Alle vorgestellten Konzepte in dieser Arbeit wurden in mehreren quantitativen und qualitativen Studien, welche deren praktischen Nutzen bestätigen, evaluiert. Diese Frameworks können als die ersten ihrer Art angesehen werden, welche explizit für den Einsatz in der OP-Domäne entwickelt wurden. Sie können als Basis für viele andere multidisziplinäre Kollaborationen und Studien in zukünftigen Arbeiten verwendet werden.

Keywords: Operationssaal, Nutzbarkeitsstudien, User Interface, Gesten.

Acknowledgments

I would like to express my most sincere and special thanks to my supervisor Prof. Dr. Nassir Navab, the head of chair for Computer Aided Medical Procedures at Technical University of Munich. He generously offered me such an interesting topic to work on, and patiently guided me through my research. Additionally, he provided me with the freedom to explore new directions in my research. This enabled me to expand my knowledge in several new directions such as usability evaluations and user interface design. My parents have always been my greatest motivation in my life. Always believing in me, they provided me with the best circumstances to go for whatever fascinated me most. I would like to express my deepest love and respect for them in my thesis acknowledgement as they truly have been the meaning of my life all over. I can never thank them enough for being such great and wonderful parents and friends to me. Mom, Dad, you are simply amazing.

Next, I would like to thank all colleagues and friends at the chair for Computer Aided Medical Procedures. My warmest thanks to Loren Schwarz, Asli Okur, Ralf Stauder, Arash Taki, Athanasios Karamalis, Alexander Hartl, Alexandru Duliu, Jose Gardiazabal, Victor Castaneda, Asad Safi, Anabel MartinGonzalez, Juergen Sotke, Tobias Sterner, Tobias Benz, Tobias Blum, Selen Atasoy, Stefan Holzer, Ahmad Ahmadi, Lejing Wang, Max Hoffmann, Vasileios Belagiannis, Tobias Lasser, Stefanie Demirci, Pascal Fallavollita, Maximilian Baust, Eric Soehngen, Selim Benhimane, Ali Kamen, Wolfgang Wein, Slobodan Ilic, Alexander Ladikos, Olivier Pauly, Razvan Ionasec, Thomas Wendler, Stefan Wiesner, Martin Horn and Martina Hilla. Without your helps and supports most of this work would not have been possible.

My very special thanks would go to Nasim Pour Aryan who has been supporting me during the hardest times for this work. Nasim you are wonderful. I would like to express my gratitude to Nastaran who never once stopped supporting me throughout my years living in Germany. Moreover, I would like to thank my dear friends who were next to me in Munich, Aida Emami, Siyavash Adnani, Mahdad Eghbali and Kuros Yalpani. Life is much harder when you are far away from family, however next to my best friends I truly had warmest support and times full of happiness.

Last but not least it would be a pleasure to thank Prof. Dr. Florian Matthes for his support in different stages of my thesis and his interest in this topic that made me even more motivated and eager to succeed. Additionally, special thanks to the defense committee, Dr. Pierre Jannin and Mrs. Aline Schmidt.

Contents

List of Figures

List of Tables

Part I.

Introduction and Backgrounds

CHAPTER 1

Introduction

RAPID advancements in technology have influenced most industry fields, and have led to spreading of complexity in many domains, including the Operating Room (OR). In the first section of this chapter, we explain some of the main problems raising up due to such complexity in the Operating Room for engineers and users. In the remaining sections, main research objectives of this study, our contributions as well as the structure of this dissertation are explained.

1.1. Problem Definition

In recent years, novel medical technologies have contributed to the improvement of modern health care. Novel technologies have enabled new methods to diagnose and treat patients, e.g. magnetic resonance imaging or minimally invasive surgery. Such approaches help to obtain precise information about a patient's health status but also open new therapeutic treatments.

Unfortunately, this progress also introduces additional complexity to health care and brings up new challenges for engineers and users of this domain. Based on [18], one of the main reasons for failure of treatments in the Operating Room is caused by human errors. As medical staff has to deal with an increasing number of complicated medical devices and equipments, the risk for errors rises. These errors can lead to so called 'adverse events', i.e. injuries caused to patients as the result of a medical intervention rather than the underlying medical condition [5]. According to an estimation published by an Institute of Medicine

released in 1999, between 44,000 to 98,000 people die each year from preventable medical errors in American hospitals. These figures exceed the annual number of fatalities caused by motor vehicle accidents, breast cancer, or AIDS.

Although many of these errors are not related to medical devices, some are directly or indirectly linked to their design and use [88]. Bleyer studied adverse events which occurred using medical devices. In 60% of the analyzed cases, use errors were identified as the cause of the incidents. This renders the wrong usage of technical equipment as the main reason for adverse events in medicine [15]. However, this is not solely attributable to the users. Frequently, users have to face poorly designed user interfaces (UI), which makes them insecure [46]. Schubert [122] presented similar results and reported that 80% of adverse events occurred while using medical devices are due to weaknesses of the user interface. Since improper and carelessly designed user interface can lead to an increased risk of patient treatment [18] and a reduced efficiency and quality of medical workflows, health care systems need to be designed with extra care. Basically, medical devices are only valuable for front-line clinicians, who make life-affecting decisions every day, if they make their job easier. Systems which increase their workload or the chance for error, will not be tolerated [143]. This is due to the fact that the success of any medical operation, besides surgeon experience, depends on the equipments used and data which is provided to the surgeon.

1.1.1. Usability Evaluations

Studying the usability of a system can improve its functionality and has become increasingly popular over the last decades [72, 108]. In many cases usability requirements are even turning to an official governmental constraint for the industries [72, 64]. The usability factor in medical domains such as the Operating Room plays a more critical role due to the fact that any mistake can lead to a serious problem.

Despite the fact that usability engineering techniques are successfully applied in many domains, creating systems with high usability can be a tremendous challenge in the OR due to the complexity of this domain [115]. When studying the usability of a system, it is crucial to have a complete understanding of the domain the system is used in. Yet in *complex domains*, this is not a simple endeavor [32]. In other words, to handle the complexity of the OR domain, usability specialists require deep domain knowledge, which they usually cannot acquire. Such lack of domain knowledge in complex domains impose challenges during all steps of conducting a usability test for usability specialists. Within the planning phase, usability experts report difficulties in knowing where and how to be-

gin, asking the right questions, developing realistic tasks, and understanding new domain-specific concepts [32]. While conducting the test, usability specialists struggle to interpret the relevance and significance of observed problems. Another problem is the huge amount of relevant usability data that has to be gathered and some how managed in such complex domains. Furthermore, compared to non-complex domains, the analysis of the collected usability data and reporting process require additional time and effort.

Considering the potential impact, which the usability of healthcare devices entails in regards to patient outcomes and the challenges posed to usability engineers when evaluating medical systems, the need for an approach to support their work becomes clear. Such supporting platform should facilitate the conducting, managing and analysis of usability data collected within multi-center studies.

1.1.2. Human Computer Interactions

As mentioned, various types of computerized medical systems are used in the OR, for instance to visualize volumetric patient data. However, when interacting with such devices during surgery, surgeons face challenges. Sterility regulations restrict the use of classical mouse-driven or touch-based user interfaces and control terminals are often required to be spatially separated from the main operating site [81]. Additionally, interventional workflow might not allow the surgeon to leave this site and the surgeon might only have very limited freedom of movement, while handling medical instrumentation. Typical solutions include delegating physical control over computerized systems to less-skilled assistants [50, 81]. This increases the amount of verbal communication, raising the chance of misunderstandings. The added level of indirection can also adversely affect precision and lead to inefficiency [81]. This problem can be targeted by replacing the traditional touch-based interfaces of intra-operative equipments with a gesture-based interface.

However, defining a gesture-based interface for the OR is not an easy endeavour. The OR is a collaborative environment and the interaction with several devices can be shared among multiple users based on the requirements of the surgical workflow. Additionally, user interaction requirements are highly dependent to the surgery type and surgeons' decisions. Lack of such prior knowledge challenges the development of predefined gesture-based interfaces. Therefore, a gesturing interface for the OR should be customizable, context aware and capable to fuse interfaces of different intra-operative devices. However, to best of our knowledge, there is no generic software infrastructure to provide this in the Operating Room. There are few medical solutions which combine interfaces of different hardware. However, they are specialized for a specific type of surgeries and (1) present

a small or hardly extensible number of intra-operative devices, (2) they are platform and technology dependent, or (3) they do not provide a flexible prototyping environment for a large and heterogeneous number of research products (such as new device prototypes, new algorithms, etc.). Most of the intra-operative solutions either exist as standalone software packages, holding a loose interaction with their environment, or have been built tightly into the target hardware devices, which obviously decreases the usability and customization possibilities for new scenarios. Furthermore, most of these solutions are providing their own interfaces to the surgeons, which increases the redundancy of the input and output interfaces and accordingly the complexity of the OR.

1.2. Research Objectives

The principal objective of this work is to propose a generic model for the OR domain and exploit it in the development of specialized solutions for supporting intra-operative usability evaluations and HCI design. The following list explains our objectives in more details:

- Defining a conceptual domain model for the OR which encapsulates the main complexity aspects influencing the usage of intra-operative medical solutions.

- Designing and developing a specified tool or framework for supporting the usability testing in the OR and overcoming challenges such as test conduction and data management.

- Defining a new approach and tool for analysing the usability data, collected during multi-center usability studies.

- Defining an applicable gesturing method considering the HCI limitations of the Operating Room.

- Designing and developing a tool or framework for integration and customization of the intra-operative interfaces, based on the requirements of a specific procedure.

- Evaluating the applicability of the proposed tools and concepts in context of synthetic and real Operating Rooms.

1.3. Contributions

The first contribution of this work is in the proposed domain model for the Operating Room. This model captures the main complexity aspects of the OR, e.g. surgical workflow, human roles and intra-operative devices. The correlations of these aspects are preserved within the mapping tables. As the next contribution, this model has been exploited in design and development of the OR-Use framework, a usability testing support tool for the OR. This framework enables usability testing in multicenter evaluations. The OR-Use also facilitates the acquisition and management of usability data as required in different stages of usability testing such as planning and conducting. A novel usability data visualization technique is also proposed and developed into the OR-Use framework to support analysis of large set of data collected within usability studies. The work related to the OR domain modeling and usability evaluations, presented in this thesis, spawned a series of publications presented at major conferences in the field of computer-aided intervention and usability engineering:

- **A. Bigdelou**, T. Sterner, S. Wiesner, T. Wendler, F. Matthes and N. Navab. OR Specific Domain Model for Usability Evaluations of Intra-Operative Systems. International conference of *Information Processing in Computer Assisted Interventions (IPCAI)*. Berlin (DE), June 22, 2011.

- **A. Bigdelou**, A. Okur, M.-E. Hoffmann, B. Azizi and N. Navab. Towards systematic usability evaluations for the OR: an introduction to OR-Use framework. International conference of *Information Processing in Computer Assisted Interventions (IPCAI)*. Pisa, Italy, June 27, 2012.

- **A. Bigdelou**, A. Katouzian and N. Navab. Model-based visualization of usability data to support analysis in complex domains. 21th International Conference of *Usability Professionals Association (UPA)*. Henderson, Nevada, USA, June 2012.

- A. Okur, A. Ahmadi, **A. Bigdelou**, T. Wendler and N. Navab. MR in OR: First analysis of AR/VR visualization in 100 intra-operative Freehand SPECT acquisitions. The 10th IEEE and ACM International Symposium on *Mixed and Augmented Reality (ISMAR)*. Basel, Switzerland, October, 2011.

Another contribution of this work is the proposed touch-free interface that gives surgeons direct control over computerized systems by means of gestures. This approach learns gestures from examples given by a surgeon who can freely define movements that

fit best a particular interventional scenario and personal habits. The proposed interaction techniques do not require the surgeons to touch anything, change their position or leave the patient while interacting with intra-operative tools. The proposed gesturing technique has been developed and tested utilizing different input modalities such as inertial sensors and Kinect. Furthermore, to successfully exploit the proposed gestureing technique into the OR, a specialized framework, Signal.NET, has been designed and implemented based on the key aspects of the OR domain model. The Signal.NET framework allows an easy assignment of learned gestures to functions of arbitrary medical devices. Moreover, the framework enables dynamically changing the assignment of gestures to target systems based on current workflow stages. In the collaborative OR setting, the proposed approach can also be used to distribute the control, based on different OR roles. A visual editing environments is also developed to facilitate the customization of the gesturing interface at runtime without further programming. Last but not least, several evaluations have been conducted to evaluate the applicability and acceptance of the proposed techniques in this work. The generality of our approaches allows an application in other domains with similar characteristics, however, the focus of this work is on the OR environment. The work related to the novel gesture recognition technique and HCI designing framework, presented in this thesis, spawned a series of publications presented at major conferences in the field of computer-aided intervention and user interface:

- **A. Bigdelou**, T. Benz, L. Schwarz and N. Navab. Simultaneous Categorical and Spatio-Temporal 3D Gestures Using Kinect. IEEE Symposium on *3D User Interfaces (3DUI)*. Orange County, CA, March 2012.

- **A. Bigdelou**, L. Schwarz, T. Benz and N. Navab. A Flexible Platform for Developing Context-Aware 3D Gesture-based Interfaces. International ACM Conference on *Intelligent User Interfaces (IUI)*. Lissabon, Portugal, February 2012.

- **A. Bigdelou**, L. Schwarz and N. Navab. An Adaptive Solution for Intra-operative Gesture-based Human-Machine Interaction. International ACM Conference on *Intelligent User Interfaces (IUI)*. Lissabon, Portugal, February 2012.

- **A. Bigdelou**, L. Schwarz and N. Navab. Learning Gestures for Customizable Human-Computer Interaction in the Operating Room. International Conference of *Medical Image Computing and Computer Assisted Intervention (MICCAI)*. Toronto, Canada, September 2011.

Several other studies have been conducted to gain the required understanding about the real-life medical applications in the operating rooms as listed bellow:

- S. Demirci, **A. Bigdelou**, L. Wang, C. Wachinger, M. Baust, R. Tibrewal, R. Ghotbi, H-H. Eckstein and N. Navab. 3D Stent Recovery from One X-ray Projection. 14th International Conference on *Medical Image Computing and Computer Assisted Intervention (MICCAI)*. Toronto (CA), September 18-22, 2011.
- **A. Bigdelou**, A. Ladikos and N. Navab. Incremental Visual Hull Reconstruction. *British Machine Vision Conference (BMVC)*. London (UK), September 7-10, 2009.
- A. Taki, **A. Bigdelou**, A. Roodaki, S.K. Setarehdan, G. Unal and N. Navab. A new tool for automated border detection and characterization of atherosclerotic plaque Composition in IVUS Images. International Conference on *Computers in Cardiology (CinC)*. Park City, Utah, September 13-16, 2009.

1.4. Overview of the Dissertation Organization

This thesis is organized in ten chapters. **Chapter 2** introduces our conceptual domain model of the Operating Room. **Chapter 3** introduces background knowledge regarding the general concepts of usability, usability in complex domains, and aspects of usability evaluations of medical devices. **Chapter 4** explains the architecture of the OR-Use framework. **Chapter 5** presents a novel usability data visualization technique, developed to assist the analysis of large set of usability data. **Chapter 6** reports about the conducted usability studies using OR-Use framework and several quantitative and qualitative experiments, performed to evaluate the OR-Use framework. **Chapter 7** provides some background and related works about gesture-based interaction and introduces a novel concept of categorical and spatio-temporal gesturing. It also contains details about our proposed gesturing technique. **Chapter 8** introduces the Signal.NET framework and its underlying architecture. **Chapter 9** shows the results of the conducted quantitative and qualitative experiments, to evaluate the proposed gesturing technique and the Signal.NET framework. Finally, in **Chapter 10**, we conclude and discuss about the future work and potential improvements of the presented methods.

CHAPTER 2

Operating Room Domain Model

In this chapter, we describe our approach for modeling complex domains from a generic perspective. Then, we introduce the key components of our approach, followed by an explanation on how to assemble them in a model. As the next step, we describe how our approach could be customized for the OR domain. This OR specific domain model could be incorporated into usability studies and can be a basis for designing context-aware interfaces for intra-operative systems.

2.1. Introduction

Although software systems are used widely in the Operating Rooms, not much work has been done for modeling and analyzing this domain from a software engineering point of view. Furthermore, available software frameworks are not defined and categorized based on a clear and uniform domain model. This issue causes many redundant implementations and a huge amount of extra work. As a solution to these problems, we present an Operating Room specific domain model. This model can be utilized to support usability study of intra-operative systems. Additionally, this model can be considered as a basis for customizing human computer interactions in this domain. The proposed model consists of various aspects of the Operating Room, which affect the domain complexity like workflow, human role and their interconnections.

2.2. Complex Domains

According to [133], a domain can be regarded as a sociotechnical system, if it is composed of several layers as illustrated in Figure 2.1. The core of this domain consists of a technical system, surrounded by the workers interacting with it. Furthermore, the system is influenced by the organizational or management structures, and its environmental context.

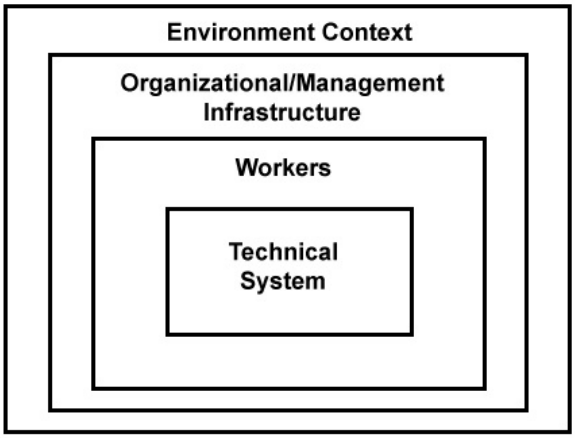


Figure 2.1.: Layers of a sociotechnical system, adapted from [133].

Considering this representation of a domain, it is obvious that a technical view alone is not sufficient in order to fully capture its complexity. Instead, the context surrounding the technical system needs to be considered, as well. Vicente presents several dimensions of sociotechnical systems, which contribute to the complexity [133]. Complex sociotechnical systems commonly rate highly on some of the following aspects:

- *Large problem space*: A complex system tends to consist of many different elements and forces. Consequently, the number of potentially relevant factors, which workers and designers need to deal with, is enormous.
- *Social*: Many people must work together to make the overall system function properly. Therefore, clear communication is required to effectively coordinate all actions.
- *Heterogeneous perspectives*: Workers in complex systems often come from various backgrounds and thus represent potentially conflicting values of a diverse set of disciplines.
- *Distributed*: The demands associated with social coordination can be complicated as

the involved people and parts of the technical system may be situated in physically different locations. In addition to this geographical separation, cultural aspects can further increase complexity.

- *Dynamic*: Usually, complex systems are dynamic and may have long time constants. It can even take days for the work domain to completely respond to an action from its workers.

- *Hazard*: There is a high degree of potential hazard in operating complex systems, as inappropriate human behavior or beliefs can have catastrophic consequences.

- *Coupling*: Complex sociotechnical systems tend to be composed of many subsystems that are highly coupled (i.e., interacting). Therefore, it is very difficult to control all effects of an action.

- *Automation*: Complex systems are often highly automated. In case of abnormal situations, which cannot be handled by automation, workers need to actively act as problem solvers.

- *Uncertainty*: The true state of the work domain is never known with perfect certainty, as there tends to be uncertainty in the data available to workers.

- *Mediated interaction*: As the goal-relevant properties of a complex system often cannot be observed directly by humans without aid, computer interfaces should provide workers with a mediating representation of the work domain. Hence, the everyday skills that people use to routinely explore the natural environment are insufficient to deal with the demands of mediated interaction.

- *Disturbances*: Workers have the responsibility for dealing with unanticipated events. Hence, they have to improvise and adapt to the contingencies of such an event rapidly, to maintain safety and productivity.

Chilana et al. argue, that workers in complex domains require deep subject-matter knowledge and experts insights [32]. Their work typically doesn't consist of well-structured tasks. Instead it involves open ended and unstructured problems [115], and comprises many unique situations [32]. Domain experts need to deal with complex information, which may be incomplete or unreliable. Their decision making can involve tedious data analysis, while time might be a critical factor, and wrong decisions can have catastrophic effects. Such complex domains often consist of tightly coupled subsystems, and require

the interaction of multiple users from various disciplines. Vicente argues that the presence of complexity is highly demanding to people who work in complex systems, and also for their designers [133]. Consequently, the evaluation and designing intuitive interactive solutions in such systems poses high demands to engineers. Typical examples of such domains include medicine, intelligence, and military [115].

2.3. Complexity in the Operating Room

An Operating Room is the unit of a hospital where surgical procedures are performed. Since an OR is a sterile environment, all personnel has to wear protective clothing, including caps, masks, gloves, shoes, and other coverings to avoid the spread of infections. Figure 2.2 shows a typical layout of an OR and the involved OR crew. The composition of an OR-team depends on the type of surgery, yet typical members are (1) the main surgeon, (2,3) assistant surgeons, (N) OR nurse, (C) a technical circulator and the (A) anesthetist. The colors indicate the level of sterility, blue means sterile and green non-sterile personnel and equipment. The patient bed is located in the center of the room. A separation sheet divides sterile and non-sterile area towards the head of the patient, which is located on the side of the anesthetist.

Apart from a range of surgical tools, contemporary equipment used within surgeries includes many complex systems, such as imaging devices, patient monitoring systems, tracking and navigation devices, and robotic solutions. The course of action during a surgery is determined by the surgical workflow of the performed procedure. In general, interventions are never performed in exactly the same way, as the experience of the surgeon and the background of the patient vary and complications may occur. Despite these differences, surgeries of one type generally share a common sequence of events. These typical steps represent the medical workflow of a procedure, which is called surgical workflow. The OR domain rates highly on several of the dimensions of a complex domain (see Section 2.2), including:

- *Social*: Many different people with distinct roles work together during a surgery, to fulfill the common goal of improving the patient's condition.

- *Heterogeneous perspectives*: Considering the users and developers of an imaging device, many different perspectives need to be understood. While surgeons look at the system from a medical point of view, developers may be physicists and computer scientists, all engaged in different perspectives.

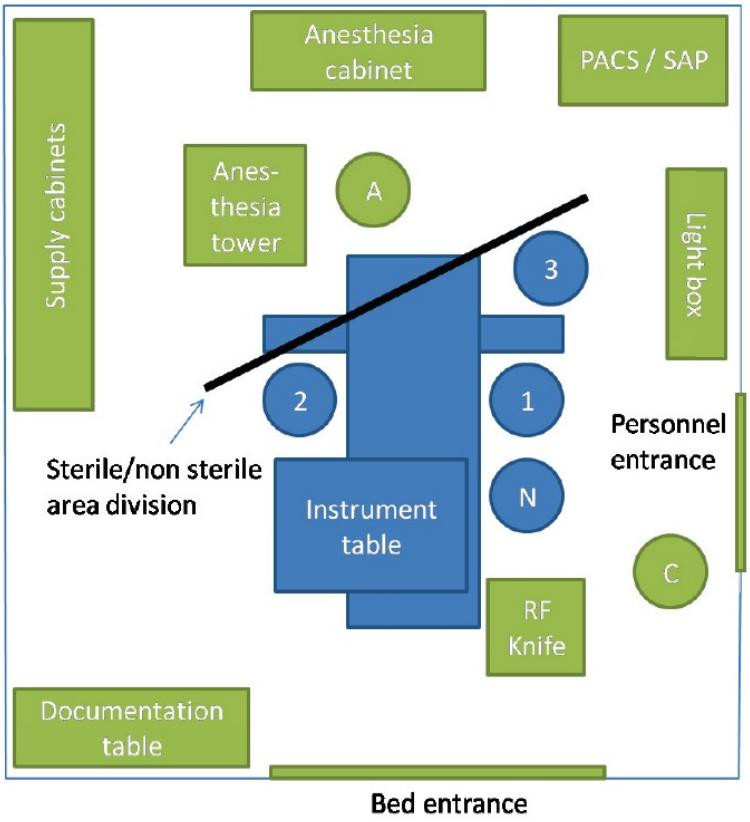


Figure 2.2.: A typical layout of an Operating Room [141].

- *Hazard*: As the life of a patient is at risk during surgery, the high degree of hazard in the OR domain becomes clear.

- *Mediated interaction*: OR-staff members cannot access vital data about the patient's condition directly. Instead, imaging devices or patient monitoring systems mediate this information. Hence, in addition to their medical skills, OR crew need to be able to utilize such complex technology.

- *Disturbances*: In case of complications, the OR-staff has to deal with unforeseen events, and therefore needs to response quickly, to maintain the patient's safety.

Bearing in mind the number of actors, the diversity of OR roles and the variety of equipment used during surgical interventions, the OR can clearly be regarded as a complex domain.

2.4. Domain Modeling Approach

A domain model, in the field of software engineering, can be thought of as a conceptual model of a family of systems which describes the various entities involved in those systems and their relationships [39]. The domain model is used in order to document the key concepts and identify the possible relationships among all major entities within the system like human roles, software and hardware resources and usually identifies their important methods and attributes. Having such a model and using various defined perspectives the problem solving process in that environment can be improved (speed, quality, flexibility).

As previously mentioned, the complexity of a domain may be rooted, among other things, in the interaction of multiple users and system components, and the presence of unforeseen events. We propose to tackle this complexity using the well-known divide and conquer principle. By decomposing the domain into its sources of complexity, and establishing a clear connection among the defined elements, a domain model could be created. In our approach, we encapsulate the complexity of a domain by a general approach, which can be specified individually depending on the domain of interest. Once such a model is defined for the target domain, it could be used in studies with similar scenarios, providing engineers with the necessary domain knowledge.

2.4.1. Main Components

Our conceptual representation of the complex domain consists of several key components. The first component is a *view*, which captures a single source of complexity. Next for each proposed *view* of the OR, we introduce a proper *view representation*. Then, based on these *view representations*, we define corresponding *view elements*. These elements embody aspects relevant to a given view, useful when focusing on the respective *view*. These components are explained in more detail in the following sections.

2.4.1.1. View

The first entity of our modeling approach is a view. It is used to capture a single source of complexity in the domain. Typically, one view captures features or functions of the system of interest, while another covers the involved stakeholders. Depending on the target domain, more specialized views are possible. For example, in the field of intelligence analysis, one view might focus on the various types of information sources, and a different view may encapsulate aspects regarding possible geographical distribution. When considering the domain of the Operating Room, one view could relate to the workflow of a

surgery, and a second one might consider the different roles of the OR-team. The number of views is not limited, and ultimately depends on how many origins of complexity can be identified for the target domain.

2.4.1.2. View Representation

Once a view has been established, one needs to decide upon data representation. Depending on the nature of the view, various ways to illustrate it may be available. For instance, the workflow of a surgical intervention can be represented using several approaches. One is to describe the logical relations of the workflow steps as a simple flow chart. Another approach is to classify entities horizontally and hierarchical decompose the respective workflow steps [106]. Furthermore, one could alter the level of granularity of the workflow steps, in both methods. Once the representation has been decided based on actual data from the domain, elements for the view can be derived from the view representation. These elements should embody relevant aspects required for engineers when focusing on the respective view. Looking at the example of surgical workflows, the view elements would consist of the identified workflow steps. Furthermore, information about possible dependencies between the elements can be encompassed by arranging them in a certain structure. The surgical workflow steps for example, could be arranged as a directed graph.

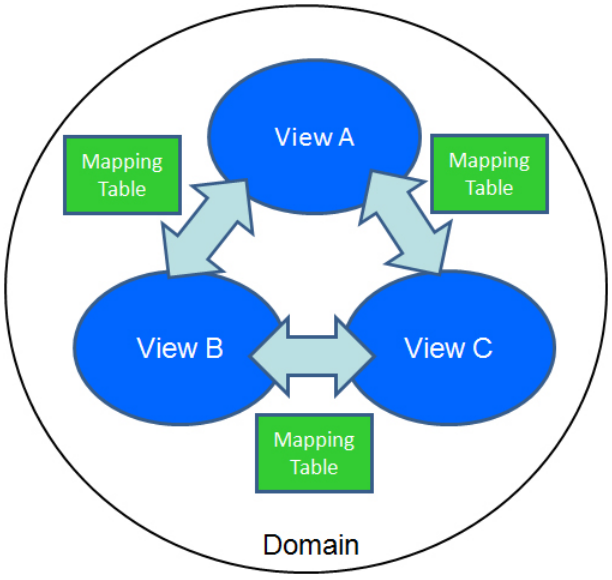


Figure 2.3.: A generic instance of a domain model.

2.4.1.3. Mapping Table

A mapping is defined as the correlation between two elements in two different views. The relationship of the elements can be specified through attributes, which may be defined based on the view context. As an example, when considering the two views actors and usecases, the participation of one actor in a particular usacase can be depicted by a simple true or false relation. A more sophisticated specification could break down this connection to responsible, accountable, consulted, and informed, similar to the well-known RACI matrix in the field of business analysis [67]. The set of possible mappings between the elements of two views are encapsulated within a mapping table, which forms the next building block of our modeling approach. As these tables abstract the complex connections between views, they provide the opportunity to easily move from one view to another, and hence allow engineers to evaluate and observe the domain, based on various perspectives and complexity aspects.

2.4.2. Domain Model Construction

In order to build the domain model, several steps need to be completed in close collaboration with domain experts, as they can provide detailed insights to understand the complexity of their field. First, the key sources of complications in the field have to be isolated with the help of domain specialists. This information can be used to define the views on the domain complexity. Next, one needs to find an effective method to represent each view, and the actual data for each view representation should be acquired in cooperation with the experts. Afterwards, one has to extract important elements from each representation, and decide upon their structure within the respective view. The last task is to select the views which will be connected by mapping tables, and define these mappings together with the domain professionals. Figure 2.3 shows a generic instance of the domain model, consisting of three different views connected by three mapping tables. In the following section we will construct an OR specific domain model, based on this approach.

2.5. Modeling the Operating Room

To create a proper domain model, we first carefully studied the OR in close collaboration with domain experts, i.e. surgeons and providers of the intra-operative devices. After analyzing the state of art in the modeling of the OR [109, 19, 106, 16, 77, 79, 78, 105, 117, 43], we identified three key components (*views*) of the OR domain, which can be

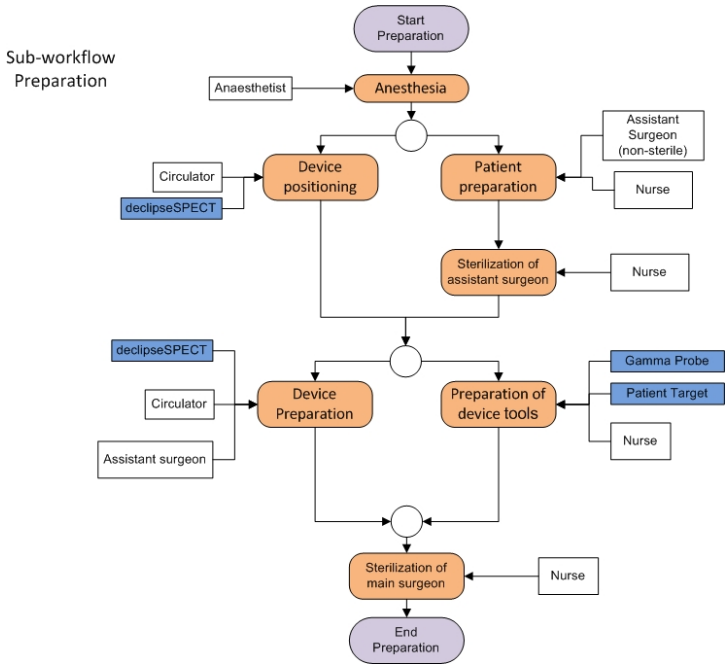


Figure 2.4.: Excerpt of a surgical workflow flow chart.

considered as the primary sources of complexity. These are *workflow*, *human role* and the *intra-operative device* itself.

2.5.1. Surgical Workflow

The *workflow* of an intervention encompasses the common steps of one type of surgery, determining the tasks of each involved role and the use of surgical devices. A *surgical workflow* may be captured manually during field observations and interviews with surgeons or with automatic OR workflow recovery methods [118, 16]. When developing an intra-operative device, it is crucial to consider that the system has to integrate smoothly into the current *workflow* of an intervention and support surgical activities. If a system does not adapt to the surgical routine, it may not be accepted by the surgeons. Furthermore, *surgical workflow* is affected by the characteristics of a device and may be altered in case a system introduces new treatment opportunities. For these reasons, we see *workflow* as one of the key aspects of intra-operative usability evaluations, and therefore include it as a *view* in our model of the OR domain. In order to represent this *view*, we model the *workflow* as a simple flow chart, describing the distinct steps of a surgery, as depicted in Figure 2.4. Moreover, we define the workflow stages as *view elements* within this *view*.

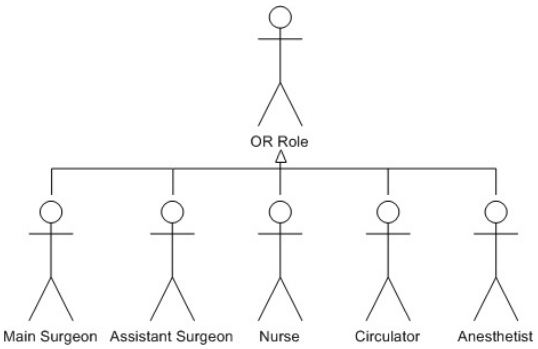


Figure 2.5.: Example representation of human roles involved in the OR.

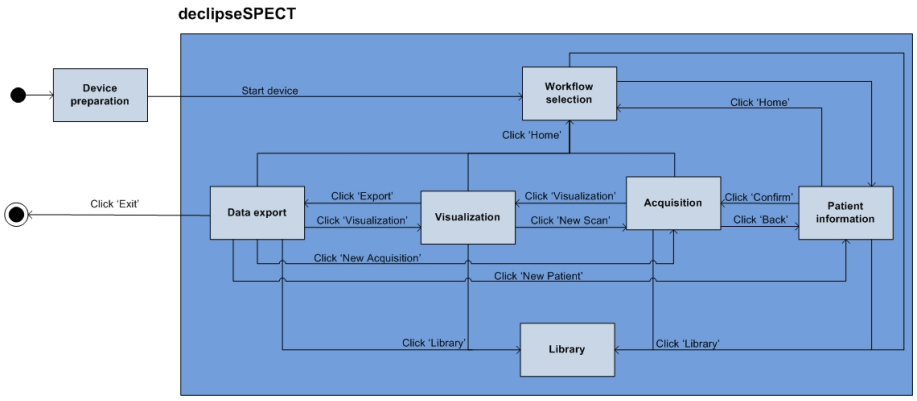


Figure 2.6.: Excerpt of a device state diagram.

2.5.2. Human Roles

As previously explained, members of the OR-team have distinct roles. Although these roles are conditional on the type of surgery, roles involved in many surgeries include a main surgeon, an assistant surgeon, a nurse, an anesthetist and a circulator. Intra-operative devices can be used by multiple users with different *roles*. Depending on the *human role*, different tasks may be performed with the system. As the background of each *human role* can vary, diverse types of information or different ways of visualization may be required. Obviously, this introduces a high degree of complexity, which is why we consider *human roles* as the second *view* in our model. Figure 2.5 shows a basic example how the roles involved in the OR can be represented. Each entity is a distinct *human role*, which has to be included as an element within the *human roles* view.

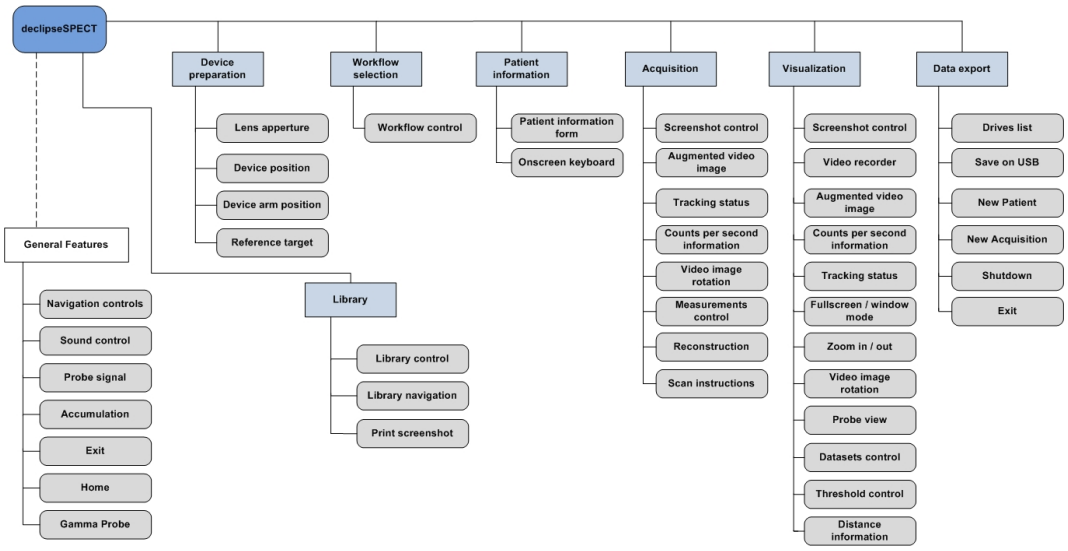


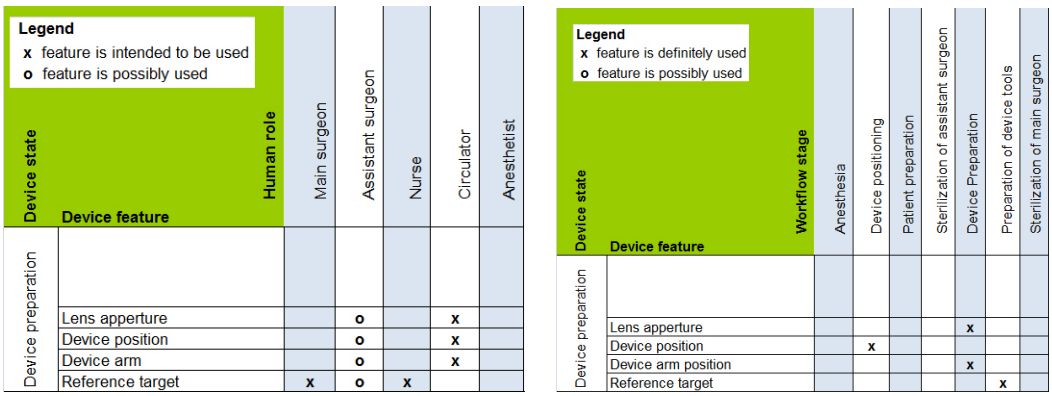
Figure 2.7.: A feature tree diagram, describing device states and related features.

2.5.3. Intra-operative Devices

With the recent technological advances, intra-operative devices themselves have turned into complex systems. They provide a number of advanced functionalities, facilitating surgical procedures. As the functions of such devices are tightly coupled with the current *workflow stage* and the *human roles* interacting with the system, the third *view* in our domain model covers the *target device* itself. We describe a surgical device by two entities, *features* and *states*. A *feature* is defined as a functionality or part of the system, which can be operated by the user. Depending on the objectives of the modeling, the granularity degree of *features* can vary from low level user interface elements (e.g. record button, cancel button, and progress bar), to high level abstractions, which summarize several components (e.g. record controls). *Features* are encompassed in logical units called *states*. For modeling the *states* of a device we use a state machine diagram, as illustrated in Figure 2.6. To represent the available *features* for each *state*, we used a simple feature tree as can be seen in Figure 2.7. An element in the *device view* consists of a pair *device feature* and corresponding *device state*.

2.5.4. Mapping Tables

To model the interconnections between various *views*, we further use *mapping tables* in our OR specific domain model. The relationship of the *elements* can be specified through



Legend
x feature is intended to be used
o feature is possibly used

Device state	Device feature	Main surgeon	Assistant surgeon	Nurse	Circulator	Anesthetist
Device preparation						
	Lens apperture		o		x	
	Device position		o		x	
	Device arm		o		x	
	Reference target	x	o	x		

(a)

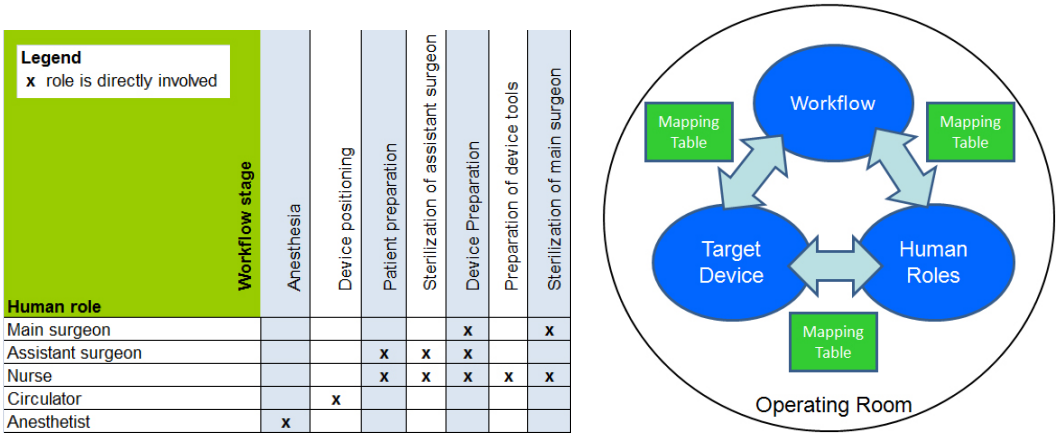
Legend
x feature is definitely used
o feature is possibly used

Device state	Device feature	Anesthesia	Device positioning	Patient preparation	Sterilization of assistant surgeon	Device Preparation	Preparation of device tools	Sterilization of main surgeon
Device preparation								
	Lens apperture					x		
	Device position		x					
	Device arm position					x		
	Reference target						x	

(b)

Figure 2.8.: Exemplary mapping tables between (a) the device view and the human roles view, (b) the device view and the workflow view.

attributes, which may be derived from the view context. For example, one can easily see in which workflow stages a certain device feature is used, or which roles utilize that particular part of the system. Figure 2.8.a, Figure 2.8.b and Figure 2.9.a show exemplary mapping tables between the different view elements.



Legend
x role is directly involved

Human role	Anesthesia	Device positioning	Patient preparation	Sterilization of assistant surgeon	Device Preparation	Preparation of device tools	Sterilization of main surgeon
Main surgeon					x		x
Assistant surgeon			x	x	x		
Nurse			x	x	x	x	x
Circulator		x					
Anesthetist	x						

(a)

(b)

Figure 2.9.: (a) Exemplary mapping table between the human roles view and the workflow view, (b) the proposed conceptual OR domain model.

2.6. Summery and Conclusion

In this chapter, we introduced the major sources of complexity in the OR domain, in order to construct a conceptual domain model for the OR. The first is surgical workflow which has to be followed for a particular intervention. The concept of workflow affects the mental model that OR-team members have of the surgical device, as they expect it to be congruent with the respective workflow steps. Next, are different OR roles which the OR-team members hold while interacting with the device. We explained that a surgical device may be operated by nurses and surgeons simultaneously, while both roles have differing demands to the system, caused by their varying requirements. Finally, the intra-operative devices are introduced which may be used for several distinct clinical applications, and can hold a variety of functions, while only a subset of them is used during one type of intervention. We further showed that how the relations among all three views can be captured using mapping tables to form a unified conceptual OR domain model as shown in Figure 2.9.b.

Part II.

Intra-operative Usability Evaluations

CHAPTER 3

Background and Related Work

THIS chapter lays the foundation for our work regarding to usability evaluations. We introduce general concepts in the field of usability, and explain the role of usability in complex domains. Additionally, we describe aspects regarding the usability evaluations of medical devices, and present the state of art in usability evaluations support tools.

3.1. Introduction

Computer-based systems in the Operating Rooms have become widely used as a means of improving the treatment process and the patient outcome. On the other hand, they can increase the risks for human error, see Chapter 1. This problem can be targeted by studying the usability of intra-operative devices. Thus, studying the usability of computerized solutions has become increasingly popular over the last decades. Usability can be defined by the ease with which a user can operate, prepare inputs for, and interpret outputs of a system or component. The aim of usability studies is to improve the user interface of a system regarding the usability components of interest. As the concept of usability is seen as a crucial factor for the commercial success of many products, it is addressed in a variety of industrial standards [2, 72].

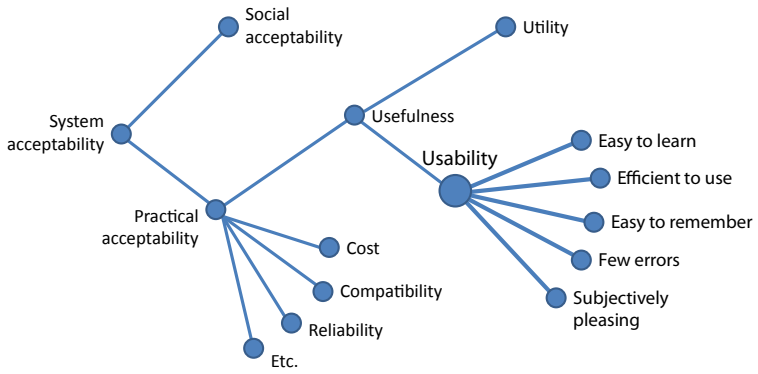


Figure 3.1.: Usability as part of system acceptability, introduced by Nielsen [108].

3.1.1. Usability

Several definitions of usability have been proposed in the literature [108, 2, 125]. An established definition of usability has been made by Nielsen in [108]. He embedded usability into the concept of system acceptability and confined it against related factors, as shown in Figure 3.1. Based on Nielsen, the overall acceptability of a system emerges from its social and practical acceptability [108]. The later component can be broken down into several categories, including usefulness, cost, compatibility and reliability. The usefulness of a system is accomplished once it can be used to achieve a desired goal. This category is divided into utility, determining whether the system provides the necessary functions to fulfill its purpose and usability. Nielsen describes usability as a set of five measurable components:

- *Learnability*: Systems should be easy to learn, so novice users can easily start getting the expected work done with the system.
- *Efficiency*: Systems should be efficient to use. After having learned the system, users should be able to achieve a high level of productivity.
- *Memorability*: Systems should be easy to remember to make it possible for casual users to return to the system after some period of not having used it, without the need to learn everything all over again.
- *Errors*: Systems should have a low error rate, which enables users to make few errors during the use of the system. In case of errors, the system should provide an easy way to recover. Catastrophic errors should not occur.

- *Satisfaction*: Systems should be pleasant to use, making users subjectively satisfied while using it.

Several other definitions for usability also have been proposed [2]. However, due to similarities, it is possible to translate other definitions to Nielsen's model. Liljegren [92] shows how Nielsen's five components can be mapped to the components of ISO 9421-11 [2]. For the remainder of this thesis, we only consider Nielsen's definition, as it is commonly accepted and in our opinion provides a clear and practicable view on usability.

3.1.2. Usability Engineering

Usability engineering is a methodological process to improve the user interface of the system under the test. Instead of determining the usability of system based on personal opinion, usability engineering measures usability attributes, and compares these measurements against predefined usability goals. The process is an integral part of the development cycle of a product, and is characterized by its iterative nature, the early involvement of users, and the usage of prototypes.

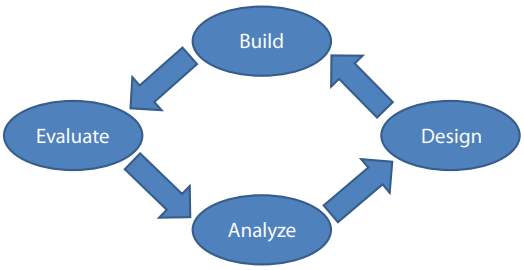


Figure 3.2.: The basic usability engineering life cycle, proposed by Butler [25].

Several usability engineering models have been presented in the literature. Although these approaches differ in detailed aspects, they essentially contain the same basic cycle of analysis, design, build and evaluation as shown in Figure 3.2. During the analysis stage, the focus lies on understanding the potential users of a system, and on developing a conceptual model of their work, i.e. analyzing the tasks they want to accomplish with the system. Furthermore, usability goals can be defined in terms of measurable criteria. These goals can be used in the evaluation phase to determine whether the iterative design is complete. In the design stage, the work is concentrated onto the user interface, in a manner that corresponds to the users' conceptual model. Within the build stage, prototypes of the user interface are constructed, which evolve to a fully developed system with each iteration

of the engineering cycle. At the end of each iteration, usability evaluation methods are applied to assess whether the design facilitates the users' tasks, or to determine how to improve the design itself.

3.2. Evaluation Methods

A variety of methodologies to evaluate the usability of a system's user interface have been proposed. This section will give an overview of existing evaluation methods, and describe a selection of methods in detail. In [73], Ivory et al. discuss the state automation in usability evaluation and therefore propose a taxonomy for categorizing usability evaluation methods along four dimensions:

- *Method Class*: a high level description of the type of evaluation, e.g. usability testing or inspection.
- *Method Type*: a description of how an evaluation is performed within a method class, e.g. thinking-aloud protocol within the usability testing class.
- *Automation Type*: a description of the evaluation steps that are automated, e.g. capture, analysis, or critique.
- *Effort Level*: a description of the type of effort required to execute the method, e.g. model development or interface usage.

In order to group existing usability evaluation methods, Ivory et al. use five different method classes of *Testing*, *Inspection*, *Inquiry*, *Analytical Modeling* and *Simulation*. Table 3.1 provides an overview of the evaluation methods which Ivory et al. have summarized in their work. The decision regarding which method should be used during a usability assessment ultimately depends on the goal of the assessment, and other factors, including the quantity of suitable test subjects (if required), time constraints, and availability of financial means. The remaining of this section describe main method classes as introduced by Ivory et al. [73].

3.2.1. Testing Methods

Usability testing is regarded as the most fundamental way to evaluate usability [108]. They refer to all evaluation methods which involve test users working with the target system within a controlled environment. Test subjects perform predefined tasks with a fully developed product or a prototype, while being observed by usability engineers. All these

Method Class Method Type	 Description
Testing	
Thinking-Aloud Protocol	user talks during test
Question-Asking Protocol	tester asks user questions
Shadowing Method	expert explains user action to tester
Coaching Method	user can ask an expert questions
Log File Analysis	tester analyses usage data
Codiscovery Learning	two users collaborate
Performance Measurement	tester records usage data during test
Inspection	
Cognitive Walkthrough	expert simulates users problem solving
Pluralistic Walkthrough	multiple people conduct cognitive walkthrough
Heuristic Evaluation	expert identifies violations of heuristics
Perspective-Based Inspection	expert conducts narrowly focused heuristic evaluation
Consistency Inspection	expert checks consistency across products
Standards Inspection	expert checks for standards compliance
Inquiry	
Field Observation	interviewer observes system use in users environment
Interviews	one user participates in a discussion session
Surveys	interviewer asks user specific questions
Questionnaire	user provides answers to specific questions
Screen Snapshots	user captures UI screens
Analytical Modeling	
Cognitive Task Analysis	predict usability problems
Knowledge Analysis	predict learnability
Design Analysis	assess design complexity
Programmable User Models	write program that acts like a user
Simulation	
Information Processing Modeling	mimic user interaction
Genetic Algorithm Modeling	mimic novice user interaction
Information Scent Modeling	mimic Web site navigation

Table 3.1.: Description of usability evaluation types [73].

method consists of four main stages as planning, conducting, analysis and report. To ensure the validity and reliability of this technique, selected test users need to be representative for the end users of the product. Furthermore, the given tasks need to reflect the real work, which will be done with the system. One advantage of usability tests is the fact that most problems can be found by only a small number of test users. Studies have shown that 8-10 subjects can identify up to 80% of the usability problems [108]. Another advantage is that testing methods capture the exact problems of the interface being assessed. Depending on the applied method, usability tests can yield qualitative data, or quantitative results. A

good way to document these results is by recording a video, as it allows for a subsequent analysis of the test session. Recordings typically include captures of the computer screen, or the interaction of users with a device from a wider perspective. However, one has to consider, that the time necessary to analyze a videotape can be between three and ten times the duration of the actual test [108]. As Table 3.1 shows, there are numerous methods to conduct a usability test. The following subsections will introduce the most common testing methods, including thinking-aloud, performance measurements, and log file analysis.

3.2.1.1. Thinking-aloud

Thinking-aloud is typically used for formative evaluations. The idea behind this method is that test subjects continuously verbalize their thoughts, while interacting with the system. This results in a considerable amount of qualitative data. The main advantages come from the communication with the users [108]. Evaluators get a good understanding, how potential users experience the system. The produced comments facilitate the detection of usability problems, and can be utilized to report such issues to the development team. However, thinking-aloud has some drawbacks as well [108]. Most people feel uncomfortable while executing this technique, and therefore interrupt their utterances regularly. Moreover, the need for constant verbalization can slow down test subjects, which renders performance measurements as less realistic. When reviewing user feedback, evaluators run the risk of overrating comments about the user interface. Hence, it is crucial to carefully interpret all observations.

3.2.1.2. Performance Measurement

Performance measurement is based on analysis of recorded usage data, while test subjects perform a given set of tasks. The method results in clearly defined performance metrics including, task completion time, number of performed tasks, success ratio, number of errors, recovery time from errors, detail statistics about usage of each feature. These metrics quantify how users perform certain tasks in a specific context and they provide an objective basis for determining the usability of a user interface [125]. They can be used for assessing whether certain usability goals is achieved, e.g. cross versions comparisons. The recommended number of users is ten, in order to receive reliable results. With the data collected with this method it is possible to compare and rank different features, however, it does not find individual usability problems [108]. Several video recording and event logging tools has been proposed to automate the capturing of usage data, thus facilitating the measurements [73].

3.2.1.3. Log File Analysis

Log file contains statistical information on the usage of a system [108]. Log file analysis is commonly applied to extract information about how a system is used in the field. Usually, the statistics show the frequency of usage for different features in the system, or other quantitative information similar to those collected for performance measurements (e.g. number of error messages, or usage time). This information is beneficial for finding seldom used features, or functions which frequently cause errors. In consequence, the system under the test could be improved, appropriately. Typically, the necessary data is captured from low-level parts of the system, like keyboard or mouse, or by logging the function calls within the software. This method facilitates the collection of large amounts of data from different users.

3.2.2. Inspection Methods

Usability inspection methods examine usability aspects of a user interface regarding its conformance to a set of predefined guidelines. Guidelines are a list of well-known design principles, which can vary from highly specific prescriptions to more general aspects. This is a heuristic evaluation method and its results rely solely on the judgement of the evaluators [73, 108]. Nielsen suggests to involve at least three to five evaluators, in order to find approximately 75% of overall usability problems. Heuristic evaluation and cognitive walkthrough are among the well-known method types in this class. A heuristic evaluation is usually performed with usability specialists where they evaluate the system according to heuristic principles. Nielsen [108] defines *heuristics* as rules that have to be considered when designing a user interface, including, feedback to user about the current system state, constructive and precise error messages and UI design consistency. The cognitive walkthrough aims at evaluating a system based on the usability attribute such as learnability, efficiency and effectiveness.

3.2.3. Inquiry Methods

Inquiry methods are based on direct feedback received from users and are often used to complement usability tests. However, the focus does not lie on studying specific tasks or measuring performance. These methods aim at high level subjective impressions, like preferences or opinions about various aspects of the UI [73]. Commonly applied inquiry methods include questionnaires and interviews. Questionnaires are useful to collect information about user subjective opinion from a large set of test subjects, as they can be filled

out without additional help from an interviewer [8, 42]. On the other hand, questionnaires require a considerable amount of preparation time. A key criteria for designing questionnaires is to standardize the pre-formulated answers. In contrast, in interviewing method questions can vary depending on the situation and allow direct contact with the users. The evaluator asks predominantly open ended questions to gather information from the user. Because of the high information density in user interviews, it would be possible to gather the required information with a smaller number of test subjects. Overall, compared to questionnaires, interviews are more flexible, but they require more time and organisational expenses [8].

3.2.4. Analytical Modeling Methods

Analytical modeling techniques supplement conventional evaluation methods such as usability testing [73]. Using some representation or model of the UI or the user, these techniques enable the prediction of usability. As Table 3.1 shows, a wide range of analytical modeling methods has been proposed. Most approaches are based on the model human processor (MHP) presented by Card et al. in [27]. Further details about these methods can be found in [80, 113, 73].

3.2.5. Simulation Methods

Similar to analytical modeling techniques, most simulation methods are based on the MHP, and supplement conventional usability evaluation [73]. Models of the user or the UI are utilized to simulate the interaction of the user with the interface and to generate reports about the interaction, e.g. similar to performance measurements. Simulations can be conducted with varying parameters to study various interface design tradeoffs, in order to facilitate design decisions. Moreover, simulations can be used to generate synthetic usage data for log file analysis techniques. Further explanations about simulation methods can be found in [73].

3.3. Usability Evaluations in Complex Domains

Although usability engineering techniques are successfully applied in many domains, creating systems with high usability can be a tremendous challenge in complex domains [115]. In complex domains, the interaction with the device is strongly influenced by its usage context. This makes the heuristic analysis process quite challenging. It is crucial to have a thorough understanding of the usage domain, while studying the usability of a

system. However, in complex domains, this is not a simple endeavor. In most cases, usability professionals are not able to fully acquire the required expertise, even after working in the same field for several years [32]. Considering this lack of domain knowledge, usability studies are aggravated by the domain-specific terminology, the presence of many unique situations, and the limited access to domain experts [32]. In general, comparing with non-complex domains, the analysis of usability data demands for additional time and more complementary effort in complex domains.

Chilana et al. [32] interviewed 21 experienced usability professionals to uncover which problems are commonly faced in complex domains. Based on this study, there are three characteristics of complex domains, which aggravate the work of usability experts:

- The domain-specific terminology aggravates the understanding of the domain and the communication with domain specialists.
- Complex domains encompass various unique situations and frequent exceptional conditions, which is why usability professionals have difficulties to fully apprehend the domain.
- Getting access to domain experts is challenging, yet crucial to understand domain-related details, and to find representative users participating in usability tests.

3.3.1. Specific Challenges for Usability Testing

Complex domains impose challenges during all steps of usability testing. Within the planning phase, usability experts reported difficulties in knowing where and how to begin, forming the right questions, defining realistic tasks, and understanding domain-specific concepts. While executing the test, usability specialists struggled to interpret the relevance and significance of observed problems, and had trouble to prepare for exceptions. Compared to non-complex domains, the analysis and reporting process demand for additional time and effort. Experts felt less confident while recommending design changes, and were overly concerned about their credibility. Bearing in mind that usability specialists usually lack in domain knowledge, the difficulties in applying analytical and simulation-based evaluation methods become clear [115].

3.4. Usability Evaluations of Medical Devices

As mentioned in Chapter 2, OR can be considered as a complex domain. Hence, the aforementioned difficulties for usability specialists can be generalized for this domain.

This section addresses some studies reporting about the importance of usability for medical staff and introduces previous works on evaluating the performance of usability evaluation methods in this domain. Furthermore, the role of usability standards for manufacturers of medical equipment will be introduced. In [76], Jannin discuss about assessment in CAI domain.

3.4.1. Usability for Medical Staff

Since the different aspects of usability may stand in conflict with each other, they cannot always be weighted equally for usability goal setting [108]. Therefore, it is crucial to realize what constitutes usability for the medical staff, before conducting tests in this domain. Liljegren investigated the importance of each Nielsen's five usability components, from the perspective of medical staff. Using a questionnaire, he asked 80 physicians and nurses of intensive care and surgical units to grade each of the five aspects regarding its contribution to usability. The most significant component for the medical staff was 'difficult to make errors', making up 30% of the overall usability. The components 'easy to learn', 'efficient to use' and 'easy to remember' were each graded with 20%. Satisfaction received a grading of only 10% [92]. The findings of Liljegren provide a good undrestanding on how to prioritize usability attributes for medical devices.

3.4.2. Best Practice in Medical Domain

In order to identify appropriate usability evaluation methods for medical devices, Liljegren compared four common evaluation methods within a medical environment [92]. He evaluated hierarchical task analysis, cognitive walkthrough, heuristic evaluation and usability tests. Based on his work, usability tests fulfill all relevant criteria and are also able to cover all components of usability. He recommended usability testing to be the primary evaluation method for medical devices. Cognitive walkthrough and hierarchical task analysis can be applied to complement usability tests. In [90], Kushniruk et al. report about applying a usability testing methodology during a period of 10 years. In general, usability testing methodologies include the following four stages:

- *Planning*: This stage include all the preparation steps which should be performed before the test such as defining test objectives, test tasks and creating the questionnaires.
- *Testing*: The real test is conducted with representative test subjects. Kushniruk et al. propose the thinking-aloud method for medical environment. The entire session is

audio and video recorded, including footage of all computer screens and the actual users while interacting with the system.

- *Analyse*: The next step is to analyze the gathered recordings. Depending on the objectives of the usability test, different approaches to analyze captured data can be used, ranging from informal reviews to formalized methods.

- *Report*: Finally, the results need to be interpreted and reported to improve the system. This may include summarizing various aspects of the system, such as frequency and importance of identified problems.

3.4.3. Regulatory Constraints

Before placing a new medical device on the market, manufacturers need to consider several usability related regulations imposed by law in most countries. As an example in Germany, the 'Medizinproduktegesetz' [116] stipulates documentation and process requirements regarding a variety of aspects of a medical system, including its usability. Although these laws are country specific, most countries have adopted harmonized standards as basis of their regulations. In order to verify the conformance with country law, manufacturers need the approval of the responsible institution, e.g. the Food and Drug Administration (FDA) in the U.S., which assesses the compliance with relevant standards. Regarding the usability of a medical device, the recently published standard *ISO/IEC 62366: Medical Devices - Application of Usability Engineering to Medical Devices* [72] should be considered. It replaces the *ISO/IEC 60601-1-6* standard, and aims at reducing the risk of medical errors due to poor interface design, by proposing a specific usability engineering process, which should be applied by manufacturers. To comply with the standard, manufacturers are required to document all aspects of the process in a Usability Engineering File, providing traceability to demonstrate that the Usability Engineering Process has been followed [64].

3.5. Usability Evaluations Support Tools

Performing usability evaluations is not always an easy endeavor due to challenges associated with the acquisition and the management of usability data as well as the analysis of huge amount of collected information. In order to overcome these challenges, several usability evaluation support tools have been proposed in different domains as reported by Ivory and Hearst [73]. Such tools reduce the required amount of labour and costs assigned to these tests and therefore they can be performed on a frequent, iterative basis, within

the development life-cycle. Morae [101], Mangold INTERACT [68] and Nodlus Observer [14] are commercially available tools which consist of a recorder and analysis tools. Several tools are further introduced for usability studies of web-based applications such as Web Usability Probe [4]. Technically, they analyze the html source of the web pages and by adding a spy script for each UI control, generate a usage report. Similarly, authors in [7] propose HUIA framework to record and visualize the interaction logs for applications on smart phones. SAVE [65] provides comparable functionalities for augmented reality applications where users interact in a virtual environment. During the analysis stage the recorded data can be played back in the virtual scene, providing interaction information in the same fashion as the recorded videos. In the OR context, there are several methodological studies available on the topic [130]. However, despite its vital effect, to our knowledge, there is no such support tool for the OR domain. Hence, usability study is often neglected due to its complexity and relatively high demand of time and resources. Furthermore, none of the previous frameworks are considering the complexities of the OR, hence making them impractical in this context. The closest work is presented by authors in [104] to record surgical workflows. This is a valuable tool to monitor activities within a surgical intervention; however it is not intended to be used for usability studies and does not record any user feedback or interaction logs.

3.6. Summery and Conclusion

This chapter presented the required background about the usability and the usability engineering. Also we explained several evaluations methods, categorized based on the classification approach, introduced by Ivory. State of art in usability evaluations in complex domains and challenges facing engineers in such conditions have been also discussed. Moreover, several aspects about usability studies of medical devices have been explained. At the end, we presented well-known usability evaluations support tools, which are mostly developed for other domains.

CHAPTER 4

OR-Use Framework

THERE are several tools to facilitate the usability testing in different domains, however, there is no such tool for the OR. This is despite the fact that OR is a complex domain and acquisition, management and evaluation of the usability data in such domains can be challenging. In this chapter, after investigating the features of the existing tools and observing the practical requirements specific to the OR, we summarize key functionalities that should be provided in a usability testing support tool for intra-operative devices. Furthermore, we introduce the OR-Use framework, a usability testing support tool for the OR, which has been designed to fulfill the introduced requirements.

4.1. Introduction

The complex nature of the OR domain makes the production of usable intra-operative devices very challenging. The OR is a collaborative environment where multiple potential users perform together, each with different individual roles, e.g. surgeon, nurse. Usually, intra-operative devices are targeting more than one of these roles and therefore their usability should be studied for each individual target role. Additionally, the usages of intra-operative devices are usually fused in activities of a higher level process model known as surgical workflow [104]. In this situation, defining atomic test tasks as it is done for websites or handheld devices is not practical. Moreover, the intra-operative devices are technically much more advanced compared to web sites. They are often composed of additional external hardware such as probes and tracking systems, which are integrated as

part of their user interface. Taking into account that interaction with all these external parts of the UI should be also considered within the usability tests, the challenges facing the usability specialist in this domain become clear. In Chapter 2, we proposed a conceptual model for the OR. This model decomposes the complexities of the OR domain into three views as surgical workflow, human roles and intra-operative device. In this chapter, we propose the main functionalities required for a usability testing support tool and explain a possible architecture for exploiting models similar to the one proposed in Chapter 2 for supporting intra-operative usability testing.

4.2. Requirements Specifications

Investigating functionalities available in preliminary tools in other domains and based on observations made with medical industry, in this section we summarize the functionalities and requirements of a usability testing support tool for the intra-operative devices.

4.2.1. Usability Testing Requirements

Usability testing is performed based on a predefined methodology. Hence, any usability testing support tools should provide certain functionalities within each stage of the considered methodologies. As explained in Chapter 3 these methodological approaches mainly are composed of four stages of *planning*, *testing*, *analysis* and *report*. This section explains the functional requirements of such a tool for each of the aforementioned stages:

4.2.1.1. Planning

This is the first step of usability testing and usability support tools in this context should provide the specialist with a proper modeling technique for initializing the parameters of the OR domain model, such as:

- *Workflow Model*, which explains the sequence of activities during a surgical intervention. Having this model, the user interactions can be associated to corresponding stage within the operation.

- *Human Roles*, which define different actors within a given surgical intervention. Modeling the OR roles helps to understand the usability measures from the perspective of different users.

- *Device Model*, which contains all the features in a system's interface that user can interact with, such as UI controls or handheld probes. This model helps to track the user interactions.

4.2.1.2. Testing

During this stage, a wide range of usability data is synchronously recorded. Within this work, we refer to all the data which has been collected during usability tests as usability data. In the medical context, it is very important that in no circumstances neither of the patients personal information should have been stored nor be recoverable from the captured data. For example in a case, where an on-screen keyboard is used for entering patient related information, only key stroke time and duration should be recorded, without any additional data about the pressed key. This is due to the fact that usability specialists, often are coming from outside of the medical center and are not authorized to access this information. The recorded data modalities can be summarized as follows:

- *Video Recording*, which is the most common technique in usability testing as it can be used to find out about the effort users make accomplishing a given task.
- *Performance Logging*, which is storing all the users' interactions with different features of the interface. Excessive amounts of clicks, scrolling and probe movements, are possible indicators for poor efficiency.
- *Recording Annotations*, which provides a great insight into the real feelings and comments of the users about the system. A dedicated tool can facilitate this and furthermore can be utilized for labeling the start and end of workflow stages. Portability and easy deployment are among the main characteristics of such a tool [104]. This is due to the fact that the monitored surgeries usually take place in different ORs, often with a very short notice. Using a portable device satisfies these needs, however, this tool itself should be highly usable due to time constraints during surgery. Small display size and limited interaction possibilities are among the main challenges for designing an annotation tool for small portable devices like smart phones.

4.2.1.3. Analysis and Report

Large sets of collected usability data are rarely explanatory on their own and should be analyzed for making conclusions. Several functionalities can be provided to assist the analysis and report processes:

- *Data Retrieval*, which is a fundamental part of many usability support tools that provides a way for searching, filtering and retrieving a subset of the collected usability data. Different aspects of the domain model can be used to filter the data.
- *Video Indexing*, which provides a mean to retrieve the corresponding video segments for each piece of usability data such as user comments or performance logs. This facilitates the analysis stage by reducing the required time for browsing the videos.
- *Visualization*, including graphs and other diagrams, which should be used where possible to draw attention to the critical issues. These can significantly reduce the time required for exploration and decision making processes. Additionally, such visualization can be used in purpose of reporting the detected usability problems to the development team.

4.2.2. Non-functional Requirements

Beside the above mentioned requirements related to usability testing methodologies, other requirements should be considered for a practical usability testing support tool in the OR. The following sections highlight some of these requirements.

4.2.2.1. Configuration Management

An interesting aspect regarding the usability of a device is to compare two different versions against their usability. This kind of cross configuration comparisons can provide important information upon design decisions and requires proper configuration management features in the testing support tool. There are several definitions of configuration management. Hass [60] defines configuration management as an unique identification and status of the selected product components, and products during the life of a system. In case of medical devices, such information include the model number, hardware and software versions as well as updates that have been installed. It is important to assess this information and create a mapping between the system configuration and the usability tests, in order to facilitate an evaluation.

4.2.2.2. Multicenter Study

Multicenter studies are commonly applied by manufacturers of medical technologies to obtain as much as possible usability relevant data for a medical device, within a reasonable time. A multicenter study is a clinical trial that is conducted in several clinical centres by

different evaluators. Since many participants are involved in this tests, a multi center study has a higher scientific value than a single center study. However, the administrative expenses are much higher [45]. This raises some challenges for management of usability data collected in these tests, including storage and organization of the usability data and data transition related aspects, e.g speed, cost and security.

4.3. Architecture Overview

In this section we describe the architecture and main components of the OR-Use framework, shown in Figure 4.1, developed to achieve the above mentioned requirements. According to the classification of Ivory and Hearst [73], the OR-Use framework solution for usability testing involves: capturing logs generated at client-side and storing them on the server-side, supporting analysis and a number of visualizations to ease the identification of the usability issues, and can be used both within test tasks and real usage of the device. The OR-Use framework exploits the modeling technique presented in Chapter 2. During the planing phase of usability testing, three views of the OR domain model is initialized by defining, surgical workflow stages, human roles and intra-operative device as well as their mapping in an XML file, as shown in Figure 4.1. Figure 4.2 demonstrates the schema of this XML file. Different parts of the OR-Use framework use this file during their initialization.

4.4. Usability Data Acquisition

In order to collect information during the usability tests, two different tools has been developed in the OR-Use framework. Additional to the collected data with these tools, captured video and other device specific files can be attached to the usability data. The timing is synchronized among different modules using an online time web service. Here, we explain main aspects of the data acquisition tools.

4.4.1. Performance Log Conversion Tool

Wide range of technologies is being used by producers of intra-operative devices, which often varies widely based on their specific requirements. In order to stay independent and extend the functional domain of the OR-Use framework, a conversion kernel is developed to convert the logs generated by a specific device to a uniform XML representation. For each new device, a component can be provided and added to the conversion kernel. As

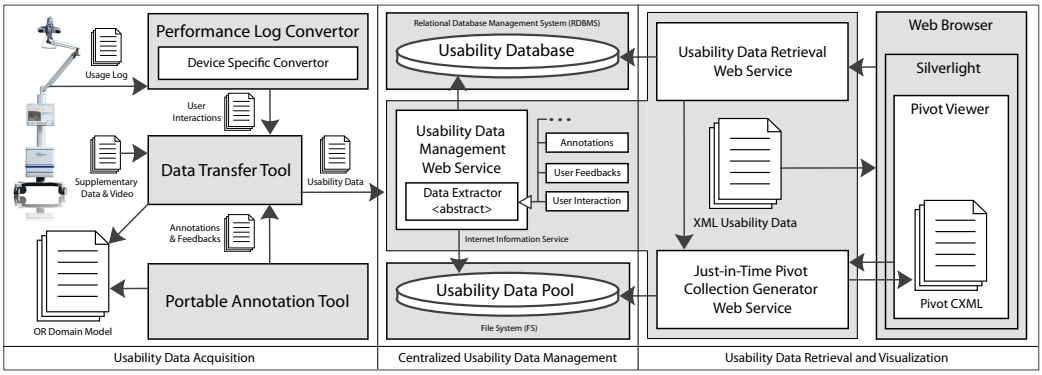


Figure 4.1.: Architecture of the OR-Use framework.

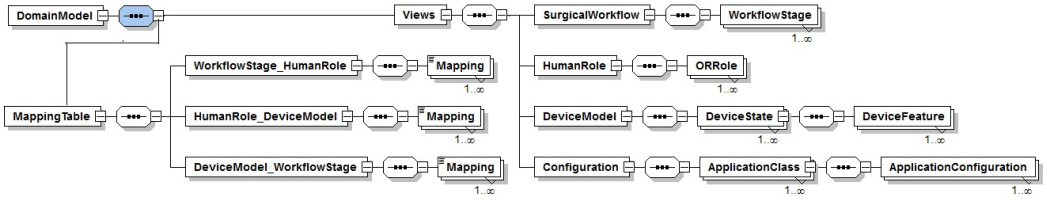


Figure 4.2.: XML Schema of the OR domain model file, used for initialization of different parts of the OR-Use framework.

shown in Figure 4.3 the performance log XML file, generated with the conversion kernel, consists of a list of interactions. After each test session this tool is used to export the device performance logs.

4.4.2. On-site Annotation Tool

Collecting user feedbacks is an essential part of usability testing. In order to facilitate this step, we developed an on-site annotation tool, within the OR-Use framework. This portable annotation tool is developed for Android smart phone devices. To annotate comments with this tool, participants and OR members are asked to use think-aloud technique

```
<UserInteractions>
     <UserInteraction control="Button" name="Load" action="Clicked" value="Pressed" time="..."/>
     <UserInteraction control="Slider" name="Zoom" action="Scrolled" value="25" time="..."/>
     ...
</UserInteractions>
```

Figure 4.3.: An examplary exported log file from the conversion kernel.

while operating the system. As mentioned before, the usability of this tool is important for the observer as they may not be familiar with the OR domain. Hence, this application is designed based on the main activities that a usability specialist needs to conduct during a usability testing session in the Operating Room. Figure 4.4 demonstrates the activity flowchart for different steps of annotation tool.

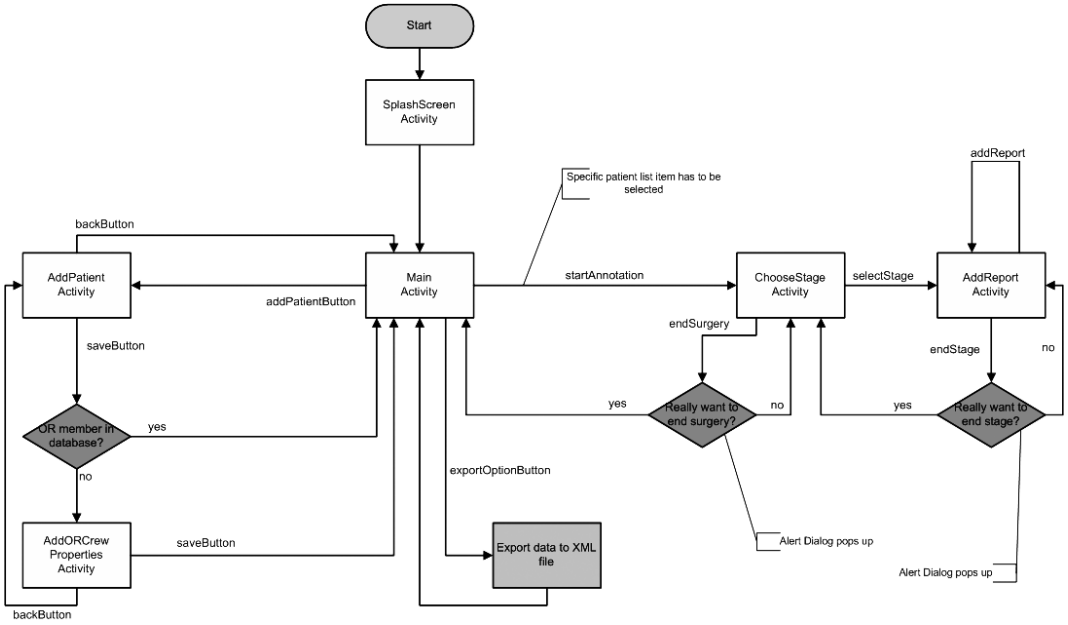


Figure 4.4.: Flowchart diagram of activities during conduction of usability test with the on-site annotation tool.

This tool internally is composed of several activity classes as shown in Figure 4.5, and other classes that are managing the databases for patients and OR members. An additional class serves for converting different time formats. The following shows a brief description of a selection of important classes. Figure 4.6 presents all the UI screens in this application.

- *MainActivity.java*: Main *Activity*, from where all actions emanate.
- *AddPatientActivity.java:* The patient information needs to be entered in this *Activity*.
- *AddORCrewPropertiesActivity.java:* This activity manages the OR members.
- *ChooseStageActivity.java:* The active *workflow stage* has to be selected here.
- *AddReportActivity.java:* In this activity user can enter the annotations.

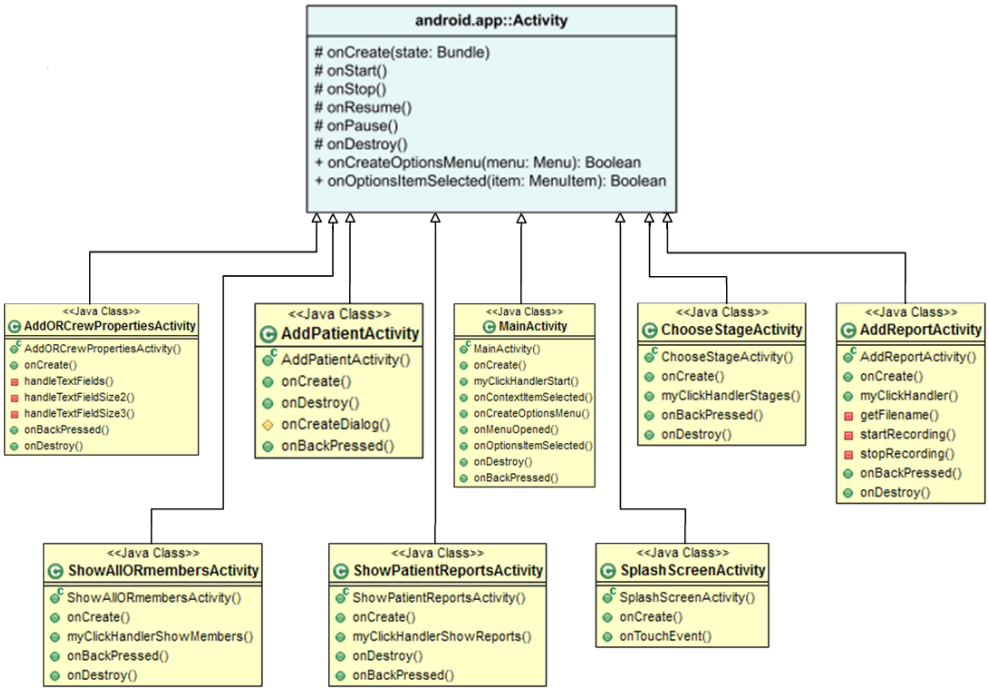


Figure 4.5.: Class diagram, showing the architecture of the on-site annotation tool.

- *ShowAllORmembersActivity.java:* This activity lists all OR members in the database.
- *ShowPatientReportsActivity.java:* This activity lists all reports for a specific patient.
- *DBHelperPatients.java:* This class manages the database of all patients.
- *DBHelperORcrew.java:* This class manages the database of all OR members.

4.4.2.1. Patients and OR Members Management

The on-site annotation tool can be used to document patient profile. Figure 4.6.b displays the form, where all patient relevant data should be completely filled. The patient information can be deduced from the operation protocol. Usually, this protocol comprises the name of the patient with a dedicated hospital ID, date of birth, date of surgery, indication and the names of the medical staff. In order to guarantee data privacy, we do not store the name of the patient. In case an unauthorized person takes possession of the portable device, he can not infer to the patients name. Additionally, before starting the test, for each member of the surgical team a test subject profile can be either defined or selected among

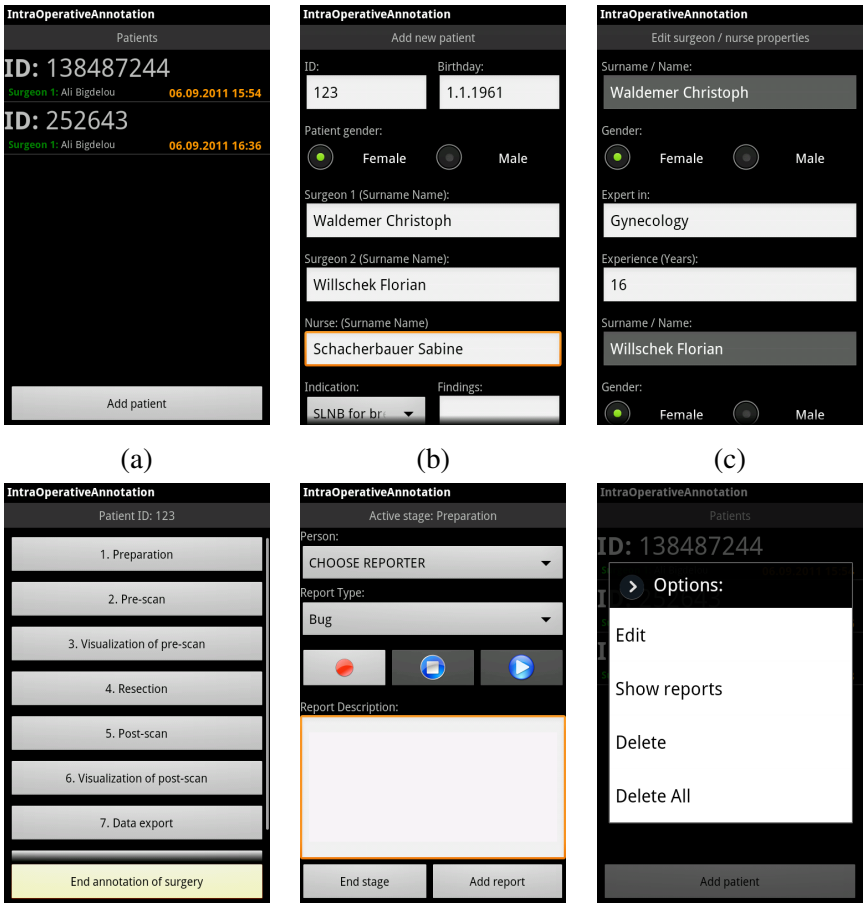


Figure 4.6.: Screenshots of the on-site annotation tool, developed for Android platform.

the previously stored profiles. As shown in Figure 4.6.c this profile contains information such as name, gender, role and level of experience. The latter two factors can play an important part for usability evaluations. For example, one query could be if the experience of a surgeon in a specific medical field correlates with the usability component learnability in respect of the medical device.

4.4.2.2. Following Workflow and Annotations

During the surgery or test, the annotation tool is used to follow the surgical workflow stages as defined in the domain model. For each workflow stage, a button is automatically generated on the workflow page and can be used by the evaluator to follow the surgical

workflow, as shown in Figure 4.6.d. To start the annotation within a workflow stage, the desired stage has to be chosen. This stores the start time of this phase in the local database and the opens the annotation page Figure 4.6.e. Every report that is made, will be dedicated to the specific stage. Thus, it is later possible to filter comments with reference to their workflow stage, in order to find the phase with the most usability problems. To add an annotation, the observer needs to select the OR member, who has a comment, and type of the comment such as complaints, suggestions etc. The comment can be documented in two different ways: typing or voice recording. Beside this information, this tool stores the time in which the comment has been added. This step continues till the annotator selects another workflow stage.

4.4.2.3. Data Export and Other Features

After the operation, all the recorded data can be exported to an XML file. In order to export, the Android *Menu* hardware key has to be pressed while the application resides in the *mainActivity*, Figure 4.6.f. The content of an exported XML file is illustrated in Figure 4.7. Moreover, several additional features are implemented in order to change missing or wrong data, clear the database, list all reports of a patient and manage OR members, as shown in Figure 4.6.f. We hypothesis that these features improve the usability of the annotation tool.

```
<IOA>
  <Header date="18.08.2011" timeStart="09:35" timeEnd="09:41">
    <PatientInfo id="1808" birthday="04.04.1971" gender="Female" indication="SLNB for breast cancer" findings="3+" />
    <ORcrew>
      <Member role="Surgeon1" name="Hoffmann Robert" experience="10" expertin="Gynecology" ismale="1" />
      <Member role="Surgeon2" name="Bonakdar Feridoun" experience="9" expertin="Orthopedic surgery" ismale="1" />
      <Member role="Nurse" name="Braun Anna" experience="" expertin="" ismale="0" />
    </ORcrew>
  </Header>
  <Operation>
    <Stage name="Preparation" stagestart="18.08.2011 09:36:28" stageend="18.08.2011 09:37:45">
      <Report datetime="18.08.2011 09:36:32" reporttype="Bug" reportperson="General" desc="" recordpath="\AnnotationAudios\1313652992853.mp4" />
      <Report datetime="18.08.2011 09:37:13" reporttype="Complaint" reportperson="Hoffmann Robert" desc="Surgeon can not find switch to start device" recordpath="" />
    </Stage>
    <Stage name="Pre-scan" stagestart="18.08.2011 09:37:47" stageend="18.08.2011 09:37:57" />
    <Stage name="Visualization of pre-scan" stagestart="18.08.2011 09:37:59" stageend="18.08.2011 09:38:31">
      <Report datetime="18.08.2011 09:38:04" reporttype="Complaint" reportperson="Bonakdar Feridoun" desc="Visualization takes to long" recordpath="" />
      <Report datetime="18.08.2011 09:38:22" reporttype="Observation" reportperson="Bonakdar Feridoun" desc="" recordpath="\AnnotationAudios\1313653012343.mp4" />
    </Stage>
    <Stage name="Resection" stagestart="18.08.2011 09:38:38" stageend="18.08.2011 09:38:42" />
    <Stage name="Post-scan" stagestart="18.08.2011 09:38:44" stageend="18.08.2011 09:39:17">
      <Report datetime="18.08.2011 09:38:49" reporttype="Positive feedback" reportperson="Braun Anna" desc="" recordpath="\AnnotationAudios\1313653129212.mp4" />
      <Report datetime="18.08.2011 09:39:04" reporttype="Bug" reportperson="Hoffmann Robert" desc="Application crash" recordpath="" />
    </Stage>
  </Operation>
</IOA>
```

Figure 4.7.: Usability annotations XML file, extracted from the on-site annotation tool.

4.5. Client-Server Architecture

For supporting multi-center usability studies, the OR-Use framework has been designed based on client-server architecture, as shown in Figure 4.1. All the captured data are transferred to a server, hosting this information. This section presents different components of the client server architecture of the OR-Use framework. Furthermore, the messaging protocols and processing of incoming data on the server side are explained in more detail. To ensure a simple and rapid transmission, we decided to use a *client server architecture* based on Microsoft's *Windows Communication Framework (WCF)*.

4.5.1. Data Transfer and Messaging Protocol

A client-side tool is developed to facilitate the transfer of the usability data from the testing site to the central management service of the OR-Use. This application merges and encrypts all the generated XML files during a single test or surgery into a single structured XML file. This application transfers this XML message and all the related files via a secure connection to the server. Multiple instances of this tool can be used simultaneously. Since the message contains all the required information about the test setup configuration, it can facilitate evaluation of different devices with different configurations conducted in various locations and hospitals.

The structure of this XML message is shown in Figure 4.8. The root element (ApplicationLog) defines the beginning of the document. This element has three children, Header, Usability Data and Annotations. The Header contains the ApplicationInstanceGUID which identifies the configuration of the medical device. The data node bundles supplementary information such as screenshots, recorded videos and other files generated by the device. The annotation element includes information about the test, configuration and annotations made using on-site annotation tool. Figure 4.9 shows a sample of a combined XML Log file.

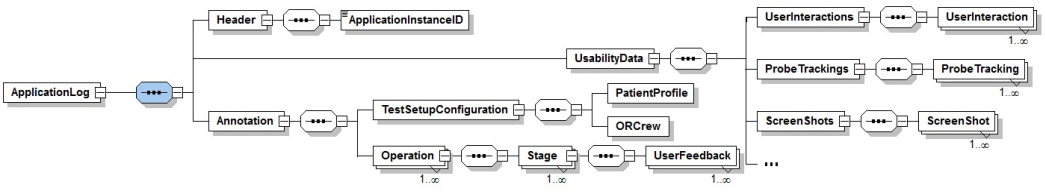


Figure 4.8.: XML schema of the generated message.

```
<DataFile>
  <Header />
  <Data>
    <UserInteractions version="se_ui_interactionLog_v10">
      <UserInteraction time="175035694730" control_name="startAcquisition" stage="pre" />
      <UserInteraction time="175046292559" control_name="pauseAcquisition" stage="pre" />
    </UserInteractions>
    <Screenshots version="se_screenshotLog_v_10">
      <Screenshot time="175032881607" file="screenshot0.jpg" stage="pre" />
      <Screenshot time="175045960465" file="screenshot1.jpg" stage="pre" />
    </Screenshots>
    <Videos>
      <Video file="clip0.avi" start_time="1311153779.531" end_time="4294967295" stage="post" />
      <Video file="clip1.avi" start_time="1311153826.022" end_time="1311153829.992" stage="post" />
    </Videos>
    <ProbeActivities version="se_probeLog_v_10">
      <ProbeActivity time="175033149326" activity="0.000000" duration="50210" stage="pre" />
      <ProbeActivity time="175488012036" activity="21.023863" duration="47565" stage="post" />
    </ProbeActivities>
    <ProbeTrackings version="se_tracker_v_10">
      <ProbeTracking time="175033" positionX="89.17" positionY="-35.90" positionZ="-746.46" stage="pre" />
      <ProbeTracking time="175033" positionX="89.18" positionY="-35.91" positionZ="-746.44" stage="pre" />
    </ProbeTrackings>
  </Data>
  <Annotations />
</DataFile>
```

Figure 4.9.: An examplary XML message, which can be uploaded to the OR-Use usability data management service.

4.5.2. Usability Data Management Service

As mentioned in Chapter 1, usability tests can be conducted within multi-center studies. In order to facilitate this requirement, we manage the usability data in a central usability data management web service. This service is developed using Microsoft WCF. Such service should be accessible from every where in the world at any time, which requires a steady connection to the Internet. The main task of the service is to receive, read and store incoming files that have been sent by a data transfer application. We defined a service contract for this web service as shown in Figure 4.10.

```
[ServiceContract]
public interface IApplicationLogging
{
    [OperationContract]
    Guid RegisterApplicationInstance(Guid appConfigure, string hostInfo);

    [OperationContract]
    void Upload(DataPacket request);

    [OperationContract]
    bool DataExtractor(Guid appLog);
}
```

Figure 4.10.: WCF Service Contract of the OR-Use usability data management service.

The *RegisterApplicationInstance* is used to submit a new application for testing. The

Upload method is called by client applications to send and store a data package from the testing site to the server. Next, the *DataExtractor* method is called to trigger the processing of data on the server.

4.5.3. Server Side Usability Data Processing

When the OR-Use data management service receives data, it performs additional processing in order to decrypt, extract and store the data on the server. Figure 4.11 demonstrates the flowchart of processing in the service. The service parses the XML message and for each node loads corresponding data extractor component. Further processing for each data node happens inside the data extractor component. Data extractors parse the XML data and store them on the server. Each extractor component is developed for processing a special type of data, e.g. user interactions or probe readings. All the data extractor components are inherited from a common base class called *DataExtractorBase*, shown in Figure 4.12. This class has an abstract method named *Parse*. The main task of this method is to parse the input data that was forwarded and accordingly update the database. Such an open architecture allows the proposed framework to be extended for new devices and makes it applicable for different usability standards.

4.6. Usability Data Management

Data management is one of the most important and difficult tasks that usability engineers have to accomplish, especially in complex domains like the Operating Room. In the OR-Use framework, all the retrieved usability data is stored on the server in the corresponding tables of a database. Supplementary materials are separately stored on the servers' file system, forming a repository named data pool. The coming sections explain the used approach to organize the usability data.

4.6.1. Data Binding

To organize the collected usability data, we propose to incorporate this data into the OR domain model, introduced in Chapter 2. This can be accomplished by binding data elements to the elements of a view, as can be seen in Figure 4.13. Since the mapping tables of our model allow for a transition between existing views, it is sufficient to connect the test data to only one view. The most effective way to do this is using the device view, as the data elements are usually related to a certain feature of the device. This helps to identify

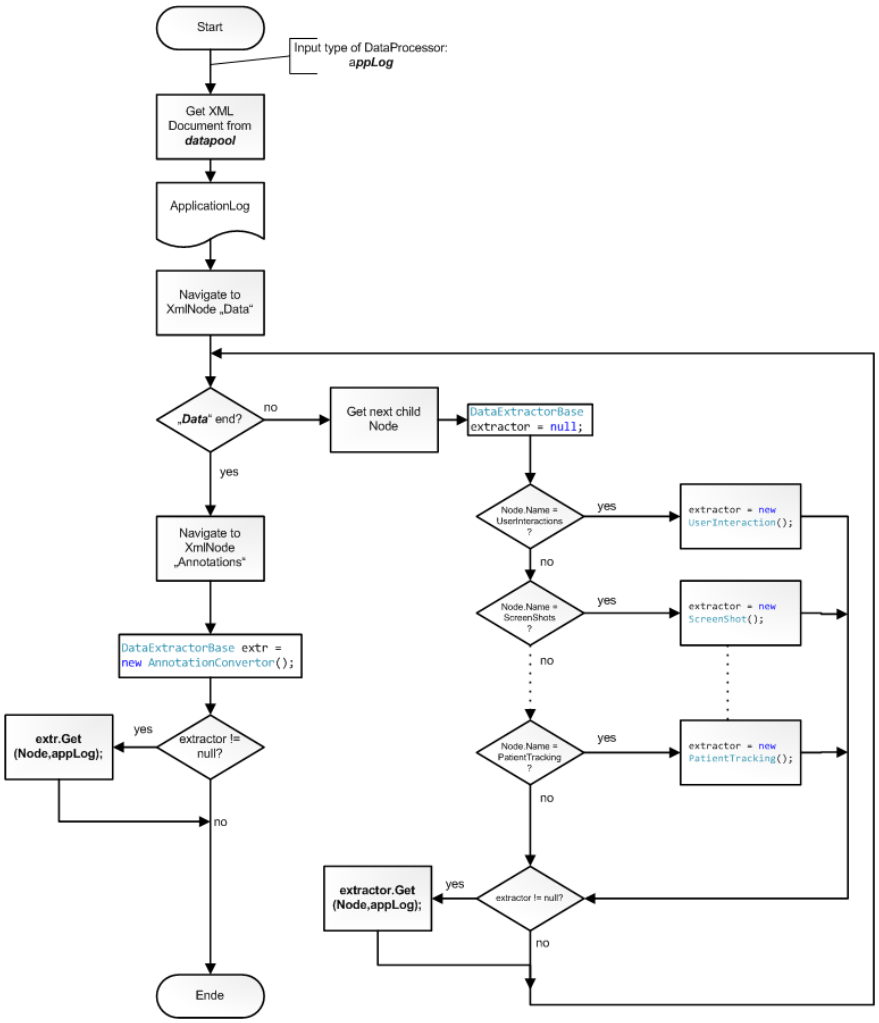


Figure 4.11.: Data processing inside the data management service.

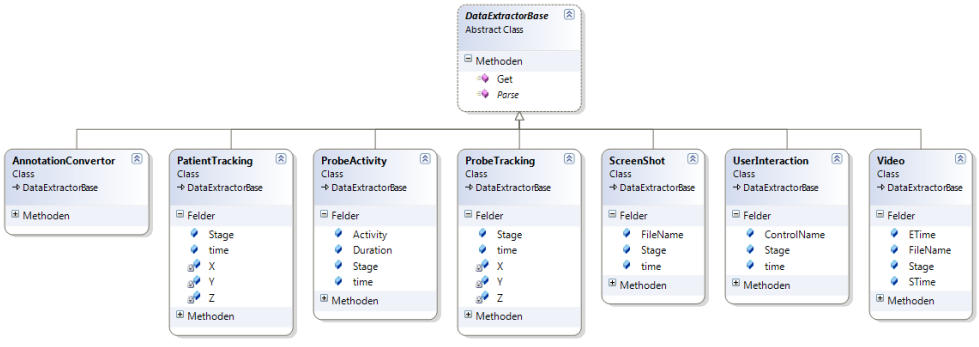


Figure 4.12.: Data extractor components, inherited from *DataExtractorBase* class.

potential correlations between the observed data and the domain model. Therefore, the task of retrieving meaningful information from the usability data can be simplified.

4.6.2. Design of Database

A relational database is used to store data in the OR-Use framework. As shown in Figure 4.13, a hierarchical representation is used in each view. The depth of this hierarchy depends on level of granularity required and is defined in the domain model file. A table is created in the database for each level of this hierarchical representation, as shown in Table 4.1. A foreign key is used to specify the relation to the parent of each node. Usability data are separately stored, each in a dedicated table such as interactions, comments, etc.

4.6.2.1. Surgical Workflow

The defined workflow stages of an intervention are stored in the *Workflow Stages* table with a reference to their dedicated type of surgery. Therefore, performed surgeries and its stages, as indicated by the on-site annotation tool, can be mapped to the device and performance logs in the database. Furthermore, workflow stages are linked to the next and previous stages in order to facilitate an efficient data mining.

4.6.2.2. Human Roles

The surgical staff information and their roles, gathered using the on-site annotation tool, are stored in separate tables of *Operators* and *Human Roles*. Therefore the name, experience and expertise of an OR member is comprised. The roles can be changed during

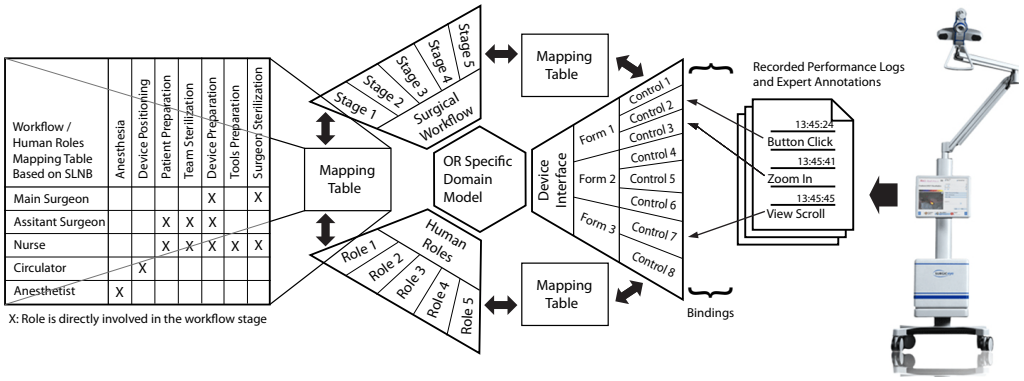


Figure 4.13.: Binding of usability data to the OR specific domain model.

View Name	Level 1	Level 2	Level 3
Surgical Workflow	Surgeries	Workflow Stages	
Device Model	Device Configurations	Device States	Device Features
Human Roles	Human Roles	Operators	
Configuration	Device Classes	Device Configurations	Device Instances

Table 4.1.: OR domain model tables inside the OR-Use Database.

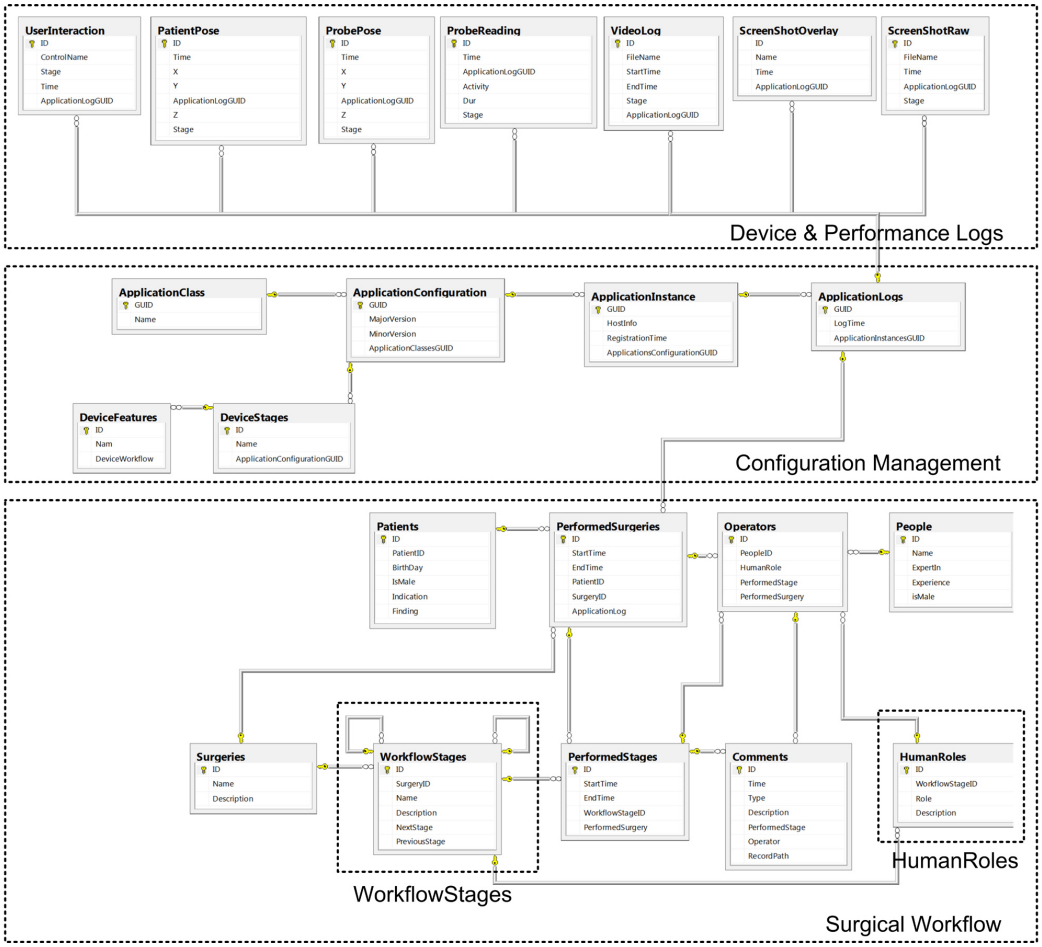


Figure 4.14.: Overview of the database tables inside the OR-Use framework.

an intervention, when shift changes in long surgeries. Hence, an instance of a surgical member is created for every stage that particular operator attended and additionally, he is assigned to the corresponding *Human Role* (e.g. main surgeon, assistant surgeon or nurse). A detailed view of the tables can be seen in Figure 4.14.

4.6.2.3. Devices and Configurations

In order to properly organize data coming from devices with different configurations, three additional tables are added to the database, *Device Classes*, *Device Configurations* and *Device Instances*. *Device Classes* table is used to store the general type of the device. *Device Configurations* table stores information about software and hardware versions and *Device Instances* table contains a list of all the device instances which currently are registered in the system for the test. Moreover, two additional tables, *Device States* and *Device Features*, store device related information defined in the OR domain model.

4.6.3. Structure of Datapool

The datapool is a place on the OR-Use server that we store all files collected during tests. A hierarchically structured file-system is used for saving the collected information as files, shown in Figure 4.15. After receiving the files from the client, a separate folder is created in the datapool with a unique name. These files also can be bound to the entities of the OR domain model, to increase the access speed.

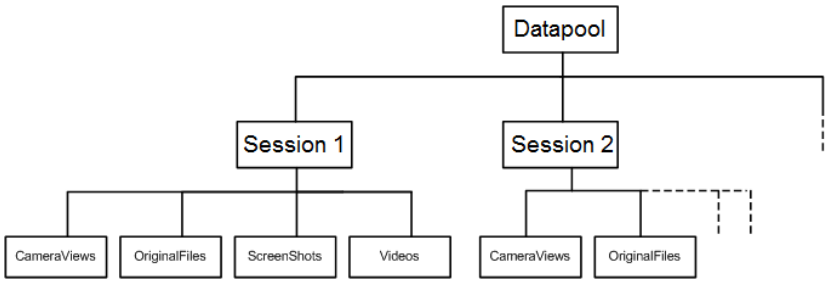


Figure 4.15.: Hierarchical structure of the files inside the OR-Use datapool.

4.7. Usability Data Retrieval

Two web services have been developed, to access the stored usability data on the OR-Use server either directly (low-level approach) or using visualization (high-level approach).

The later, visualized data retrieval interface, is designed to facilitate the analysis stage of the usability evaluation. Further details on this web service is provided in the next chapter. The web-based nature of these services allows multiple and simultaneous access to the data, which is important for comparison.

The stored usability data on the OR-Use server can be retrieved in form of XML, from standard web browsers, sending a query to a data retrieval web service, as shown in Figure 4.1. This query contains a set of key and value pairs, which are used to filter the retrieved results. The first key is *collection*, which specifies the usability data type. For all the database tables shown in Table 4.1, a key with similar name is defined. On the data retrieval service, this query is parsed and a SQL command is generated and executed against the database. The retrieved results are then returned as XML, which can be used for the development of additional analysis support tools such as data visualization or data mining. For example, the following query allows the usability engineer to retrieve the interactions that the surgeon has done with the brightness slider, during the scanning stage, on version 2.3 of the device under the test:

```
HostAddress?collection="UserInteraction"&Configuration="2_3"&Workf
lowStage="Scanning"&HumanRole="Surgeon"&DeviceFeature="Brightness"
```

4.8. Technology

Several technologies and platform has been used during the development of the OR-Use framework. The portable annotation tool is developed for Android OS. The client-side tool to convert and upload the data is developed as a Managed Windows Form application using Microsoft .NET platform. The server-side usability data management web-service is developed using Windows Communication Foundation (WCF) technology and is hosted on an internet information service (IIS) 7.0. The Microsoft SQL Server 2008R2 is used as relational database management system (RDBMS). This section provides some background about these technologies, however, a detailed discussion on these topics is not in the scope of this work.

4.8.1. Android Platform

Android is an operating system for smart phone devices, built on a Linux 2.6 kernel and supports core tasks like storage and process management, security management and network communications. The support of different hardware is handled by a special driver

model. Its Application Framework provides the basic frame for unified application architecture [103, 146]. Android contains a number of core libraries, which are providing most of the functions of the Java core libraries. Every Android application runs in a separate process with its own instance of the Dalvik Virtual Machine (Dalvik). *Views* are an important component in the development of Graphical User Interfaces for Android applications. A *View* occupies a rectangular area of the screen and is responsible for drawing graphics and handling events. All graphical elements like *buttons, labels* and *text fields* are subclasses of *View*. An *Activity* lays the foundation for an application with graphical interface. It represents a single screen with a user interface. Almost all activities interact with the user, so the *Activity* class takes care of creating a window in which the User Interface can be placed. Therefore, one or several *Views* are called by an *Activity* to display the elements on the screen of the mobile device.

4.8.2. XML Web Services

Web services are software components with an interface to communicate over the internet. With the release of .Net 3.0, Microsoft introduced the *Windows Communication Foundations* (WCF). WCF is an application programming interface in the .NET Framework that combines and unifies all advantages of predecessor communication technologies (e.g. Enterprise Services, Web Services etc.)[89]. Each web service can be accessed with a unique address over the internet, called URI. An URI is composed as *scheme://domain[:port]/[path]*. The *scheme* declares the transport protocol being used, such as TCP or HTTP. The *domain* refers to either a machine name or web domain. In case the communication *port* varies from the default protocol identified by the *scheme*, it can be changed [24, 137]. Additionally each WCF web service requires a contract. A service contract is a collection of operations that are exposed to the outside world [97]. It also defines the communication path in regard to the incoming and outgoing messages as well as the format of the data which is transmitted [137]. Service contracts are typically defined using XML formats, e.g. using Web Service Description Language (WSDL) and XML Schema (XSD). The *eXtensible Markup Language (XML)* is a meta language for representing hierarchically structured data in form of text files [138]. A markup is a mechanism to specify structures within a document. XML files have a tree structure, whereupon the arrangement of *nodes* (also called *elements*) and their children is significantly important [136] [138]. Due to the hierarchically structured data of XML files, they are readable by human viewers and can be automatically evaluated by machines. XML is suitable for storing and exchanging data, which can be represented meaningful as text. [55].

4.8.3. Relational Database

The OR-Use framework employs a *Relational Database Management System* (RDBMS) for managing its data. It is built based on Microsoft SQL Server 2008R2. A Relational Database Management System is a type of *Data Management System* (DMS) of databases, which puts the content in relations to each other. According to Codd [33], it is an essential characteristic of relational database that any relation between data is referred via key values. A relational database is a collection of tables called *relation*, where each of it contains several rows named *tuple*. A *tuple* is a set of *attribute* values. The *schema* of a table specifies the number and types of attributes. To get access to individual tuples of a relation, they have to be uniquely identifiable by some combination of its attribute values, which is referred to as the *primary key (PK)*. Key values, describing relations between different tables are known as *foreign keys (FK)*. A foreign key is a sequence of attributes, which refer to the primary key of another relation.

4.9. Summery and Conclusion

In this chapter, we proposed a framework architecture for supporting usability testing of intra-operative devices. This framework is designed based on the OR specific domain model, introduced in Chapter 2. OR-Use framework provides to record usability logs with respect to the workflows of a surgical intervention. To store the gathered data, a client server architecture is used to transmit the data to a central data management service, where further analysis can be applied. The proposed architecture for the OR-Use framework meets all the requirements described in this chapter for an usability testing support tool, specialized for the OR. In the next chapter, we will describe a novel usability data visualization technique to support analysis, which has been developed inside the OR-Use framework. This framework has been successfully utilized during several usability studies up until now and the results will be presented in Chapter 6.

CHAPTER 5

Usability Data Visualization to Support Analysis

ANALYSIS of usability data in complex domains can become challenging due to lack of domain knowledge as well as often complex relations between collected data. In this chapter, we propose a usability data visualization technique to facilitate the analysis process, using the OR specific domain model and an off-the-shelf visualization tool.

5.1. Introduction

Usability testing is a widely accepted technique, performed to discover difficulties that end-users may face while utilizing an application. After collecting the usability data, any data produced during the whole process of a usability evaluation, usability specialists analyze this information in order to extract key findings about the system under test. It has been suggested that heuristic analysis of usability data is the most effective methodology to analyze the usability evaluation data [86]. Generally, the findings of a usability test are highly dependent on the expertise of the evaluator, especially during analysis. However, the amounts of usability data gathered during a study can be enormous. Additionally, as this data only contains low level facts, careful interpretation is necessary to extract useful information. In complex domains, analyzing usability data is an even bigger challenge, as the complexity of the domain is embodied in the gathered data. As usability professionals do not possess thorough domain knowledge, it is hard for them to perceive the potential correlations between the usability data and its usage context.

In order to support the analysis process in complex domains, we propose a new visu-

alization technique. By applying the proposed modeling approach in Chapter 2, the basis for a comprehensive analysis is set. We hypothesize that the domain specific model is a powerful concept for organizing as well as visualizing the usability data. Since the findings from the evaluation have been connected to the model, the evaluator can easily navigate through the data, connect them in a meaningful way, and interpret the information in the context of the available views. As an example, the evaluator can see the impact of the identified problems on different stages of the *workflow* or on specific *human roles*.

In this work we utilize an off-the-shelf innovative visualization tool published by Microsoft [99]. The main advantage of the proposed methodology is that it allows usability specialist to get a quick overview of particular information that is contained in the usability data and help them to understand its relationship to the usage domain. Such a high level visualization of the usability data can facilitate the analysis process by reducing the analyzing time. Despite simplicity of presented technique, it can be utilized and performed in other complex domains.

5.1.1. Information Visualization

Information visualization has become popular in the last few decades. The main objective of information visualization is to gain knowledge and provide an insight about the underlying patterns in presented data. Therefore, it can be used for exploration, decision making, and explanation [30]. It can also significantly reduce the time required for searching data by grouping the related information [40]. Many visualization techniques have been proposed to efficiently provide users with abovementioned benefits. Among them, Pivot is a fairly new and free visualization technology, released on February 2010 by Microsoft Live Lab, which revolutionized the way large amounts of data can be viewed and browsed. After its introduction, several works have been reported about applications of Pivot in different scientific areas such as [148]. An screenshot of pivot is shown in Figure 5.1 and Figure 5.2.

5.1.2. Usability Data Visualization

While the data acquisition in usability tests may be completely automated, the analysis step often requires usability specialists' heuristic experience through review of the data. When the capturing is automated, the volume of data produced by low-level events can be overwhelming. Hence, visualization plays a crucial role and several visualization methods have been introduced to support analysis of the usability data. Among the first proposed visualization techniques, [57] and [51] showed that annotations and performance logs can

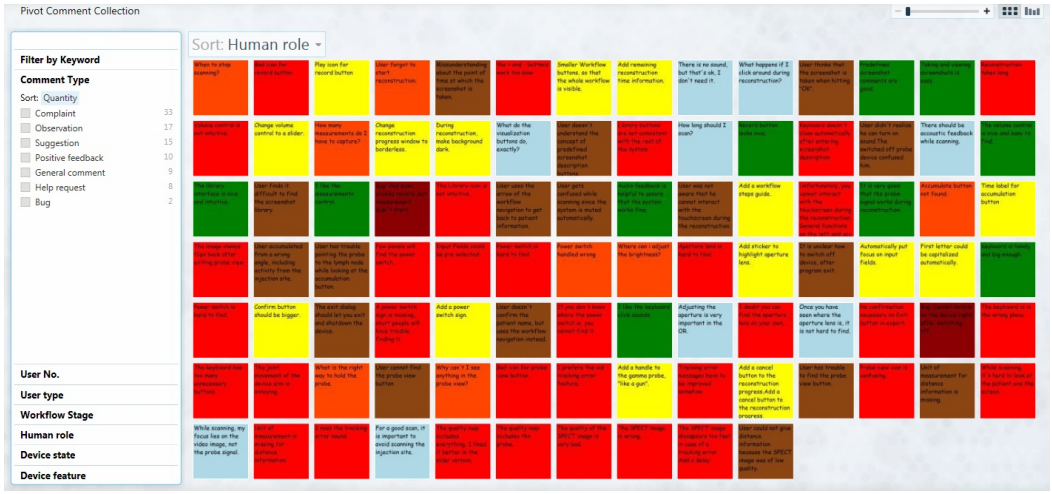


Figure 5.1.: Item view in Microsoft Pivot.

be transformed into timelines. Using these approaches, observers can get a quick overview of the usability test results and recognize underlying occurrence patterns of each event. In a separate study, Badre et al. [9] demonstrated how recorded videos can be indexed using logged events during test session and proposed the possibility of using timelines to navigate throughout video. Using these approaches, it is possible to understand the dynamic of the tests and find similar usage patterns by observing very low-level data plotted over time. There are some other studies in which, higher level concepts of the usability evaluation are exploited in visualization. For instance, Ivo et. al. described how hierarchical structure of task models can be used to group and visualize test logs [94]. Authors in [142] proposed a visualization technique that relies on multiple views of usability data, which is in fact advantageous for data comparison. Pyla introduced Vizibility and the idea of organizing the data according to a hierarchical framework of usability concepts, ensuring consistency and improvement of the analysis process [114].

The visualization methods, described above, have become popular and have been implemented in different usability engineering tools such as [53]. However, to the best of our knowledge, none of the previous usability data visualization techniques are considering the usage context directly in their method. In contrast, our method is implicitly based on such a model. Through our proposed approach, we can inspect the usability data, including performance logs, video streams, and high-level expert annotations according to their impact on the usage context and therefore can be effective for usability studies in complex domains.

Figure 5.2.: Graph view in Microsoft Pivot.

5.2. Domain Modeling

To organize the usability data in a way that can be visualized properly, we use the proposed conceptual domain modeling technique in Chapter 2. This approach decomposes the domain of interest into its sources of complexity and establish connections among the defined fragments. As presented in Chapter 2, a view is used to capture a single source of complexity in the domain. A view can be considered as a set of subcomponents called view elements. Typically, one view captures features or functions of the system of interest, while another covers involved stakeholders. Depending on the target domain, more specialized views can also be defined. As described before, in the OR, three views can be defined as surgical workflow, human roles and device features. To represent the structure of a view in more details, we use a hierarchical model where each view element is assigned to several children view elements. The view elements are then considered as conceptual segments inside the specified view. For instance, in a surgical workflow view, the root node is defined as the operation name and in the sub-layers, view elements for each stage of this workflow are defined such as sterilization, patient entrance, preparation etc. As another example, in the assigned view for the device interface model, one may use this hierarchical representation to separate different forms and menus in the first layer and then define features or controls as their children. In this approach, there is no limit on the number of views and the depth of the hierarchy of its view elements. These depend on the level of abstraction that one considers for the usability evaluation. A similar representation for views

is used in the database of the OR-Use framework. After defining the main views, mapping tables are defined to preserve the correlation between elements of different views. While mapping tables abstract the connections among different views, they provide the opportunity to move from one view to another. Figure 4.13, shows the OR specific domain model that we considered in our case study.

5.2.1. Usability Data Binding

As explained in the previous chapters, before starting the analysis session, the domain specific model should be defined in close collaboration with domain experts, as they provide detailed insights in order to understand the complexity of their field. The next step is to create proper bindings between each piece of the collected usability data and the view elements in the model, as illustrated in Figure 4.13 and discussed in Chapter 4. Given a dedicated view for the application interface, encapsulated high level UI segments (i.e. forms, dialogs, etc) at the first layer, and controls as well as features at the next layer, the process of binding usability data to the elements of this view could be automated in the usability study tool and in our case OR-Use framework. This is due to the fact that performance logs and expert annotations are usually rooting from a certain feature or control in the device interface. Therefore, they can be bound to the corresponding view element in the device interface view. Additionally, for each portion of the usability data, further bindings to the remaining model views are constructed by tracing the cross view connections stored in the mapping tables of the model.

5.3. Usability Data Visualization using Pivot

In order to visualize the collected usability data, we use Pivot [99]. The underlying data structure of Pivot is a collection of items called Pivot collection. The items in a collection have common facets (properties). The collection can be browsed, filtered, and sorted by the facet values of its items. Figure 5.1 shows the user comments collection visualized via pivot. Internally, collections are represented as XML files with CXML extension, which contain items and all other information required for visualization. Collections are either static or dynamic. The static collections are simply used when no variations in the items are expected and typically the total number of items is not exceeding 3,000. In contrast, there is no size limitation for dynamic collections and the CXML file is generated dynamically as a response to a web query. The static collections are simple to generate but dynamic collections necessitate a sophisticated web service to generate CXML file upon request.

The main advantageous of using Pivot for usability data visualization are as follow:

- Pivot is capable of handling large set of data and enables users to sort, browse and filter data in a unique way.
- The underlying data format is quite general and can be used in many scenarios.
- It is based on Silverlight technology, which makes it usable even over the web.
- Due to its client-server architecture, multiple users can access the visualization interface simultaneously.
- It provides a strong and still simple to use SDK for integration of different scenarios.

5.3.1. Pivot Web Service

In the OR-Use framework, we decided to use dynamic Pivot collections, developed as a web service to generate the CXML files dynamically. This method is called just in time (JIT) collections in Pivot terminology. The main benefit of dynamic collections is to have more flexible structure, which enables fast responses to complicated queries. For instance, using custom request, it is possible to filter data with a given condition e.g. female test subjects or between specific periods of time. We deployed our pivot JIT service on IIS 7.0 running on a Windows Server machine. User can query the CXML files with a similar format as data retrieval interface. This query contains a set of key and value pairs, which are used to filter the retrieved results. The first key is *collection*, which specifies the usability data type. For all the database tables shown in Table 4.1, a key with similar name is defined. The following query allows the usability engineer to retrieve the interactions that expert nurses have done with the contrast slider, during the preparation stage, on version 1.4 of the device under the test:

```
VisualizationServiceAddress?collection="UserInteraction"&Configur
ation="1_4"&WorkflowStage="Preparation"&HumanRole="Nurse"&DeviceFe
ature="Contrast"&UserExperience="Expert"
```

The Pivot web service forwards this query to the OR-Use data retrival web service, explained in Chapter 4. The CXML file is later generated, based on the retrieved results from the data retrieval interface, shown in Figure 4.1. All the collected usability data is stored in a Microsoft SQL Server database running on the same machine as part of the OR-Use framework. In case of automated usability data collection, where the data is directly stored in the database, this approach makes it possible to visualize the results even during

Facet name	Description or possible values	Comments	Interactions
Control	Name of the control being interacted.	Implemented	Implemented
Comment type	Bug, Complaint, Feedback, Help request, Observation, Positive feedback, Suggestion	Implemented	N.A.
User	Information about the test subject.	Implemented	Implemented
Expertise	Level of expertise of the user.	Implemented	Implemented
Human role	Human role of the person in the OR.	Implemented	Implemented
Workflow stage	Relevant workflow stage.	Implemented	Implemented
UI section	Form or dialog containing this control	Implemented	Implemented

Figure 5.3.: List of the facets for comments and interactions in Pivot collections.

the recording or as soon as the usability data is uploaded. Furthermore, multiple experts can use the visualization, using a web browser equipped with Silverlight.

5.3.2. Usability Data Facets

To form a Pivot collection based on usability data, collection types and their facets should be clarified. Facets are assigned to each item based on the available information in the corresponding table and additionally other properties traceable via mappings. Basically there are two different pivot collections, which are populated based on the collected usability data. First is user comments collection in which each item represents a user feedback. Each user feedback contains several facets as provided in the following list:

- The message of the comment.
- The type of comment, as listed below (e.g. complaint, suggestion...).
- The test user, who made the comment.
- Timing data, i.e. start and end time of the comment annotation in the video.
- The device feature, which is addressed by the comment.
- The device state of the feature.
- The workflow stages, in which the feature may be used.
- The human roles, who interact with the feature.

Moreover, the captured feedbacks can be categorized with various types of comments about the device features. These categorizations are annotated by the evaluator and include:

- *Complaints*: The complaints made by test subjects about the device features.
- *Suggestions*: The suggestion made by test subjects for improvement.
- *Positive feedback*: Positive comments made by test subjects about a device feature.
- *Help requests*: The conditions where the user could not continue the assigned task without help.
- *General comments*: All the interesting statements made by the subjects which are not covered by other categories.
- *Bugs*: Explanations about the actual errors occurred in the system.
- *Observations*: Any observation made by the conductor, in case the user's actions contained significant information for the analysis.

Another possible option is user interactions collection where each item basically demonstrates a single interaction that user has made to a control in the device such as button or a pressed key on the keyboard. To associate facets to these items, beside common information stored for each item such as time, user etc., additional information can be extracted from the bindings of this usability item to the view elements of the model, formed in the previous step, such as workflow stage, human role or device UI. Figure 5.3 shows lists of the facet that we used for each of these collections in our case studies.

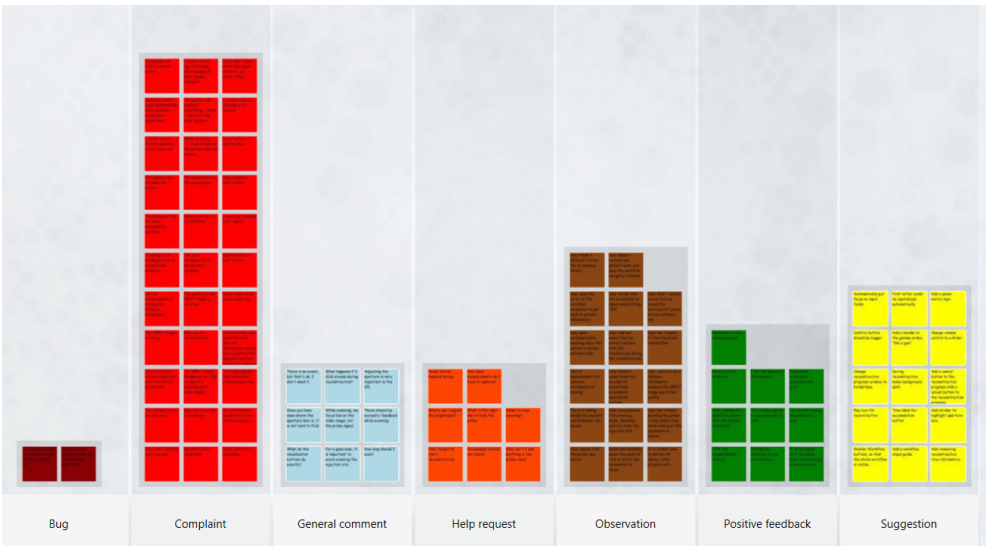


Figure 5.4.: Collection of comments in Pivot, sorted by comment type.

5.3.3. Visual Representation

In addition to the facets, for each item, a visual representation as an image can be defined, Figure 5.1 and Figure 5.2. The image of each item can be used to display a descriptive text, shortened version of the comment for comments collection, and control name for interactions collection. Furthermore, the items background can be color coded according to comment type for comments collection and control type for the interactions collection.

5.3.4. Video Indexing

Each item in the Pivot can be associated with several URIs for linking it to other resources. We use this option to index recorded videos and provide a link to the video file and corresponding time of the item to navigate through video toward point of interest. This feature can be considered as a new way for indexing video streams, recorded during usability tests. By clicking the link for each item, corresponding video at the given time will be displayed. Figure 5.4 shows the resulting collection of comment items in Pivot.

5.4. Application in Analysis

Using the proposed technique, usability data can be retrieved as shown in Figure 5.5.a and organized based on the comment types as shown in Figure 5.5.b. Basically, each comment item has been bound to the corresponding device feature. The mapping tables of our model enable us to visualize the feedback based on various parameters defined in the model. For example, Figure 5.5.c is representing the distribution of comments over workflow stages or human roles. Moreover, the filtering feature of PivotView allows us to selectively study the positive and negative comments, as can be seen in Figure 5.5.d-e. During the analysis phase, this structured visualization of comments helps the usability engineer to identify usability problems of the target system. The first case study in the next chapter will provide a more detailed example on the usage of this visualization technique in real-life scenarios.

5.5. Summery and Conclusion

The proposed domain modeling technique helps the usability specialists to capture different aspects of the usage domain. The domain model breaks down the complexity into distinct views and a connection between the usability data and the model can be made. Hence, usability specialists are able to consider the portion of data, which is interesting for them, based on the context of any available view. This helps identifying potential correlations

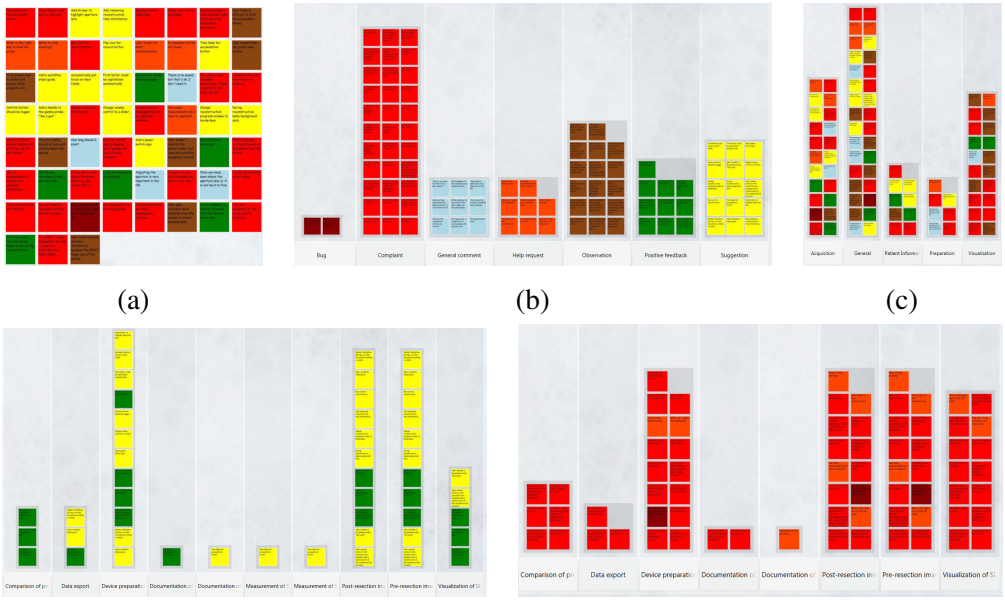


Figure 5.5.: Examplary screenshots of Pivot, showing (a) all gathered usability comments, (b) comments sorted by type, (c) the distribution of comments over *workflow stages*, (d) positive comments / suggestions over *workflow stages*,(e) negative comments / help requests / bugs over *workflow stages*.

between the observed data and the domain. Therefore, the task of retrieving meaningful information from the test data is simplified. The simplicity and flexibility of the proposed model suggest its applicability in other domains as well. Furthermore, the video indexing is possible with a slightly different retrieval approach, compared to existing methods. The video segments can be accessed through the time stored in each element of usability data. Modeling the tasks as a separate view provides similar results as described in [94]. Since the Pivot is a web-base visualization tool, our method inherits similar benefits for analysis within multiple views as suggested in [142].

CHAPTER 6

Case Studies and Evaluations

THE OR-Use framework has been utilized in several usability evaluations up until now. In this chapter, we report about two such conducted usability tests. Additionally, we report about several performed tests to evaluate the applicability and performance of this framework and compare it with similar tools in other domains.

6.1. Introduction

The OR-Use framework has been used to conduct two usability tests on an intra-operative SPECT imaging system developed. The evaluation methodology was based on the usability testing approach presented in [90], and was hence conducted with the thinking-aloud technique (see Chapter 3). In this chapter, the domain modeling Section 6.2 describes the instantiated model for these usability tests, based on the proposed modeling approach in Chapter 2. Next, two conducted usability tests are reported in the following case study sections. The remaining sections present several quantitative and qualitative evaluations performed to evaluate the performance and applicability of the OR-Use framework.

6.2. Domain Modeling

As explained in Chapter 2, the key sources of complexity for usability evaluations in the OR domain are different human roles, the target device, and the surgical workflow. In order to define the domain model for the usability test of the SPECT imaging device, one

of its target applications, the sentinel lymph node (SLN) biopsy, was chosen to be the procedure of interest. This intervention is applied in a variety of clinical cases, including breast cancer and melanoma patients. The aim of this procedure is to inquire whether the lymph nodes close to the carcinoma have metastasized. To locate these SLNs, a radioactive tracer is injected in the affected area. The SPECT imaging device is used to create a 3D SPECT image, which is superimposed on a camera image of the patient. This visualization allows the surgeon to find and dissect the SLNs. Next, each node is examined to determine whether cancer cells are present. A negative SLN biopsy suggests that cancer has not spread to further lymph nodes. A positive result indicates that cancer may also be present in other lymph nodes. In this case, the surgeon will remove more nodes during the procedure [1].

We instantiated our model by capturing 23 *workflow stages* of the SLNB [120, 141, 23], identifying 5 involved key *human roles* and 45 *device features*, grouped in 8 *device states*. This study has been performed in close collaboration with technical staff and cooperating surgeons of the Klinikum rechts der Isar, Munich, Germany. The defined elements are explained in the remaining of this section.

6.2.1. Human Roles

We identified the following roles typically involved during a sentinel lymph node biopsy:

- *Main surgeon:* Responsible for the surgical procedure, and operating the gamma probe of the SPECT imaging device.
- *Assistant surgeon:* Commonly two assistant surgeons are involved, one of them operating the touchscreen of the SPECT imaging system.
- *OR nurse:* Assists during the intervention, prepares the patient and the SPECT imaging device.
- *Anesthetist:* Responsible for the anesthesia of the patient.
- *Circulator:* Observes the surgery, provides assistance to the sterile team, anesthetists, and prepares the SPECT imaging system.

6.2.2. Intra-operative SPECT Imaging Device

The SPECT imaging device, shown in Figure 6.1.a, is used to create a 3D SPECT image directly in the OR. The system also superimposes this information on live video images

within the OR. This visualization allows the surgeon to find and dissect the SLNs. The main elements of the intra-operative SPECT imaging system are the device arm, which contains a tracking system and a regular video camera, a touchscreen to operate the device, and a tracked gamma probe (see Figure 6.1.b). During the study of the SPECT imaging device, the first step was to capture the various states of the system. Next, we identified the features available in each device state. The resulting features tree can be seen in Appendix 12. State - feature pairs were used as elements for the device view.

(a) (b)

Figure 6.1.: (a) The intra-operative SPECT imaging device, (b) the tracked gamma probe of the imaging device.

6.2.3. SLNB Workflow

Together with technical staff we captured the workflow of a SLNB using the SPECT imaging system as described in Appendix 12. It is important to notice that this workflow is idealized, and is not necessarily followed exactly in clinical practice. Each identified workflow stage was selected as an element for the workflow view. The following stages were defined as workflow stages in the OR model of these studies:

- *Preparation:* While the patient is brought to the operating theatre and is being prepared for the surgery, the initial steps which are necessary for the usage of the medical device can be executed (e.g. preparation of the accessories, correct positioning, turning on the device and adding patient information to the system).

- *Pre-scan:* In order to have a reference value of the patients condition before and after the intervention, a pre-incision scan is recommended.

- *Visualization of pre-scan:* The system processes the readings and reconstructs a 3D image of the pre-incision scan and displays it on the monitor.

- *Resection:* This workflow stage involves all the sub-stages where the actual intervention takes place.

- *Post-scan:* After the OR crew have finished the surgical procedure, they may want to check the result of their work by comparing the pre-incision and post-excision scans. Therefore another acquisition is needed.

- *Visualization of post-scan:* The reconstructed image of the post-scan is displayed on the monitor to be interpreted by the users. For better understanding, the image can be compared to the previous ones.

- *Data export:* To document the surgical procedure, most novel medical imaging devices offer the opportunity to export data like 3D DICOM-images of the reconstructions as well as program log files, screenshots and videos of the resulting scans, if available.

- *Dismantling:* The medical device and all modules have to be dismantled and the accessories have to be sent to sterilization unit if needed, in order to be ready for the next surgery.

6.2.4. View Mappings

After the analyse and construction of three views and identifying the corresponding view elements, the mappings between the views were created. Excerpts of the resulting mapping tables can be found in Appendix 12.

6.3. Case Study 1: Synthetic Operating Room

This case study was conducted in winter 2010. Beside the usability testing for the intra-operative SPECT imagin device, the study was designed to demonstrate the strength of the proposed visualization technique of the OR-Use framework. This study has been conducted in collaboration with the manufacturer. The findings of this study provided the development team with a better understanding about the usability of SPECT imaging device in the OR and in general helped to evaluate the usage of mixed reality in this context [110].

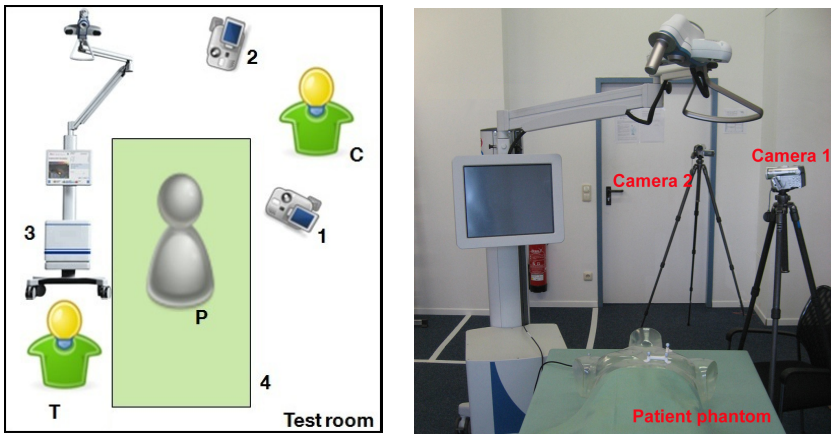
6.3.1. Objectives

As mentioned in Chapter 3, manufacturers of medical devices are obligated to assess the usability of their systems, and are advised to document their evaluation process according to the ISO/IEC 62366 standard [72]. Since the SPECT imaging system had reached a development stage in which it could be released to the market, the initial motivation for the conducted usability study was the compliance with regulatory requirements. The actual goal of the test was to identify usability problems which might occur during the application of the device. Furthermore, the system developers were interested in seeing how the device is used in practice, and in acquiring feedback from the target users, to discover opportunities for future developments.

6.3.2. Study Design

The target users of the SPECT imaging system are surgeons. However, as this usability evaluation was the first case study of the OR-Use framework, we decided to conduct a pilot study with technical staff. The test was conducted with 7 participants, who have been selected with various level of domain expertise. Many of the participants were involved during the development of the SPECT imaging and therefore had experience in using the device. Yet a few of the subjects had never interacted with the system before, and were hence regarded as novices. Figure 6.2.a illustrates the layout of the test environment. A patient phantom (P) was placed on a table (4) in the center of the test room, covered by surgical sheets. Two cameras (1, 2) were used to record the test scenery and the screen of the imaging device. The device (3) was placed next to the head of the phantom. The test conductor (C) was seated behind the dummy patient, instructing and observing the session. Figure 6.2.b shows a photo of the test environment.

Based on the view elements of our model, we selected 15 different tasks to be performed



(a) (b)

Figure 6.2.: (a) Layout of the test environment, (b) a picture showing the test lab.

by the participants. While creating these tasks, we attended to their representativeness regarding real tasks performed with the SPECT imaging system in the OR. Furthermore, we tried to cover all important features of the device. This endeavor was without a doubt simplified by our model. In order to take into account differing levels of experience with the SPECT imaging, we separated our 15 tasks into two test cases: *Test case 1* was used for experienced subjects, as it consisted of one long task, covering all important system states. Subjects were asked to determine the depth of the patient's lymph node and make a screenshot of it. *Test case 2* was composed of a combination of 14 short tasks. These tasks were designed to be performed one after the other. Once one task was finished, users were given the instructions for the next one. With this approach, we made sure that even users who were not familiar with the SPECT system, were able to perform the tests. The following list presents a selection of the tasks included in test case 2:

- *Switch on the device and select a workflow:* To begin a test session, a rather simple task was chosen, so subjects can get accustomed to the test situation.

- *Scan and reconstruct:* Users were asked to scan the patient and create a SPECT image. This task covered the most critical steps when using the SPECT system, as the quality of the SPECT image is highly dependent on the quality of the performed scan.

- *Perform an accumulation:* Users had to accumulate the activity of a hot spot over

ten seconds. As the system provides a special feature to do so, we assessed the user's awareness of this function, by assigning this task.

- *Make one screenshot covering all hot spots:* Subjects had to switch to fullscreen mode and make a screenshot in which all hot spots are visible. This task aimed to inquire whether users notice the icon of the fullscreen mode, and possibly the zoom button.

- *Change patient name:* After finishing the main test tasks, subjects were asked to change the patient name they had initially entered. While this task is not representative for real device usage, it revealed how the navigation control is used when given a task out of the regular context.

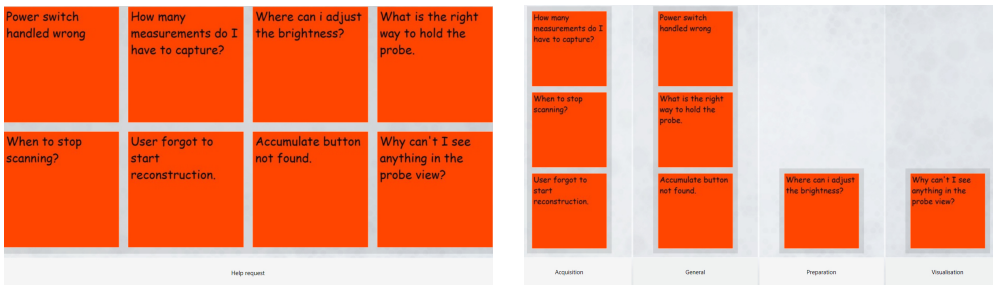
6.3.3. Conducting the Test

Each test was conducted as a standard usability test session [85], lasting between 30-40 minutes, involving a formal greeting, an introduction and a signing of agreements. Several scripts have been developed for the test conduction, including the already mentioned medical case description. In order to expose participants to identical conditions prior to the test, an introductory script were read verbatim to each subject. Each test session was structured as follows. Once the participant had been welcomed, the test conductor read an introduction script verbatim to the subject, explaining the aim of the study, the course of action, and what he or she will be doing. In order to put the test user at ease, an important part of the introduction is the reinforcement that only the product is being tested, not the participant [85, 119]. Afterwards, the test user was presented with a letter of agreement. By signing this document, the subject agreed that any data gathered during the test may be stored and processed within the scope of the study. Vice versa, the conductor confirmed that personal information will be handled confidentially. Next, the participant was asked to fill out a background questionnaire. In order to put the test tasks in perspective, the medical case description was read to the subject once the questionnaire had been completed. In the following step, each participant was given a short demonstration on how to use the device, to assure that he or she has a basic idea how the SPECT device works. At this point of the test procedure, the video cameras were switched on, in case the test user asked questions relevant for the evaluation of the device. After that, the conductor explained the thinking-aloud method, and gave three examples how to verbalize thoughts while operating the system. Finally, the actual usability test was conducted. Each subject was asked to perform tasks of either test case 1, or test case 2, while all interactions were recorded.

During this procedure, the conductor did not interfere or answer questions, unless the subject could not continue the assigned task independently. Occasionally, participants had to be encouraged to think out loud. Once the tasks had been completed, the test session was concluded with a short wrap-up talk.

6.3.4. Analysis and Interpretation

As explained in Chapter 4, after conducting the tests, OR-Use framework associates all comments and performance logs to corresponding device features and states, the affected human roles and workflow stages. Comments are categorized as *complaints*, *suggestions*, *positive feedbacks*, *help requests*, *general comments*, *bugs* and *observations*. This enable us to effortlessly traverse through comments collection, based on the three views of our OR domain model. In total, 94 individual comments have been made. Several findings were reported to manufacture after this stage. As an example, a comparatively high number of complaints has been made regarding the device preparation stage. Further investigation revealed the circulator as the mainly affected role. This highlighted device features which require further improvement. The following two cases demonstrate how the proposed visualization technique in OR-Use framework has facilitated the interpretation of the collected usability data.



(a) (b)

Figure 6.3.: (a) All gathered help requests, (b) help requests in different device states: Acquisition, General, Preparation, Visualization.

6.3.4.1. Help Requests

In case a participant asks for help during a test, a usability problem is likely to be found. Hence, we started to interpret the acquired comments by looking at help requests (see Fig-

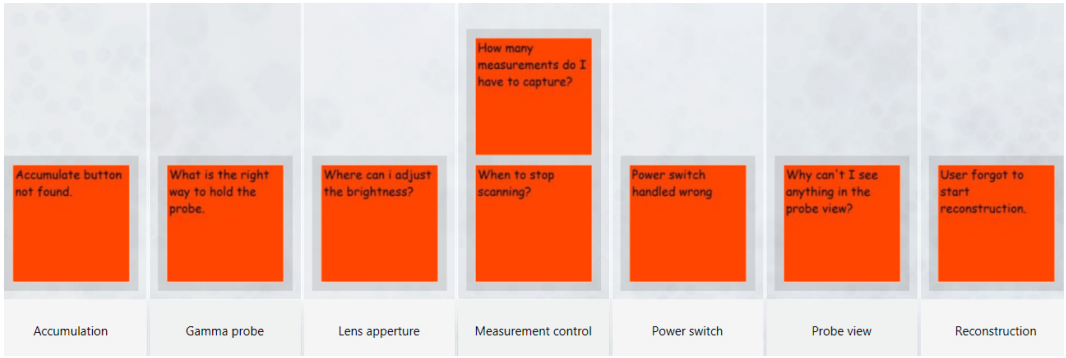


Figure 6.4.: Help requests for each device feature.

ure 6.3.a). To see how these requests are distributed among device features and states, we sorted them accordingly, as can be seen in Figure 6.3.b and Figure 6.4. One incident was logged for the *accumulation feature* (see Figure 6.5). The related task was to accumulate the activity measurements of the lymph node over ten seconds. After reviewing the corresponding video sequence, we discovered that the caption of the accumulation button ('Accumulate') did not correspond to the subjects expectation. Instead, the user was looking for a button with an icon similar to a clock.

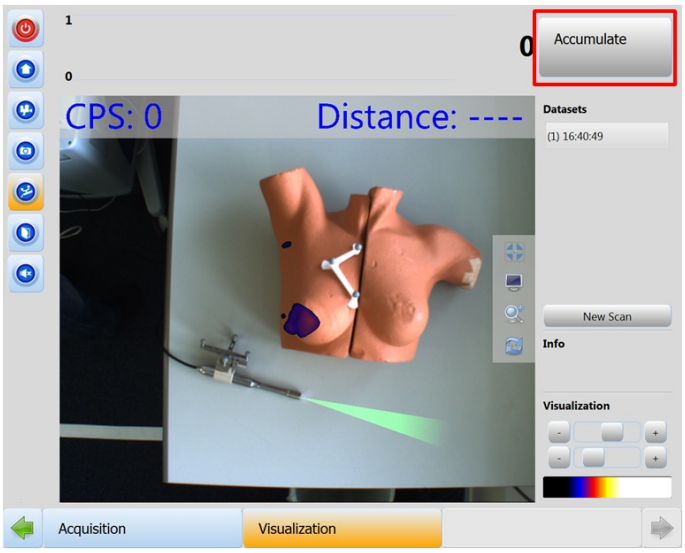


Figure 6.5.: Screenshot of the SPECT imaging in the visualization stage, including the accumulation button (upper right corner).

In order to determine the severity of this problem, we looked at the workflow stages dur-

ing which the this feature is required, and the involved human roles (see Figure 6.6.a and Figure 6.6.b). As the accumulation feature is used by two different roles, and is involved in three different stages in the surgery, one can argue that this problem can have serious effects on the surgical procedure, and hence needs to be attended to.

(a) (b)

Figure 6.6.: (a) Workflow stages affected by the help request regarding the accumulation feature: Measurement of SLN activity in vivo, Measurement of SLN activity ex vivo, and Documentation of residual activity, (b) Human roles affected by the help request regarding the accumulation feature: Assistant Surgeon and Main Surgeon.

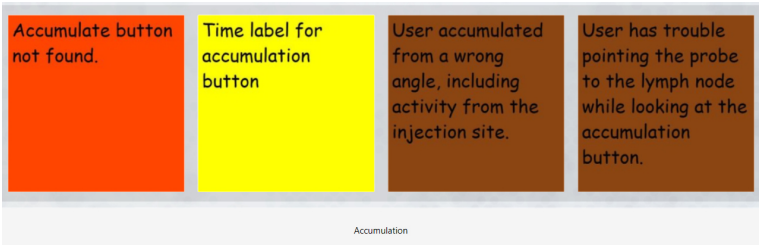


Figure 6.7.: All comments about the accumulation. Colors, *red*, *yellow* and *brown*, represent the corresponding comment type *complaint*, *suggestion* and *observation*.

Figure 6.7 demonstrates all comments which have been gathered for the accumulation feature, including two conductor observations and one suggestion by a participant. Although the observations needed further investigation, the suggestion regarding the label of the accumulation button could immediately be used as input for potential improvements of this feature.

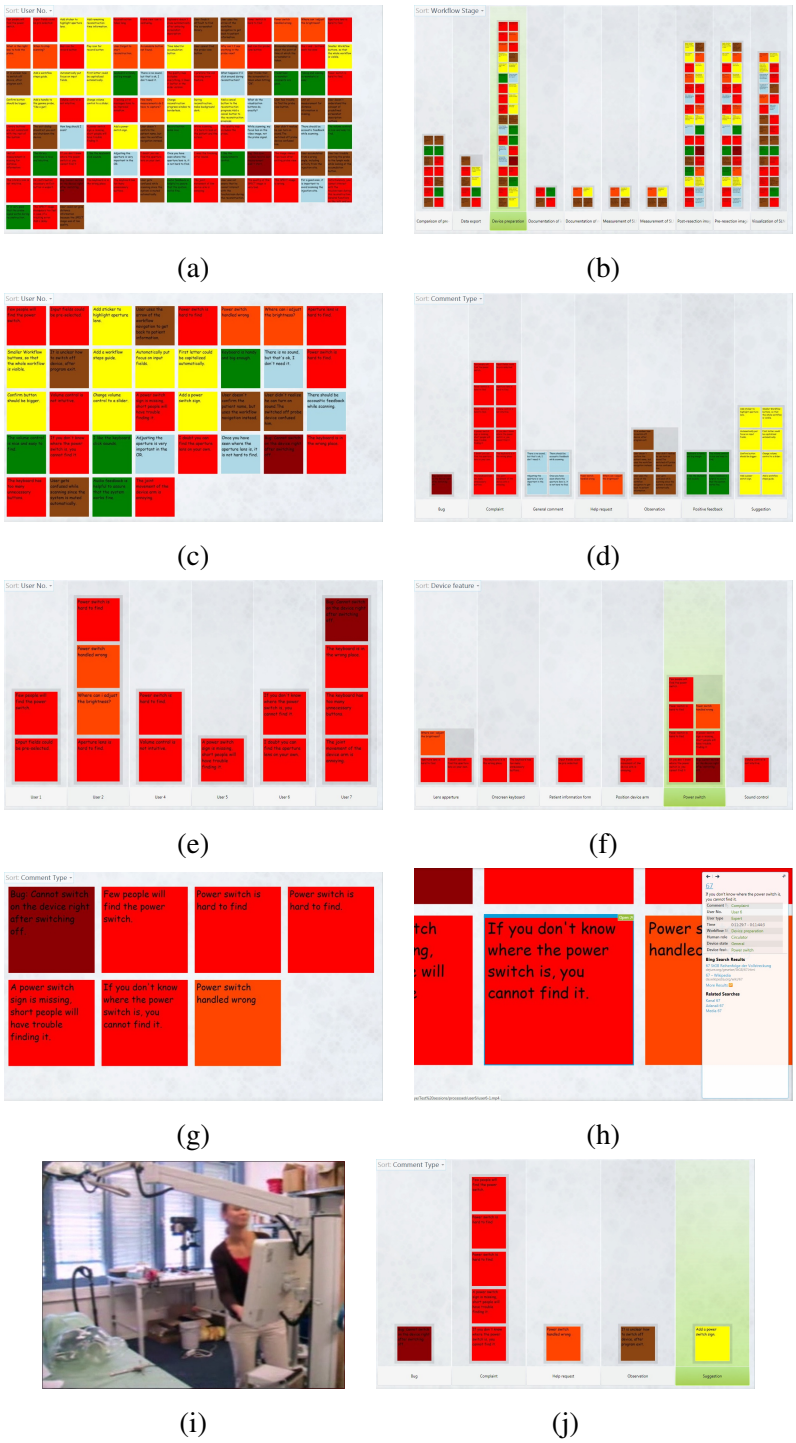


Figure 6.8.: Different screen shots of Pivot, captured during analysis of the usability data.

6.3.4.2. User Complaints

This exemplary analysis case demonstrates how complaints made by test participants can be analysed to extract meaningful results. After loading the collection of user comments with Pivot, the visualized data is not very informative at the first glance, as shown in Figure 6.8.a. Since the facets of comment items were generated based on their relations to the OR model, we visualized them based on their relations to specific workflow stage as shown in Figure 6.8.b. Now, it is simply visible that the device preparation stage has many comments, increasing the chance of facing a usability problem. The subset of comments relevant to this specific workflow stage can be selected by choosing its column in the Graph View of Pivot. As shown in Figure 6.8.c, the remaining items are those, which are related to this workflow stage. Again, to understand the distribution of different comment types about this stage, items can be visualized based on their comment types as shown in Figure 6.8.d. This suggested that there were several complaints (items in red) about this stage of the workflow. By further filtering the comments, which are labeled as bug, complaint, and help request and showing them based on the test subjects of our experiment, as shown in Figure 6.8.e, we found out that six of seven participants in our experiments had made a negative point about this workflow stage. This suggested presence of visible usability problem related to this step. Thanks to the device view in the conceptual model, these comments could be visualized based on the corresponding device features. As it can be seen in Figure 6.8.f and Figure 6.8.g, many of these comments are related to the device power switch. By clicking on any of these items, the property panel opens up in the right side of the screen as shown in Figure 6.8.h. By clicking the provided link in this panel, a video segment, which had been recorded while the user had made the comment was also displayed, Figure 6.8.i. The video is starts from 5 seconds before the comment time and stops after 20 seconds. Interestingly, among the other made comments with respect to this feature, Figure 6.8.j, there is also a suggestion, shown as yellow, which offered a very simple solution for this problem. The user suggested adding a switch power sign, which was truly appreciated by the manufacturer. This power sign has been added in the new version of the product.

6.4. Case Study 2: Real Operating Room

This case study was conducted in winter 2011. Beside the usability testing for the SPECT imaging device, the study was designed to test the functionalities of the OR-Use framework and also to measure the classical usability components for the SPECT imaging device.

This study has been conducted in close collaboration with the manufacturer and surgeons of Klinikum Rechts der Isar.

6.4.1. Objectives

The main objective of this study was to evaluate the subjective satisfaction and learnability of the SPECT imaging device. Beside this the manufacturer wanted to measure and compare the efficiency of device second version prototype, with the preliminary version.

- **Subjective satisfaction:** This refers to how pleasant users find it to use a system. This is an important measure since it helps to improve the user experience in order to increase customer acceptance. It is mainly evaluated based on heuristic feedback collected from test subjects. As opposed to typical scenarios such as websites, where there is one user working with the system, in the OR satisfaction should be evaluated separately for each potential human role. Presenting usability information based on different aspects of the OR domain, e.g. workflow stages, it would be possible to prioritize the required improvement based on different surgical steps.
- **Learnability per human role:** Learnability can be defined as a measure of the degree to which interaction with a device can be learned quickly and effectively. It can be measured either with time or comparing number of performed interactions of new users to an optimum set performed by an expert user, accomplishing the very same task. In collaborative environments such as OR, this should be measured for each individual role. Smooth integration and reduced training cost are among the main benefits of a learnable system.
- **Cross configurations efficiency comparison:** In context of iterative evaluations, it is very important to compare the efficiency of the two successive versions. A system is called efficient when a user can use it productively with a minimum amount of resources such as time. One of the most common techniques for evaluating efficiency is measuring the task completion time. This may be achieved using instances running on different locations within multi-center studies.

6.4.2. Study Design

This study was performed with experts and surgeons in 12 user studies and 7 surgeries, who were selected with various levels of expertise, in Klinikum Rechts der Isar. The OR-Use framework, introduced in Chapter 4 was used to conduct these tests and collect all the

usability relevant data. An expert documented all the comments that users made using the on-site annotation tool.

6.4.3. Analysis and Interpretation

Overall, about 2000 user interactions were recorded from the interactions users made with the the SPECT imaging device. Additionally, 163 user feedbacks, annotated with one of the 8 comment types, have been collected using on-site annotation tool and thinking allowed technique. Here we report on some of the results, addressing the defined objectives.

6.4.3.1. Subjective Satisfaction

Subjective satisfaction was evaluated using direct feedbacks received from the users through thinking aloud process. Positive feedbacks and negative feedbacks, such as complaints and help requests, are typical indicators about users satisfaction of a system. Figure 6.9 shows these feedbacks, presented based on surgical workflow, human roles and device states.

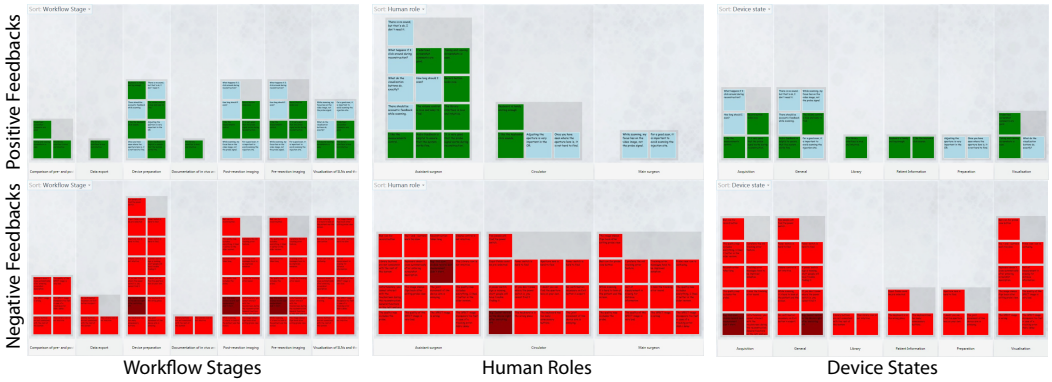


Figure 6.9.: Distribution of collected positive comments (top row) and negative comments (bottom row) over surgical workflow, human roles and device states.

6.4.3.2. Learnability

Learnability per human role has been evaluated by comparing the number of interactions, required to accomplish a given task, between an expert user and new users. This is a common technique for understanding the additional effort that new users require to spend in order to perform a given task. Required time to finish a task can be also computed and compared in the same manner using the OR-Use framework. Figure 6.10 shows these results per human role, distributed over different workflow stages compared to the minimum

interactions, performed by an expert. Several points can be highlighted, e.g. the close results of an expert and new users in stage 6 shows that this stage is much more learnable for surgeons compared to stage 7 where this difference is larger.

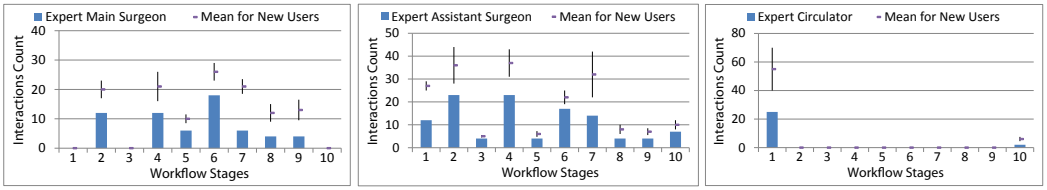


Figure 6.10.: Plots showing number of interactions per workflow stage for OR human roles (comparing expert with test subjects): surgeon, assistant and circulator.

6.4.3.3. Efficiency Comparison

Cross configurations efficiency comparison is performed, computing the task completion time using two different versions of the application. Table 6.1 shows these times (in seconds). In the new version the value for a setting related to visualization was computed automatically, removing the need for manual tuning with the user. Since this feature is used in the visualization stage, this difference is higher in corresponding columns of the table (5 and 6).

	1	**2**	**3**	**4**	**5**	**6**	**7**
Time Using Previous Version	85	142	121	97	257	56	137
Time Using New Version	81	145	122	96	168	31	131
Change in Time	-4	3	1	-1	-89	-25	-6
Change Ratio to Original Time	5%	2%	1%	1%	35%	45%	4%

Table 6.1.: Task completion times, measured in seconds, for two versions of the device.

6.5. Tools Effectiveness

In this section, the effectiveness of some existing usability support tools which are used in other domains is examined and compared to the OR-Use framework. Table 6.2 demonstrates how well each tool meets the functional requirements, discussed in Chapter 4. The legend used for the table follows: M = Morae[101]; WBT = Web based tool[10]; WBX =

WebXACT[131]; HUA = HUIA framework[7]; WUP = Web Usability Probe[4]; SAV = SAVE[65]; ORU = OR-Use.

Phase	Criteria	M	WBT	WBX	HUA	WUP	SAV	ORU
Planning	Workflow/Task modeling	x	x	x	x	x	x	o
	Multiple roles	x	x	x	x	x	x	✓
	Device model	x	x	o	✓	o	x	o
Conducting	Performance logging	✓	✓	x	✓	✓	✓	✓
	Observerś annotations	✓	x	x	x	x	✓	✓
	Video & supplementary materials	x	x	x	x	x	✓	✓
	Configuration management	o	o	o	o	o	o	✓
Analysis	Data retrieval	✓	✓	✓	✓	✓	✓	✓
	Video indexing	x	x	x	x	x	✓	✓
	Visualization	✓	✓	o	✓	o	o	✓

Table 6.2.: Effectiveness of existing tools. Legend: ✓= Implemented, o = Achieved to some extent, x = Not supported; M = Morae, WBT = Web based tool, WBX = WebXACT, HUA = HUIA, WUP = Web Usability Probe, SAV = SAVE, ORU = OR-Use.

6.6. Usability of On-site Annotation Tool

As mentioned before, the on-site annotation tool should be usable for the evaluators. In order to evaluate its usability, we conducted an additional user study with 12 participants (biomedical students aged 22 to 32). After performing a set of tasks based on real activities that a usability specialist should perform during an operation, users were asked to fill three AttrakDiff [6] questionnaires. One about profile management and workflow follow up functionalities and two about the two alternate methods, one for keyboard-based and one for voice-based methods for documenting the user feedbacks.

The results, shown in Figure 6.11, which place this tool in terms of pragmatic quality (PQ) and hedonic quality (HQ). PQ addresses different aspects of human needs and usability factors related to control, learnability and ease of use. HQ deals with human desires for excitement, including novelty and satisfaction. The profile management and workflow follow-up is categorized as “Task-Oriented”. High PQ value means that these features have high usability factors. The large confidence area in both dimensions highlights the fact that users had diverse ideas about these features. Furthermore, keyboard-based and voice-based feedback documentation methods have been compared, where the latter is rated as more usable and interesting for intra-operative usability studies.

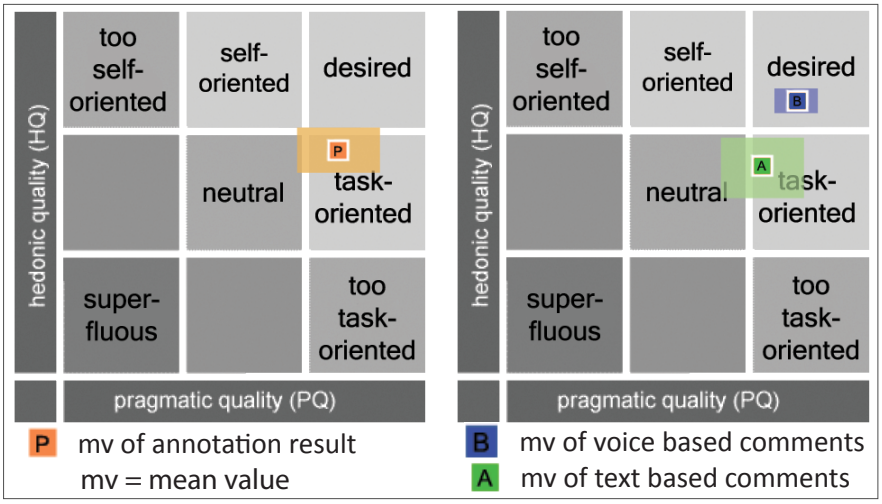


Figure 6.11.: PQ-HQ reports from the user studies on Portable Annotation Tool.

6.7. Performance Evaluations

In this section, we aim at measuring the performance of different parts of the OR-Use framework in practice. This information can be helpful for planning new usability studies using this framework.

6.7.1. Usability Data Growth

In order to evaluate the growth of the data on the server side, we performed a test using the SPECT imaging device, which took 19 minutes. The experiment was conducted by two test persons, whereupon one acted as a main surgeon who interacts with the device, and the other one operated the on-site annotation tool.

6.7.1.1. Over Time

The chart in Figure 6.12 represents an experiment over 19 minutes where the number of user interactions, screenshots and videos gathered by the medical device are counted. Additionally, annotations and associated audio files are recorded by an observer who operates the on-site annotation tool. As it can be seen in Figure 6.12, the user interactions and screenshots have a similar linear course, which refers to the coherency of creating screenshots manually and the resulting user interaction to accomplish this task. One has to bear in mind that the SPECT imaging system stores automatically screenshots during the time it

is running, so this is the reason why these utilize the most space during our test scenarios.

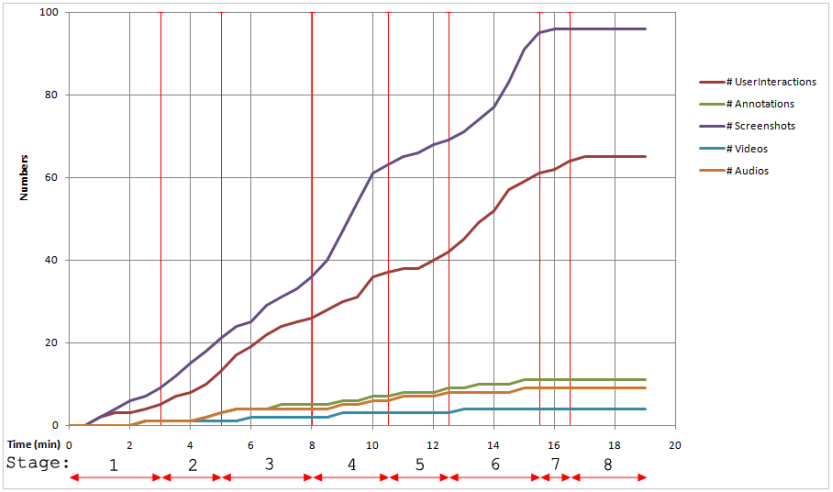


Figure 6.12.: 19 minutes test scenario of a surgical intervention using SPECT. The red lines designate the beginning and ending of a specific workflow stage.

In the first stage, the patient and the device get prepared for the surgery (e.g. target positioning, adding patient information to the system, etc.), so there are usually only few usability annotations because this stage does not comprise difficult task. During the visualization of the pre-scan (stage 3), surgeons observe the area of interest before they start the resection in order to have a reference value of the patients condition before and after the intervention. Therefore, this stage usually includes several screenshots, videos and user interactions. The workflow stages 4 and 6 (resection and visualization of post-scan) show the largest changes in the chart, since surgeons most commonly use the device in these particular phases. Ordinary, the resection phase is the most time consuming stage for the defined workflows, however we simulated this period over only two minutes to present significant results. During the visualization of the post scan, surgeons commonly overlay the images before and after the intervention for interpreting the results. Since the last two stages are related to the autonomous data export and dismantling of the device, we expect few usability relevant data in these phases.

6.7.1.2. Over Workflow Stages

Figure 6.13 illustrates the number and size of gathered files during the defined workflow stages for the 19 minutes test scenario. It is obvious that the number of files and the size have the same growth speed over the entire test phase. As it can be seen in Figure 6.13,

phase 3 and 6 (visualization of pre-scan and visualization of post scan) contain the majority of data collected during the entire test scenario. During these two phases, surgeons are frequently using the medical device to make screenshots or videos of the patients condition before and after the resection. The characteristic curves of Figure 6.13 shows that phases 7 and 8 do not entail much data increase.

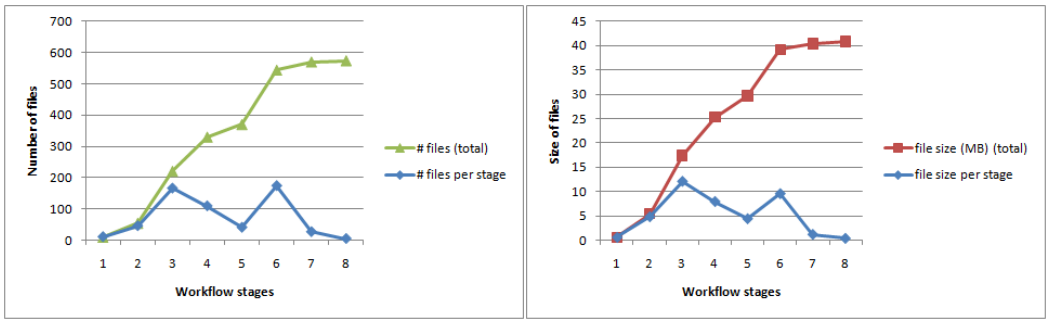


Figure 6.13.: Number (left) and size (right) of usability data over the workflow stages.

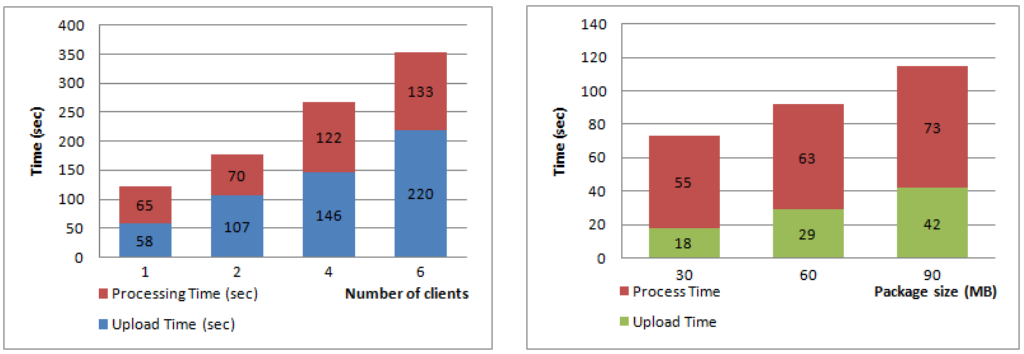
6.7.2. Upload and Processing Time

With the proposed architecture, multiple clients can simultaneously access and consume resources of the server. In order to evaluate the performance of the proposed client-server architecture, we have measured the uploading and processing time with presence of different number of clients and varying data load. The following sections present various performance test of the OR-Use framework to give an impression of the system performance. It has to be taken in mind that the bandwidth connectivity for uploading the data to the server, plays an important role. During these test sessions all files were uploaded from the same broadband connection.

6.7.2.1. Number of Clients

During this performance test, one package with a fixed size of 67.9 MB was sent to the server from 1, 2, 4 or 6 clients at the same time. The results are four charts, which represent the time that was required to upload the fixed package size from varying amount of participating clients and further more calculate the process time on the server side. In Figure 6.14.a it can be seen that the upload and processing time increase with the number of clients. However, the upload time depends on the speed of available broadband connectivity but this does not affect the processing time, which is regulated by the server.

Figure 6.14.a shows that the processing has only minor changes for multiple clients, which highlights the importance of connection bandwidth.



(a) (b)

Figure 6.14.: (a) Number of clients per uploading and processing time, (b) sending file packages with different size in a given time.

6.7.2.2. Files Size

The goal of this examination was to measure the time for sending usability data packages to the server, which vary in the size. As it can be seen in Figure 6.14.b, the test was conducted with a package size of 30, 60 and 90 MB. Again, the upload time depends on the internet connection speed, however this does not effect the processing rate. Despite a doubling and tripling of the file size, the process time increases with a constant factor. This can be very utile for packages with very big size, because the user do not have to wait very long after uploading to the server.

6.7.2.3. Files Count

Figure 6.15 demonstrates the performance of the OR-Use framework when uploading a fixed file package of 10 MB, containing different number of files. Obviously, the upload time is low for a small number of files, however the usability data of the SPECT imaging device usually contains more than 500 files and thus the upload time increases by a considerable factor.

The presented results demonstrate that by increasing the size of data and the number of clients, the processing time does not change as much as the uploading time. This means

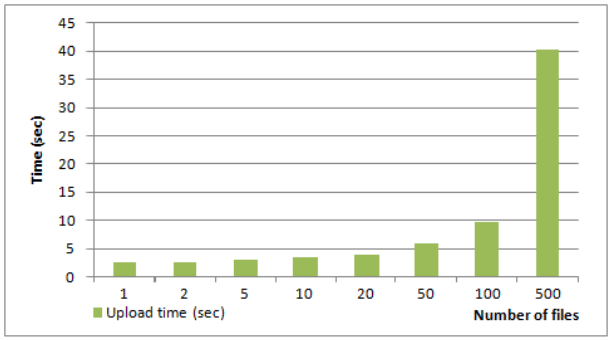


Figure 6.15.: Measuring the time for sending a data package with a fixed size of 10 MB with a varying number of files.

that the main bottleneck of the whole process is the uploading phase and it highlights the importance of the bandwidth in a larger setup.

6.8. Summery and Conclusion

The case studies and experiments presented in this chapter demonstrate that the proposed conceptual model for the OR can be successfully applied during the evaluation of intra-operative devices. This model facilitates different steps of usability testing such as conducting and analysis. The integration of this model in the OR-Use framework made it possible to manage usability data in a centralized web service. The usability of the proposed annotation tool also has been approved. Our user studies revealed that voice based annotation is more usable for evaluators. However, usage of such techniques strictly depends on the condition of the OR and requires permission of the surgeons. The analysis and interpretation steps were highly simplified, thanks to the OR domain model. The model views allowed us to investigate the collected comments in the context of each complexity source of the OR domain. This helped us to discover the potential impact of identified problems. In combination with a powerful visualization technique, such as offered by Pivot, our model enabled us to present usability problems and their underlying cause in a convincing manner to the development team of the SPECT imaging device. Furthermore, we were able to perform most of the evaluation steps without assistance of domain or system experts, as a lot of domain knowledge could be retrieved from the model itself.

Part III.

Intra-operative Human Computer Interaction Design

CHAPTER 7

Categorical and Spatio-Teporal Gesturing

STERILITY requirements and interventional workflow in the Operating Room often make it challenging for surgeons to interact with computerised systems. As a solution, in this chapter, we are aiming at development of gesture-based interaction techniques for the Operating Room. This approaches allow surgeons to define a personalized set of gestures for controlling arbitrary medical computerized systems. We introduce two methods for learning such gestures and explain their usages with two possible input modalities, inertial sensors and Kinect.

7.1. Introduction

With the recent advances of computer-based systems, human-machine interaction has become an even more essential and important issue. However, interaction with computer-based medical devices in the Operating Room is often challenging for surgeons due to sterility requirements and the complexity of interventional procedures. Typical solutions, such as delegating the interaction task to an assistant, can be inefficient. In this situation gesture-based interaction can be a replacing solution. Since the introduction of multi-touch technology, 2D gesture-based interaction methods have been successfully utilized in a wide range of commercially available products, such as tablets and smart phones. In these scenarios, information such as the number of touches, their location and the amount of exerted pressure are extracted, while users interact with a touch-aware surface. In addition to these developments, many studies have been conducted on 3D interfaces, utilizing

various data modalities, such as wearable inertial sensors [123] and time of flight (ToF) cameras [49]. Although all these studies further highlight the need for a touch-less 3D interaction method in different domains, such as Operating Rooms [123, 47, 82, 48], robotics [107] and entertainment [29], only very few products have hit the mass market. Different reasons can explain this: the need for a complex and expensive hardware setup to capture human motion; the social acceptance of performing gestures to interact with a computer-based system; or the challenges associated with recognizing and interpreting gestures as well as hard coding their relation to the target functionalities and commands of the system.

In this work, we target the latter problem by proposing two gesture recognition methods which can facilitate the development of 3D gesture-based interfaces using different input modalities such as inertial sensors or Microsoft Kinect. Our gesture recognition methods build upon machine learning techniques that provide simultaneous categorical and spatio-temporal control. In other words, they recognize the gesture type and the relative pose within a gesture at the same time. These methods learn a set of gestures from training data recorded with a modality of choice, such as body-worn inertial sensors or the Kinect. Internally, dimensionality-reduced models of the gestures are learned. After training, the type of gesture is automatically recognized (categorical control), as well as the relative body pose within this gesture (spatio-temporal control). We hypothesis that using these two types of information, users can simultaneously select features to be controlled and fine-tune continuous parameters with their gestures. We compare using Principal Component Analysis (PCA) and a manifold learning technique for dimensionality reduction. These methods allow surgeons to interact with medical systems by means of gestures. Based on the circumstances and the workflow of a particular interventional scenario, a surgeon can define a set of gestures that are most suitable. To avoid the user-specific training, we also propose several types of feature extraction methods for skeletal model.

7.1.1. The User Interface

When producing an interactive computer-based product, designers have to think beyond the functionality the system needs to provide. They have to take into account the interaction that occurs between the user and the system, and therefore must consider the 'user interface' of the product. This term has been defined by Moran as 'those aspects of the system that the user comes in contact with' [102], which means 'an input language for the user, an output language for the machine, and a protocol for interaction' [31]. A user interface includes both hardware and software components of a product. It should be designed in a way that makes it easy, efficient and enjoyable to operate a machine in the way which

produces the desired results. This means that the operator needs to provide minimal input to achieve the designated output, and also that the machine minimizes inadvertent outputs to the human. The input of a machine allows the user to manipulate a system, in contrast to the output, which admits the system to indicate the effects of the users' manipulation.

7.1.2. Customization Requirements

In order to smoothly integrate a gesture-based user interface into a Operating Room, the application may provide some customization possibilities. In case of gesturing interface, this means that the user should be provided with a functionality to change movements, defining gestures. Several authors emphasize the inter-person variability of human gestures and propose methods that adapt to person-specific variations in performing a given set of gestures [93]. This is due to the fact that selecting an appropriate subset of gestures can be dependent on the usage context. Usually this kind of knowledge is highly application-specific and is not available in advance. For instance, in the OR domain, there are situations in which the surgeon requires to access some settings of an intra-operative device with just one of his hands, since the other hand is occupied holding a medical tool.

The following medical application scenario demonstrate the importance of the customization in the Operating Room. During minimally-invasive aortic repair procedures, a stent graft is inserted through an artery into the aorta to exclude an aneurysm from blood circulation. Positioning the stent graft is guided by imaging modalities that combine intra-operative X-ray views and tomographic patient scans [34]. For the complex task of inserting the stent graft, a hypothesized surgeon would like to control the visualization himself, instead of instructing an assistant. Knowing that his right hand is occupied during catheter insertion, the surgeon trains the gesture recognition system on three gestures he performs with his left arm. He uses a circular movement for browsing slices in the tomographic scan, a vertical movement for zooming and a horizontal movement for translating the image. During an actual intervention, the surgeon needs to adjust the visualization while inserting the stent graft. To activate gesture recognition, he pronounces a voice command and starts tracing a circle with his left arm. The system immediately recognizes the intended browsing functionality. Having found the region of interest, the surgeon keeps his arm still and moves it back and forth for fine adjustment. To increase the zoom factor, he then moves his arm horizontally. The image is scaled in proportion to his arm movement. Another voice command stores the current setting and deactivates gesture control. The surgeon can now continue inserting the stent graft without accidentally modifying the visualization with his movements.

7.2. Related Work

Gesture-based human-computer interaction has been studied intensively and several surveys are available e.g. [75, 100]. Different sensors and modalities are used to acquire gestures, e.g. pens [144], inertial sensors [123] and cameras. Previous research on gesture recognition using wearable inertial sensors includes [83, 58, 150, 132, 41]. In [83], short hand gestures are measured using accelerometers and evaluated for their usefulness in controlling home entertainment devices. In [150] the authors fuse accelerometer and electromyogram signals and recognize hand gestures using Multi-stream Hidden Markov Models. The authors present an HMM-based gesture recognition approach. Hand gestures, such as quick up-down movements or circles, are recognized in [58] based on a Dynamic Time Warping (DTW) mechanism. The authors of [132] and [41] recognize the suitability of *orientation* sensors for gesture recognition, as opposed to pure accelerometers. After its introduction, the Microsoft Kinect has been used in many studies for different applications. In [66], the authors propose a virtual 3D keyboard where depth information is used to extract a users hand and their movements. For active cinema also a gesture-based interface has been introduced in [29]. Byung et al. in [149] propose a remote interface for smart displays. Multimodal interfaces utilizing Kinect have been also studied in several works [69]. Different methods such as HMMs [147], PCA and Kalman filtering [140] have been effectively employed in modeling human gestures.

The idea of gesturing in the Operating Room also has been targeted by several authors, e.g. [84, 50, 129, 139]. For instance, a stereo camera setup is used in [50] for tracking the hand of a surgeon and thereby controlling the mouse pointer on a screen. In [84], a medical image viewer can be controlled by hand gestures, tracked with two video cameras. Recent Time-of-Flight cameras, measuring the depth of a scene in realtime, are employed in [129] for a similar purpose. Such vision-based systems face challenges in realistic OR environments, where optimal lighting and line-of-sight between the surgeon and the cameras cannot always be ensured.

The main contributions of our proposed gesture recognition techniques, compared to the aforementioned methods, can be summarized as follows:

- Gestures can be defined freely according to the requirements of a particular interventional scenario. By demonstrating an example per gesture to the system, surgeons do not have to adhere to a pre-defined set of movements that might interfere with their hand usage during surgery.

- Gestures are not only recognized for triggering discrete actions, e.g. ”click the

mouse" (*categorical* control [56, 83]), but simultaneously allow tracking the movements within a gesture for fine-tuning continuous parameters, such as the size of a displayed image (*spatio-temporal* control).

- Concepts introduced in our techniques can be generalized to facilitate gesture recognition with different types of input data. In this chapter we show how such technique can be used with data received from inertial sensors and Kinect.
- Our gesture recognition method for Kinect can perform independent of the body proportions of individual users. This eliminates the need for training and keeps it as an optional feature.

7.3. Simultaneous Categorical and Spatio-Temporal Gestures

In this section, we explain the novel concept of categorical and spatio-temporal properties of our gesturing interfaces. In order to do so, we first explain these concepts using examples in two dimensional interfaces. In model view controller (MVC) applications, users need to select or activate a proper controller or tool, in order to modify the data as desired. The selected or activated controller specifies the changes which would be applied on the data, based on the upcoming user interactions with the views. For example, in a drawing application on WIMP (Windows, Icons, Menu, Pointer) environments with a single pointing device, a corresponding tool needs to be activated in order to draw a line or rectangle. This can be achieved either using shortcuts or by clicking on a representative toggle button before the real drawing action. However, in multi-touch screens, this process can be simplified for the users, since based on the number of touch spots the targeted controller can be automatically inferred. As an example, in a map viewer application, the presence of one or two touching spots distinguishes between panning and zooming controllers. In other words, in multi-touch environments, gestures can provide both *categorical* and *spatio-temporal* information. Using the categorical property of a gesture, the type of the user action can be determined, e.g. pan or zoom. On the other hand, the spatio-temporal information conveyed by the gesture, such as the location of the touched spot or the distance between two touched spots, provides the controller with the information required to update the targeted view or model accordingly. Inside the application, these two streams of data, the gesture type and the gesture state, can be used accordingly for activating and updating the target controller.

Similar notions apply in many 3D gesture-based interfaces. For instance, in [48] the authors propose different gestures in order to interact with a medical image viewer ap-

plication and several movements have been defined in [21] for interacting with Google Earth. However, in most of the proposed works, recognition of gesture types and other calculations for extracting required spatio-temporal data from the acquired skeletal information needs to be hard-coded during development. Although for 2D gestures, this kind of information may be achieved quite easily, using a 3D skeletal model it is much more complicated. Besides all the challenges associated with such implementations, the proposed methods are lacking extensibility, as they are just applicable in the given scenario. Furthermore, such techniques are not customizable and it is hard to change gestures at run-time. This is due to the face that they are compiled into the application binary. As opposed to the aforementioned gesture-based user interfaces, our gesture recognition methods learn gestures based on sets of body poses defined by the user in a training phase. At runtime, the recognition method detects the index of the current gesture as well as an activity state specifying position of the user within that gesture. As a result, this information can be used seamlessly to activate the proper controller and change the parameters of interest. Unlike typical gesture-based user interfaces that either give users *categorical* or *spatio-temporal* control [56, 83], our method combines both concepts. Surgeons can select discrete functions of the target system, as their gestures are automatically recognized and classified (*categorical* control). At the same time, their particular arm movements in space are used to precisely adjust continuous parameters related to the selected functions (*spatio-temporal* control). Altough a similar concept has been introduced and studied for 2D gestures on touch-based interfaces [54, 112], this notion is new for 3D gestures. We can thus use gestures not only to issue commands, but also for finer-graind control of particular, related parameters. Consider the scenario of intra-operative navigation in a set of fluoroscopy images [81]. The physician can choose a planar arm movement for browsing through the images in temporal sequence and a circular movement for zooming in and out. Besides triggering these functions with the respective gestures, the user can adjust the temporal and scaling parameters with his relative body pose.

7.4. Input Modalities

There are different input sensors to acquire information about the users movement such Kinect, inertial sensors, depth cameras and finger tracking gloves. As mentioned before, our gesture recognition techniques are not dependent to a specific input modality. Within this work we consider two such input modalities, inertial sensors and Microsoft Kinect. The following sections introduce these modalities.

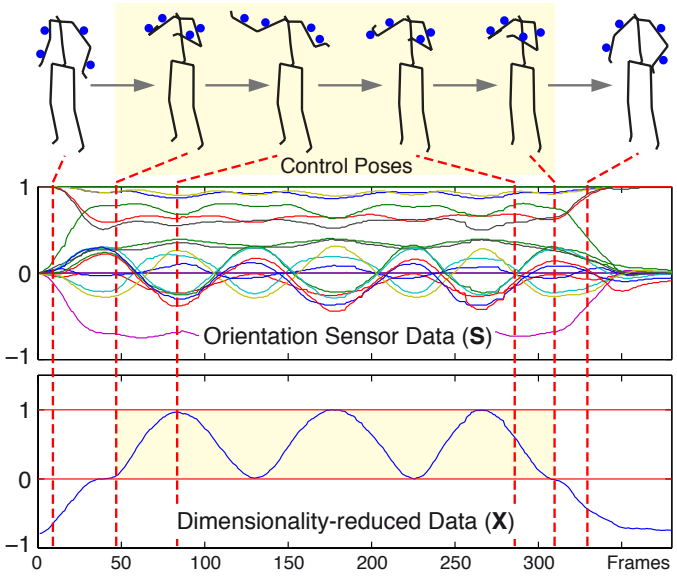


Figure 7.1.: Selected body poses for a zooming gesture with indicated inertial sensors (*top*), highlighted in a stream of orientation sensor data (*middle*) and the corresponding dimensionality-reduced signal (*bottom*). The resulting one-dimensional value between 0 and 1 serves to control arbitrary parameter values.

7.4.1. Inertial Orientation Sensors

Inertial sensors can measure their orientation with respect to the earth reference frame. Sensors of this type are internally comprised of an accelerometer, a gyroscope and a compass that are combined by means of sensor fusion [41]. As opposed to pure accelerometers, orientation sensors exhibit relatively low levels of noise and drift. Orientation sensors are particularly suitable for gesture-based human-computer interaction, as different body poses generally result in different sensor measurements (see Figure 7.1). Pure accelerometers depend on body movement for extraction of characteristic signal patterns. In our implementation, we used wireless orientation sensors that are approximately 2cm $\times$ 2cm $\times$ 3cm of size and allow for 8-10 hours of battery-based, wireless operation. We use the quaternion values provided by each of the sensors, avoiding problems of other representations, such as gimbal lock.

Gestures can be captured using a few wireless inertial sensors on the surgeon's arms that can easily be integrated into garment, as shown in Figure 7.2. This approach makes the gesturing interface independent of cameras, enabling robust gesture recognition in the

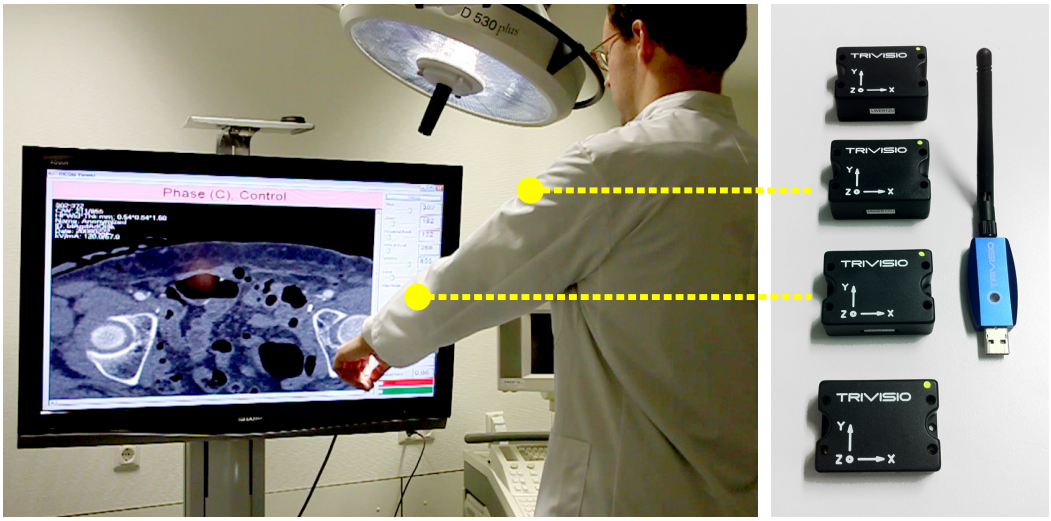


Figure 7.2.: Gesture-based control of a medical image viewer application. Four wireless inertial sensors (*top right*) are placed on the person's arms underneath the blouse. In this example, a vertical arm gesture is used to browse the slices in a volumetric dataset.

presence of critical OR conditions, including difficult lighting, cluttered backgrounds and occlusions by staff and equipment. Additionally, gestures are recognized regardless of the surgeon's position and orientation. As each inertial sensor can be identified uniquely, gestures can be also assigned to multiple persons, e.g. a surgeon and an assistant, for distributing the interaction workload. Previously, inertial sensors have mainly been used for activity recognition, e.g. [93, 124], and patient monitoring [3].

7.4.2. Microsoft Kinect

Launched in the end of 2010, the Kinect is a motion capturing device that enables users to interact with games and other applications without the traditional need for having a physical connection to a game controller, mouse and keyboard, etc. This is achieved through tracking the user's body and by utilizing gestures and speech recognition [98]. The device is composed of a regular color camera, an infrared light (IR) projector and an IR camera, as well as a microphone array, which can reduce the noise. In order to acquire depth information of the scene, the Kinect projects IR light patterns into the scene and computes the per-pixel distance using the deformation of these patterns in the captured scene, with a dedicated IR camera. Due to the usage of IR light, the Kinect works in all indoor lighting

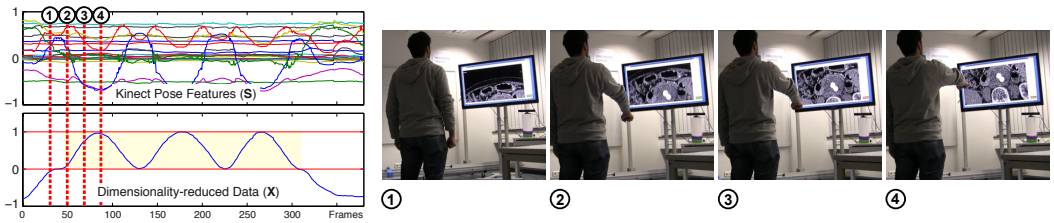


Figure 7.3.: Selected body poses for a scrolling gesture and a medical image viewer as exemplary target system (*right*), highlighted in a stream of Kinect pose features (*left, top*) and the corresponding dimensionality-reduced signal (*left, bottom*). The resulting one-dimensional value between 0 and 1 is mapped to the parameter range of the function to be controlled.

conditions, whether in complete darkness or in a fully lit room. This is an important factor since in some usage contexts like the OR, lighting conditions can vary widely and are not suitable for vision-based capturing techniques [123]. The device can simultaneously track and extract skeletal data for two active users [98]. Figure 7.3 demonstrates the acquired data from Kinect while user is performing a gesture.

7.5. Gesture Recognition Methods

We propose two simple and effective methods for gesture recognition and tracking based on dimensionality reduction approaches. These methods can embed high-dimensional data in an arbitrary lower-dimensional space. Our methods consist of an offline training phase, where gesture models are learned from sensor data for all considered gestures, and an online phase, where the models are used to recognize gestures from previously unseen sensor data. The main idea behind dimensionality reduction approaches is initiated based on the observation that high-dimensional data is often much simpler than the dimensionality would indicate. There are several approaches to dimensionality reduction based on a variety of assumptions and used in a variety of contexts. Our methods do not build upon a particular type of dimensionality reduction technique. However, in this work we use two different approaches. A traditional approach, Principal Component Analysis (PCA), and a relatively new approach called Manifold learning. In the following sections, we first provide a short introduction over used dimensionality reduction techniques and then describe our gesture representation and the gesture recognition methods, consisting of a training and a testing phase.

7.5.1. Modeling Gestures

In this work, a gesture is at the same time categorical and spatio-temporal [56]. A user performs a gesture to invoke a particular function of the target system and to control this function in a fine-grained, continuous manner. The gesture recognition method therefore has two goals:

1. To recognize which gesture a user is performing.
2. To identify the relative spatial pose within that particular gesture.

To this end, we define a gesture to be an arbitrary movement including one or both arms that has a temporal extent and encompasses many smoothly-varying poses. We refer to all poses actually intended for controlling the target system as *control poses* (see Figure 7.3). We assume that *control poses* are preceded and followed by introduction and termination movements, such as raising and lowering an arm, that are not directly used for control. In the example shown in Figure 7.3, a user chooses a horizontal movement of both arms for zooming a virtual object. As soon as he starts raising his hands, the intended gesture is recognized and the zooming function is activated. When the user reaches the *control poses* in front of his body, his exact hand movement is used for controlling the zoom parameter of the virtual object.

As we allow gestures to have a temporal extent, we introduce a simple phase model for our gesturing interface, as shown in Figure 7.4. Each gesturing movement is subdivided into three phases $\{I, E, C\}$. Phase I indicates the beginning and end of a gesture, phase E contains introductory movements, such as raising a hand, and phase C is actually used for controlling a target system. The points $[\mathbf{x}^c_{\min}, \mathbf{x}^c_{\max}]$ indicating the boundaries of phase C, mapped to a minimal and maximal parameter setting, can be defined by the user in the training phase, e.g. by holding the respective poses for several seconds. The additional phase $\emptyset$ represents poses that do not belong to any of the learned gestures. Phase T is used to terminate parameter adjustment.

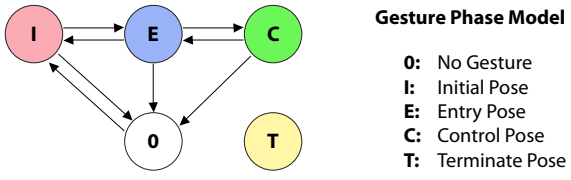


Figure 7.4.: Gesture phase model as is used in this work.

7.5.2. Principle Component Analysis

In this section, we present our gesture recognition technique which has been developed based on PCA. First, we shortly introduce the PCA dimensionality reduction technique and in the coming sections explain how this method can be used to learn gestures during training stage and recognize and track them during runtime.

7.5.2.1. Background

PCA is dimensionality reduction technique which can be utilized to find an optimal subspace in the data, where the variance of data is maximized [128]. It can be categorized as a linear technique as it ignores protrusion or concavity in the input data space [63]. The input and output data for PCA can be defined as $\Psi : \mathbb{R}^D \rightarrow \mathbb{R}^d$. Considering N as the number of input points, the PCA can be computed with the following steps:

1. Compute the empirical mean vectors per input dimension $j \in 1 \cdots D$:
   $\mu[j] = \frac{1}{N} \sum_{i=1}^{N} X[i,j]$.

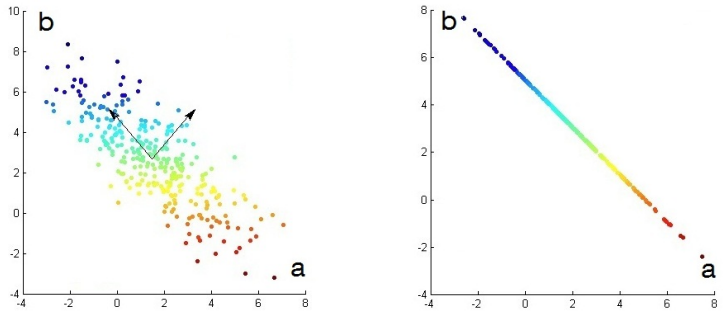
2. Subtract μ $(D \times 1)$ from each column of the $D \times N$ input matrix X. The subtracted matrix $B = X - \mu h$, where h is a $1 \times N$ vector of ones.

3. Calculate the $D \times D$ covariance matrix $C = \frac{1}{N-1} B \cdot B^\top$.

4. Find the eigenvectors to form the matrix V of eigenvectors, so that $V^{-1} \cdot C \cdot V = P$ with P being the diagonal matrix in decreasing order of corresponding eigenvalues. $V^\top = V^{-1}$ as all the eigenvectors are orthogonal and they form an orthonormal basis.

5. Project the data into the new d-dimensional subspace, using the first d columns of V. The value of d is chosen according to some measure such as data energy or highest variance: $Y = [v_1, \ldots, v_d]^\top \cdot X$.

Figure 7.5.a shows a Gaussian distribution, where the first two principal components is computed using the aforementioned algorithm. These vectors are eigenvectors of the matrix C. The data is linearly color coded based on the values of a and b. Figure 7.5.b demonstrates the projection of data points to the eigenvector with the largest eigenvalue. With this technique the variance of data can be preserved.



(a) (b)

Figure 7.5.: An example of PCA. Colors have linear dependency to values of a and b. (a) A Gaussian distribution with the two computed principal components, (b) Projections of data on the eigenvector with the largest eigenvalue.

7.5.2.2. Learning Gestures

In a training phase, our method learns prior models of gestures from input sensor data. Let $\mathbf{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ be a dataset of n sample vectors for N gestures of interest. Each vector $\mathbf{s}_i \in \mathbb{R}^d$ represents a particular body pose. The known gesture labels for the training data are denoted by $\mathbf{C} = \{c_1, \ldots, c_n\}$, with $1 \leq c_i \leq N$. Using PCA for dimensionality reduction, we construct a one-dimensional intermediate representation $\mathbf{X} = \{x_1, \ldots, x_n\}$ from the training features, such that every $x_i \in \mathbb{R}$ corresponds to one $\mathbf{s}_i$. For most effectively capturing the movement characteristics of distinct gestures, we separately process the samples $\mathbf{S}^c = \{\mathbf{s}_i \in \mathbf{S} | c_i = c\}$ for each gesture c. The resulting dimensionality-reduced datasets $\mathbf{X}^c$ are then normalized such that, for each gesture, the *control poses* map to the range $[0, 1]$ (see Figure 7.1). This can be simply achieved by truncating the datasets $\mathbf{X}^c$ at their beginning and end. Concatenating all gesture-specific sets then gives $\mathbf{X} = \{\mathbf{X}^1, \ldots, \mathbf{X}^N\}$. Note that, for every index $1 \leq i \leq n$, the training dataset contains a triplet $(\mathbf{s}_i, c_i, x_i)$ of a sample vector, its gesture label and the corresponding one-dimensional value.

7.5.2.3. Recognizing and Tracking Gestures

The dimensionality-reduced intermediate representation $\mathbf{X}$ allows us to translate movements for the learned gestures into a continuous, one-dimensional signal that can be used for adjusting arbitrary user interface parameters. After training, we are given new, pre-

viously unseen feature vectors $\mathbf{s}_t$ at any time t. Our aim is to determine the intermediate representation $\hat{x}_t \in \mathbb{R}$ corresponding to $\mathbf{s}_t$ and to identify the performed gesture $\hat{c}_t$, $1 \leq \hat{c}_t \leq N$. For this purpose, we define a kernel regression mapping that projects arbitrary feature vectors to the dimensionality-reduced representation $\mathbf{X}$. We predict the value $\hat{x}_t$ for a feature vector $\mathbf{s}_t$ as

$$\hat{x}_t = f(\mathbf{s}_t) = \sum_{i=1}^{n} \frac{w_i(\mathbf{s}_t)}{\sum_{j=1}^{n} w_j(\mathbf{s}_t)} \cdot x_i. \tag{7.1}$$

In this summation, the estimate $\hat{x}_t$ is combined from the known projections $x_i \in \mathbf{X}$ of the feature vectors $\mathbf{s}_i \in \mathbf{S}$ in the training data. Each value x_i is weighted in proportion to the similarity between its corresponding feature vector $\mathbf{s}_i$ and the new measurement $\mathbf{s}_t$. Dividing by the sum of all weights ensures that the summation is a convex combination and $\hat{x}_t$ lies within the range of values in $\mathbf{X}$. The weights are given by a Gaussian kernel $w_i(\mathbf{s}_t) = \mathrm{k}(\mathbf{s}_t, \mathbf{s}_i) = \exp(-\frac{1}{2}\|(\mathbf{s}_t - \mathbf{s}_i)/\sigma\|^2)$ with a width σ derived from the variance of the training features $\mathbf{S}$. Using the weights, we determine the current gesture index $\hat{c}_t$ as

$$\hat{c}_t = c_{k_t}, \quad \text{where} \quad k_t = \arg\max_i w_i(\mathbf{s}_t). \tag{7.2}$$

Here, the index k_t identifies the sampled vector in the training data that is most similar to $\mathbf{s}_t$ and thus has the largest weight. The gesture label corresponding to this training sensor vector is used as an estimate for $\hat{c}_t$. As Equation 7.1 only depends on the current sensor measurement $\mathbf{s}_t$, it does not take into account previous observations or recognized gestures. However, such temporal information can increase the robustness of the gesture recognition method with respect to noise and outlier sensor measurements. We therefore modify the kernel regression-based weighting scheme and replace the weights $w_i(\mathbf{s}_t)$ with

$$\bar{w}_i(\mathbf{s}_t) = w_i(\mathbf{s}_t) \cdot \exp(-\frac{1}{2}((i - k_{t-1})/\sigma)^2). \tag{7.3}$$

In analogy to k_t in Equation 7.2, k_{t-1} identifies the sensor data measurement in the training data with the largest weight in the previous time step $t-1$. Multiplying $w_i(\mathbf{s}_t)$ with the exponential term essentially centers a Gaussian at this sensor measurement. This way, when computing the convex combination in Equation 7.1, sensor vectors around the best-scoring vector at time $t-1$ are favored at time t.

7.5.3. Non-linear Manifold Learning

This section explains our second gesture recognition approach which has been developed based on Manifold learning technique. Manifold learning techniques have shown to provide compact, low-dimensional representations of human motion data [37] and have been

used for human pose tracking from various types of information [74, 124]. In the light of gesture recognition, we combine multiple, gesture-specific manifold models and subdivide the embeddings into phases allowing us to assign particular poses to arbitrary parameter settings. This method also naturally handles the problem of gesture segmentation by means of a predictive confidence measure.

7.5.3.1. Background

Non-linear methods such as Manifold learning in particular, typically are famous on preserving neighbourhood properties within the data. Non-linear manifold learning methods are typically graph-based and can be computed using the following basic steps:

1. Create an undirected similarity graph $G = (V, E)$, where the vertices V are defined by the data points x_i.

2. Predict local properties such as the weight matrix W to define the weighted similarity graph $G = (V, E, W)$. $w_{ij} \geq 0$ represents the weight for the edge connecting vertex i and vertex j. Weights can be computed using a kernel. A weight of 0 demonstrates the fact that the vertices do not have any connection.

3. Define a global embedding Ψ which is optimized and preserves the expected local properties.

Three methods are usually used to build the similarity graph G. The first method is known as the ϵ-neighbourhood graph. This technique connects all vertices with a distance $\|x_i - x_j\|^2$ smaller than a defined ϵ. Graphs created using this method are naturally symmetric [135]. In contrast to this, local connection method uses a similarity function that incorporates local neighbourhood relations, e.g. the Gaussian function: $w_{ij} = exp(-\|\|x_i - x_j\|^2/(2\sigma^2))$. The result graphs using this technique are fully connected graph. As it implicitly defines the weights, this method directly leads to the third step [128]. Finally, the *k*-nearest neighbour (kNN) graphs connect each vertex only to its *k*-nearest neighbours.

In this work, for defining our weight matrix, we use Laplacian Eigenmaps [12]. Laplacian Eigenmaps is based on ideas from graph theory. The key step is to analyse the eigenvector of the Laplacian Graph and use the k-nearest neighbourhood method. We construct the mapping based on the eigenvector corresponding to the smallest eigenvalues. Among the main characteristics of Laplacian Eigenmaps is that they try to preserve distance relations [12]. Additionally, they can be solved by one sparse eigenvalue problem [128]. As

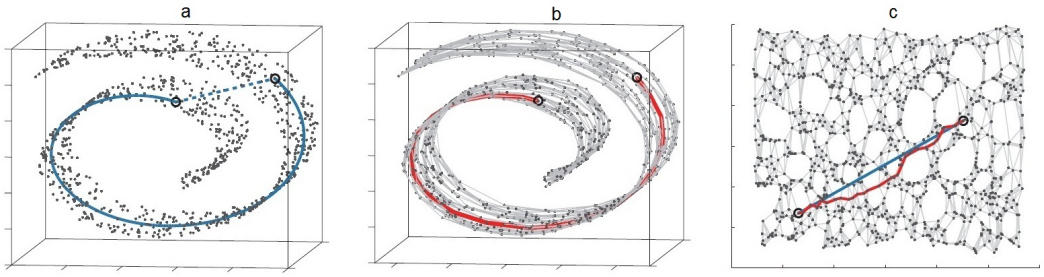


Figure 7.6.: (a) Shows that the Euclidean distance between two points do not reflect their similarity along the manifold, (b) demonstrates the geodesic path calculated in the first step of the Isomap algorithm, (c) displays the two dimensional embedding defined by Isomap [128].

shown in Figure 7.6 a swiss roll is unfold using this methods and the geodesic distances are also preserved. An embedding Ψ can be computed using the following three steps:

1. Define an undirected similarity graph such as $G = (V, E)$.
2. Pick a weighting matrix W by simply setting $W_{ij} = 1$ for all connected vertices. Another approach is to use a heat kernel with parameter $t : w_{ij} = exp(-\|x_i - x_j\|^2 / t)$. In case of a sparse graphs, not fully connected, continue with third step with each connected sub graph.
3. Compute the eigenvectors $v_1, ..., v_n$ and their corresponding eigenvalues $0 = \lambda_1 \leq ... \leq \lambda_n$, utilizing the generalized eigenvalue problem: $L_v = \lambda D v$, where L is laplacian matrix and D is degree matrix (number of edges connected to that node, for every entry ij). Now, the embeding can be defined as: $\Psi : x_i \rightarrow (v_2(i), ..., v_d(i))$.

The Laplacian Eigenmaps, which are a special case of diffusion maps, handle only manifolds which are sampled uniformly. Although such data is not common in many machine learning scenarios, data collected using input sensors from user interactions perfectly matches this requirements. The eigenvectors and eigenvalues of the Laplacian can be used to reveal the information about the graph structure such as its completeness or connectedness.

7.5.3.2. Learning Gestures

Similar to the PCA-based technique, in a learning phase, we learn prior models of gestures from sensor data. Let N be the number of the trained gestures and let $\mathbf{S}^c = [\mathbf{s}_1^c, \ldots, \mathbf{s}_{n_c}^c]$,

$1 \leq c \leq N$, be a dataset of n_c labeled sensor measurements. Each vector $\mathbf{s}_i^c \in \mathbb{R}^{d_s}$ consists of data collected from input sensor e.g. inertial sensors or Kinect. To obtain a compact parametrization of feasible sensor values, we use Laplacian Eigenmaps [12], as introduced in the previous section. In particular, we map the training data $\mathbf{S}^c$ for each gesture c to a low-dimensional representation $\mathbf{X}^c = [\mathbf{x}_1^c, \ldots, \mathbf{x}_{n_c}^c]$, such that $\mathbf{x}_i^c \in \mathbb{R}^{d_x}$ and $d_x \ll d_s$. Figure 7.7 shows exemplary two-dimensional manifold embeddings for three gestures. The crucial property of the manifold embeddings is that the local spatial distribution of vectors in the original, high-dimensional representation is preserved. In particular, similar sensor measurements will map to close-by embedding points, even if they occur at different times within a gesture. In other words, this makes our gesture recognition method invariant to movement speed.

We relate the space of sensor measurements and the low-dimensional manifold embeddings using kernel regression mappings. The mappings allow projecting new sensor values $\mathbf{s}^*$ to points $\hat{\mathbf{x}}$ in embedding space (*out-of-sample mapping*) and predicting sensor vectors $\hat{\mathbf{s}}$ from given embedding points $\mathbf{x}^*$ (*reconstruction mapping*). Following [124, 28], we define the *out-of-sample mapping* for gesture c as $\hat{\mathbf{x}} = f_c(\mathbf{s}^*) = \frac{1}{\phi^c(\mathbf{s}^*)} \sum_{i=1}^{n_c} k_s^c(\mathbf{s}^*, \mathbf{s}_i^c)\mathbf{x}_i^c$, where $\phi^c(\mathbf{s}^*) = \sum_{j=1}^{n_c} k_s^c(\mathbf{s}^*, \mathbf{s}_j^c)$.

We use a Gaussian kernel k_s^c with a width determined from the variance of the training sensor data. The mapping is a weighted average of all manifold embedding points $\mathbf{x}_i^c \in \mathbf{X}^c$, with the largest weights attributed to points projected from sensor values $\mathbf{s}_i^c$ which are

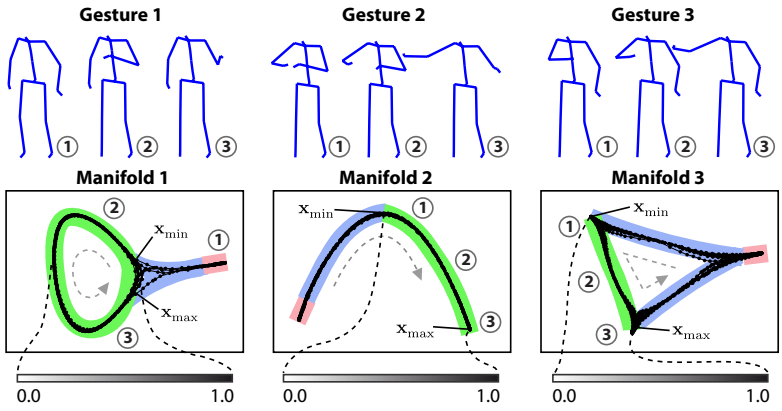


Figure 7.7.: The proposed method recognizes multiple user-defined gestures (*top*) and tracks the relative pose within a gesture by means of learned gesture manifolds (*middle*). The relative gesture pose, given by a value in the interval $[0, 1]$, is used for smooth parameter adjustment (*bottom*).

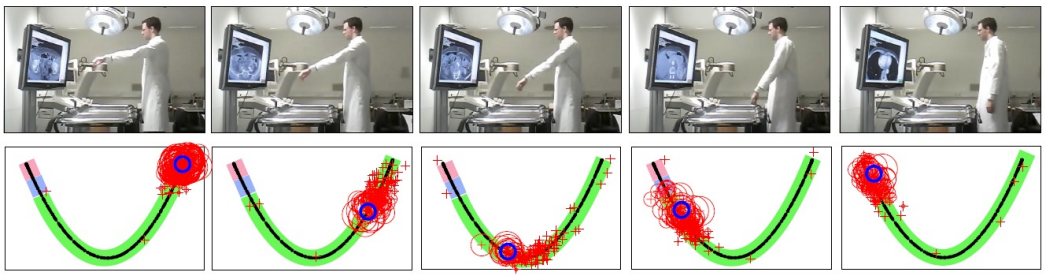


Figure 7.8.: *Top row*: Sample images of a test subject performing one of the learned gestures. *Bottom row*: Manifold embedding for the same gesture with distribution of particles, shown in red, for each of the above images. The point $\hat{\mathbf{x}}_t$ is given by a dark circle.

similar to $\mathbf{s}^*$. By interchanging the roles of sensor values and manifold embedding points, we obtain the *reconstruction mapping* $\hat{\mathbf{s}} = g_c(\mathbf{x}^*)$. That allows us to predict a sensor measurement $\hat{\mathbf{s}}$ from any location $\mathbf{x}^*$ in the manifold embedding of gesture c.

7.5.3.3. Recognizing and Tracking Gestures

After training, any pose corresponding to one of the learned gestures can be represented as a pair $(c, \mathbf{x})$, where c identifies one of the N manifold embeddings and $\mathbf{x}$ is a point in that embedding. Every point in the learned manifold embeddings represents a particular pose within one gesture. Using this notion, we represent the system state at any time t as a pair $(c_t, \mathbf{x}_t)$, where c_t identifies one of the N manifold embeddings and $\mathbf{x}_t$ is a point in that embedding. This state representation allows us to distinguish between different gestures and to determine the relative pose within a gesture.

In the runtime phase, our aim is to estimate $(\hat{c}_t, \hat{\mathbf{x}}_t)$ at each time t, given only sensor measurements $\mathbf{s}_t$. We employ a particle filter [70] that continuously explores the gesture manifolds to find the manifold index $\hat{c}_t$ and point $\hat{\mathbf{x}}_t$ that best explain the sensor measurements $\mathbf{s}_t$ at any time t. We follow a generative tracking approach where a state space (here: the manifold embeddings) is continuously explored to find the parameters that best fit current observations. The manifold embeddings provide us with a low-dimensional state space representation that we explore using a particle filter [70, 71].

Every particle $\mathbf{p}_t^i = (c_t^i, \mathbf{x}_t^i)$, $1 \leq i \leq n$, represents one pose hypothesis of a manifold index and a corresponding embedding point, where c_t^i identifies a manifold embedding and $\mathbf{x}_t^i$ is a point within that embedding. Initially, all n particles are randomly distributed across the manifold embeddings. In every iteration of the particle filter, the particles are propa-

gated through the manifold embeddings, ensuring that only positions close to the learned embedding points are sampled (see Figure 7.8), followed by a resampling step where the best particles are favored. The particle filter is an iterative Bayesian state estimation technique that maintains a large number of state hypotheses (i.e. particles) [70, 71]. With a certain probability, particles are allowed to switch between different manifold embeddings. We model this probability to be high in embedding space regions that correspond to an idle pose separating gestures. After switching to another gesture manifold, particles are randomly placed close to the respective learned points, thus ensuring that only meaningful states are considered. To evaluate the fitness of a particle, we define the observation likelihood $p(\mathbf{s}_t|c_t^i, \mathbf{x}_t^i) \propto \mathcal{N}(g_{c_t^i}(\mathbf{x}_t^i); \mathbf{s}_t, \mathrm{cov}(\mathbf{S}^{c_t^i}))\, \mathcal{N}(f_{c_t^i}(\mathbf{s}_t); \mathbf{x}_t^i, \mathrm{cov}(\mathbf{X}^{c_t^i}))$. The first normal distribution is centered around the observation $\mathbf{s}_t$, giving a high weight to a particle if the sensor value predicted from its position $\mathbf{x}_t^i$ is close to $\mathbf{s}_t$. In the second normal distribution, centered around $\mathbf{x}_t^i$, particles are favored that are close to the projection of $\mathbf{s}_t$ into embedding space.The final estimated gesture index $\hat{c}_t$ is selected as the most frequent index c_t^i among the best-scoring particles. Among these particles, those with $c_t^i = \hat{c}_t$ are used to compute the final position $\hat{\mathbf{x}}_t$ in embedding space.

7.5.3.4. Temporal Gesture Segmentation

Having estimated $\hat{c}_t$ and $\hat{\mathbf{x}}_t$, we determine the corresponding phase in our gesture phase model (Figure 7.4). Phase $\emptyset$, indicating an unknown (or non-gesture) movement, is activated when the prediction confidence $\lambda_t^{\emptyset} = -\log(k_s^{\hat{c}_t}(\mathbf{s}_t, g_{\hat{c}_t}(\hat{\mathbf{x}}_t)))$ falls below a preset threshold. This measure evaluates how closely the estimated state $\hat{\mathbf{x}}_t$, projected to sensor space, matches the true sensor observation $\mathbf{s}_t$. When in phase $\emptyset$, the gesture prediction will likely be incorrect and can be disregarded. Note that our phase model permits transitions into phase $\emptyset$ from any other phase, allowing the user to exit gesture recognition at any time. To identify the *initial* phase I, and thus the beginning of a gesture, we define $\lambda_t^I = k_s^{\hat{c}_t}(\mathbf{s}_t, \mathbf{s}_0)$, which evaluates the similarity between the sensor measurements $\mathbf{s}_t$ and the idle pose $\mathbf{s}_0$ used for separating gestures. While λ_t^I is above a preset threshold, we assume the idle pose is taken, implying the onset of a gesture. If neither of the phases $\{\emptyset, I\}$ are active, we assume the current phase is one of $\{E, C\}$ and compute a relative pose value $\hat{a}_t \in [0, 1]$ from the manifold position $\hat{\mathbf{x}}_t$ (see Figure 7.7). To this end, we transfer the Cartesian coordinate $\hat{\mathbf{x}}_t$ into a polar representation (r_t, θ_t), such that the pole is at the centroid of the embedding points $\mathbf{x}_i^{\hat{c}_t}$, and keep the angular component θ_t. Since the angular representation $[\theta_{\text{min}}^c, \theta_{\text{max}}^c]$ of the points $[\mathbf{x}_{\text{min}}^c, \mathbf{x}_{\text{max}}^c]$ labeled in the training phase is

known, we can compute the desired relative pose value as

$$\hat{a}_t = \begin{cases} (\theta_t - \theta_{\text{min}}^{\hat{c}_t})/(\theta_{\text{max}}^{\hat{c}_t} - \theta_{\text{min}}^{\hat{c}_t}) & \text{if} \quad \theta_{\text{min}}^{\hat{c}_t} \leq \theta_t \leq \theta_{\text{max}}^{\hat{c}_t}, \\ 0 & \text{otherwise.} \end{cases} \tag{7.4}$$

We define phase C to be active when $\hat{a}_t \neq 0$. By changing the pose within the boundaries of phase C, the user can fine-tune parameters. When a suitable parameter setting has been found, the user can trigger a transition from phase C to the *termination* phase T. The current parameter value is then stored and movements are ignored for a certain amount of time, allowing the user to return to an idle pose.

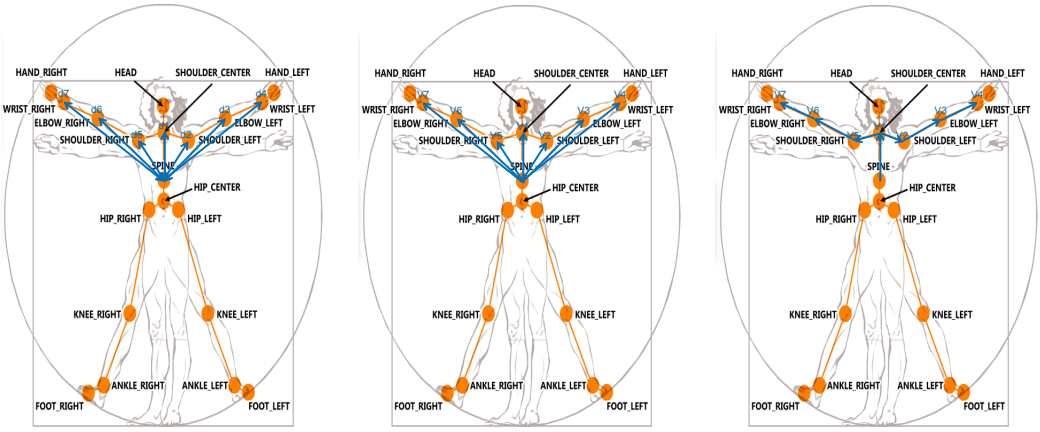
7.6. Sensor Data and Feature Extraction

In general, the performance of a recognition system primarily depends on defining suitable features to represent the input data. In case of the inertial sensor, the aforementioned approaches can be used directly, where each sample of the data is a vector consisting of all the values extracted from the inertial sensors. However, for Kinect other representations of data should be considered to extend the functional domain of the system. For example, having well-defined representation of the skeletal data makes the gesture-based interface independent from the users location and orientation as well as the placement of the Kinect sensor itself in the scene. Furthermore, defining features can help to ignore unwanted confusion originated from noise or other irrelevant parts of the data. As an example in our user studies, the upper part of the body has been used for gesturing and therefore any additional movements in the legs can be filtered using a proper feature set, capturing only relevant movements. In this section we propose three different feature representation approaches as well as two normalization techniques.

7.6.1. Feature Representations

The original human body model employed by the Kinect consists of a set of limbs connected through joints. Each joint has a unique identifier that characterizes the corresponding body part. This data includes $M = 20$ joints for spine, head, left hand, left wrist, left elbow, left shoulder etc. At each time instant t, every joint is represented as a point $\mathbf{p}_i^t = [x, y, z]^\top \in \mathbb{R}^3$, where $i \in \{1, \ldots, M\}$. The origin of this coordinate system is the Kinect device and the unit is meters. Additionally, each joint contains a confidence value λ_i^t which shows the confidence of the tracking system about the joint location. Here, we propose three different ways of extracting feature vectors $\mathbf{s}_t$ from the joint location data

that are particularly suitable for the proposed gesture-based interaction methods. We will compare these features in Chapter 9.



(a) (b) (c)

Figure 7.9.: Three different feature representations for Kinect, (a) distance representation, (b) displacement representation, and (c) hierarchical representation.

- **Distances:** In this type of features, we represent each joint by its distance to the center of the human body. We simply assume that the spine joint $\mathbf{p}^t_{\text{spine}}$ is an estimation for the center of the skeletal data. Therefore, the distance-based feature vector at time t is $\mathbf{s}_t = [d^t_1, \ldots, d^t_M]^\top \in \mathbb{R}^M$, where $d^t_i = \|\mathbf{p}^t_i - \mathbf{p}^t_{\text{spine}}\|$ for all joints including the spine itself. This feature representation is independent of the camera or user location and orientation. Figure 7.9.a demonstrates a visual representation of this feature type.

- **Displacements:** Similar to the distance features, we also consider the spine as the origin of the coordinate system for the displacement features. However, for each joint $\mathbf{p}^t_i$, a displacement vector $\mathbf{d}^t_i = \mathbf{p}^t_i - \mathbf{p}^t_{\text{spine}}$ is defined, resulting in a feature vector $\mathbf{s}_t = [(\mathbf{d}^t_1)^\top, \ldots, (\mathbf{d}^t_M)^\top]^\top \in \mathbb{R}^{3M}$. Compared to the distances, the displacements contain more motion information and consequently the learning method can decide more easily based on this additional data. As a downside, the outcome might be more sensitive to the way gestures are performed by users in space. Figure 7.9.b shows how this features can be defined for a given body pose.

- **Hierarchical:** In order to investigate the possibility of reducing the mentioned

sensitivity, we propose an additional, hierarchical feature representation. Similar to the displacements, in this model each joint $\mathbf{p}_i^t$ is represented with a vector $\mathbf{h}_i^t$. However, unlike previously, this vector is originated at the parent $u(\mathbf{p}_i^t)$ of each joint. We consider the skeletal body model as a tree where joints are nodes and the spine joint is the root. For instance, the shoulder joints are the two children of the spine, which is their parent joint. The feature vector thus becomes $\mathbf{s}_t = [(\mathbf{h}_1^t)^\top, \ldots, (\mathbf{h}_M^t)^\top]^\top \in \mathbb{R}^{3M}$, with $\mathbf{h}_i^t = \mathbf{p}_i^t - u(\mathbf{p}_i^t)$. Besides being independent of the position and orientation of the Kinect and the user, in this approach the possible variation in the movements will be distributed as smaller changes stored in all the joints which are part of the movement. Figure 7.9.c shows this type of features for a given body pose.

7.6.2. Feature Normalizations

Additional to the feature representation, we propose two schemes for normalization of these extracted features. Applying such normalization techniques can make the proposed features independent of the body styles and proportions. The aim of having these normalizations is to reduce the sensitivity of the system to the way different individual users perform a given gesture. This can improve the success rate of the recognition approach especially when the trainer user, the one who provides the system with his training data, is not the end-user of the system. In Chapter 9, we evaluate all combinations of representation and normalization methods in order to find the best pair of feature representation and feature normalization techniques for our recognition technique.

- **Relative normalization:** One dissimilarity source between the captured data from different individuals is related to their height. To overcome this, the acquired skeletal data should be scaled properly, simply by dividing all limb lengths by a value that is proportional to a given user's height. We use the distance between spine and head joints for this purpose.

- **Unit normalization:** Another possible difference can be related to the proportions in the users body. The relations between the length of different body parts e.g. legs, arms, etc., are not necessarily the same among all users. We therefore propose to scale all limb segments connecting two joints to unit length before computing the aforementioned features. This way, the vectors lose their length and keep their directions only. This solves both the differences in height of the users and proportions of their body segments.

7.7. Summery and Conclusion

In this chapter we explained two gesture recognition methods based on PCA and Manifold learning dimensionality reduction techniques. Two possible input devices, inertial sensors and Kinect, that can be utilized with the proposed techniques are also introduced. While the proposed techniques can be directly applied on the inertial sensors, for Kinect we proposed several feature extraction techniques. These feature extraction methods are composed of a representation and a normalization. Three feature representations and two feature normalization methods also have been proposed. Consequently, the proposed recognition techniques can perform independent of the body size and proportions of individual users. Using this technique, the very same training data can be used for different users. However, if customization is a point of interest in a usage context, the training data can be replaced with the data provided by the user at runtime. This is due to the fact that our gesture recognition methods rely only on the training data and compare the input data at runtime against it. With such a customizable recognition method, users do not have to adhere to a predefined sets of gestures and gestures can be defined by recording new training data. Additionally, while existing methods typically treat gestures as single commands, such as ”click the mouse”, the proposed gesturing techniques enable us to automatically recognize the performed gesture and track the movements within a gesture for fine-tuning continuous parameters. Next chapter proposes a specialized software framework, Signal.NET, which facilitates the incorporation of these gesturing techniques in the OR. Chapter 9 presents several quantitative and qualitative studies, conducted to evaluate the applicability of the proposed ideas.

CHAPTER 8

Signal.NET Framework

GESTURING interfaces can provide the surgeons with a direct control over the intra-operative devices. However, exploiting them in the OR requires additional care due to domain complexity. In this chapter, we present Signal.NET framework, which is designed to facilitate the integration of gesturing interfaces in the OR. Furthermore, we propose a methodological approach to exploit such a context-aware and customizable HCI designing technique in the real life OR scenarios.

8.1. Introduction

As mentioned in Chapter 2, the OR is a collaborative environment. In other words, the interaction with several devices can be shared among multiple users. Additionally, the interaction requirements vary based on the surgery type in each stage of the surgical workflow. Knowing these requirements are crucial in order to design a usable interface. However, in most cases these requirements can not be identified in advance as they depend on different information such as patient background and surgeon's decision. One solution to this problem is to provide the OR crew with a mechanism to customize the intra-operative user interfaces based on their specific requirements. This can be achieved using a prototyping approach which does not require programming knowledge. Prototyping is an important step in developing multimodal user interfaces, as it allows users and designers to customize the system behavior at runtime. With this approach, different modalities can be plugged and played in the system. This include various input sensors, different devices

to be controlled or novel gesturing techniques. Our goal is to develop a software framework for prototyping multimodal gesture-based user interfaces in the OR. We hypothesize that the structure of such framework should be based on the assembly of several components, namely different input modalities, gesture recognition techniques, and also various intra-operative computerized systems to be controlled.

In order to achieve aforementioned goals and to facilitate the development of gesture-base interfaces solutions in the OR context, we developed Signal.NET framework. The Signal.NET can be considered as an extensible software platform for supporting the effective and dynamic prototyping of multimodal gesture-based interfaces for the intra-operative systems. This framework fuses the interfaces of all intra-operative systems in a unified interface. It allows the users to customize the interface behavior based on the requirements of a specific surgery and complexity sources introduced in the OR domain model. It has been designed and implemented as a thin integration platform, able to manage the mentioned aspects, and thus provide the engineers of the OR domain a novel approach for development of user interface for medical solutions. Thanks to its component-based design, different medical devices can be integrated in the Signal.NET framework. This platform is equipped with a visual editors, a set of general and reusable components and techniques to assemble various input modalities, while preserving the performance of the integrated solution.

This chapter first presents existing works for the design and implementation of multimodal applications by focusing on the heterogeneous and extensibility aspects of each tool. Next, several requirements of an intra-operative gesturing platform will be defined. We then provide a short overview of the Signal.NET framework. The dedicated graphical front-end, and the runtime platform is then presented. In order to integrate the proposed 3D gesture recognition techniques, in Chapter 7, we introduce several reusable components. These demonstrate the applicability of the proposed framework for defining and customizing gesture-based interfaces. Finally, before conclusion, a methodology is proposed for utilizing such frameworks in practice in the OR.

8.2. Related Work

Multi-modal interfaces are composed of input streams from different input devices and rely on their underliyng software architecture. Several toolkits have been proposed for supporting multi-modal application development. In this section, we explore several well-known platforms designed for this purpose and explain about their shortcomings that Signal.NET

aims to overcome. In [52, 91, 126], the authors propose a modular framework for combining multiple input devices for developing multi-modal interactive systems. Bogdan et al. [17] extend this idea by proposing a high level modeling approach. This facilitate inexperienced users to manipulate the interaction interface. In [11], authors introduce iStuff, targeting UI prototyping in ubiquitous computing environment. A visual editor allows the end-user to customize the mappings of the Patch-Panel. ICON [35] enables interactive applications to achieve a higher level of input adaptability and is developed in the java platform. Basically, it can support several native input devices. Such input devices can be attached to the toolkit using JNI, Java Native Interface, which allows low-level integration with programs created using C. Another java-based platform is ICARE [20], which is a component-based framework for creating multimodal software solutions. The components are defined based on Java Beans, and should be developed in java programming language. This framework can not be extended easily and the components are not reusable. To integrate new input devices or components, additional programming is required. CrossWeaver [127] is a UI development tool and facilitates planning multimodal solutions. It enables the designers to define multimodal interfaces in a prototyping fashion. It only supports few input and output components and extending it with new components is not easily possible. IMBuilder [22] is another tool which models the interactions using finite state machines. Transition between nodes are connected to commands and actions. HephaisTK [36] framework uses a central agent from which all the components communicate. It is also equipped with a special scripting language to configure the interactions. Exemplar [59] enables the users to concentrate on design thinking. This means the developers do not have to work in a sensor abstraction level. Their main task is to define the way that user interaction should be performed. The editor is defined within Eclipse and also offers modification of live sensor data.

Although there are several software frameworks for developing new medical applications based on existing components, most of these environments are limited to image based problem solving, appropriate for pre-operative planning. MeVisLab [96] from MeVis Medical Solutions AG and Fraunhofer MEVIS and SCIRun [38] from University of Utah are such problem solving and prototyping tools which are widely used in research labs for 2D and 3D image processing. On the other hand, Slicer3 [111] provide the base platform for implementing medical components. Although it is widely being used during pre-operative planning and patient studies, the underlying architecture is not suited for real-time data streaming and synchronization. Furthermore, prototyping new application based on workflow and patient data model is not possible in this tool. Figure 8.1 shows the graphical user interfaces of these applications. MITK [145], VolView [134] and MIPAV

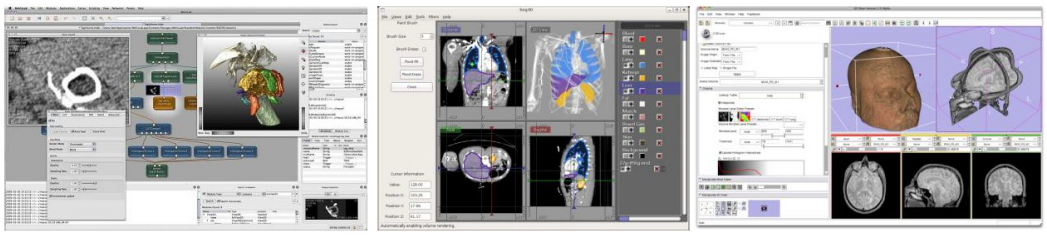


Figure 8.1.: Screenshots of MevisLab, SCIRun and Slicer3.

[95] are other end-user applications for viewing and navigating medical images which let the user do various image processing tasks. However, they do not allow prototyping new applications based on pre-existing components.

Most of the aforementioned solutions are targeting a specific technology or support a limited number of input modalities such as mouse, pen, text, and voice, and are designed for specific interaction scenario. Compared to the these methods, the main contributions of the Signal.NET framework can be summarized as follows:

- Signal.NET focuses on providing an interaction and device independent flexible solution for the fast prototyping of multimodal gesture-based interfaces through the facilitation and reuse of preliminary software and technologies.
- Unlike applications developed in medical UI builders such as MeVisLab, the proposed architecture runs as a standalone system and is not hardcoded into the medical solution itself. This architecture is much more flexible and applicable, considering diverse set of OR devices, workflows and preferences of different schools of surgeries.
- Signal.NET framework is designed specificity for the OR, considering main intraoperative aspects (e.g. surgical workflow, OR human roles and interaction requirements with arbitrary hardware and software solution) as defined in the OR domain model in Chapter 2. The underlying software architecture is capable of hosting gesture-based interfaces and enables dynamically associating learned gestures to features of arbitrary target systems, taking into account different workflow stages. Additionally, interaction tasks can be distributed between multiple users in the collaborative OR environment.
- The visual programming interface of Signal.NET enables design of interaction at runtime and allows composition of components through techniques such as design-by-demonstration or direct manipulation.

- We propose a methodology to bring such HCI designing concepts in the OR. Currently, surgeons are limited to predefined interfaces. Using this methodology, OR UI can be customized based on surgeons preferences (considering patient state, OR team, etc.), before the surgery. Such concept is novel in the OR and has never been proposed or implemented before.

8.3. Clinical Use-case

The concepts proposed in this chapter can be beneficial in many types of interventions, however, in this work we consider an exemplary surgical scenario of the implantation of a stent. This example is defined in close collaboration with experienced surgeons and contains a complex workflow, where the OR team interacts with multiple devices. The first step is to find an insertion point for the catheter in the artery of the groin or arm. While in most cases this does not involve imaging, for obese patients the use of ultrasound might be required to find an insertion point. Then, the catheter, the guide wire and the stent are inserted and the correct placement is confirmed using angiography. In complex cases also IVUS might be used to assess the location and amount of calcification. After opening the stent, its correct placement and opening is validated using angiography. As an alternative to angiography also CTA might be used. Prior to or after the use of any intra-operative imaging modality, re-adjustment of the lighting conditions in the OR may be required for better perception.

8.4. Requirements Specifications

In this section, we discuss about the practical requirements of a multimodal gesture-based UI designing framework for context of the Operating Rooms. These requirements should be considered in order to smoothly integrate such new interaction techniques in the surgical workflow. An important requirement for this platform is extensibility. This is due to the fact that different kinds of features in a wide range of intra-operative systems might be target of the interaction. For example, in the envisioned scenario outlined before, an intra-operative image viewing application lets the surgeon to select a patient profile, load the relevant preoperative image data, and change the viewing parameters, e.g. current slice number, zoom and pan, to find the region of interest.

8.4.1. Customization

As an important requirement of an interaction prototyping framework, it should allow the users to customize the interactions based on their requirements. Chapter 7 demonstrated how gestures can be trained with the end user. Additionally, in a gesturing interface, the mapping of trained gestures to the features of a target device should be defined in a customizable manner. Although a wide range of body movements can be defined as gestures, selecting an appropriate subset of gestures is strongly dependent on the usage context. Usually this kind of knowledge is not available in advance. For instance, as highlighted in the exemplary scenario, there are situations during a clinical intervention, in which the surgeon requires to change some settings of an intra-operative system. However, only one of his hands may be free as the other hand is occupied holding a medical tool. In this case the the framework should enables the user to map a trained gesture with the free hand to the required functionality.

8.4.2. Workflow-Awareness

Reducing the number of gestures increases the usability and reduces the duration of the training phase [26]. However, in complex domains there are many features of a system which might be necessary to access. To reduce the number of gestures as much as possible and still provide necessary control over the target system, a proper understanding of the usage workflow is necessary. At each workflow stage, usually only a limited number of settings are required. By defining a separate interaction configuration for each workflow stage, the proposed gesture framework can use the same set of the gestures to apply different settings in each workflow stage. One of the main advantageous of such a context aware design is that these mappings can be later defined based on the specific user requirements at any time. For example at the beginning of the surgery where no data is loaded, a given gesture can be mapped for loading data and afterwards when this data is loaded the same gesture can be used for closing the file. This context awareness also enables the users to interact with a larger set of features even with few number of gestures.

8.4.3. Multiple Users

Moreover, the Operating Room is a collaborative domain in which multiple users interact with shared resources to achieve a common goal [87, 44, 13]. In such scenarios, the interaction with the target system may be distributed between different human roles. Since each orientation sensor can be identified uniquely, the framework should provide functionalities

to rewire for supporting multiple-user gesture interfaces [87, 44]. For example, the main surgeon may need to browse the medical images, while the anesthetist is checking different live signals on the screen. Each of these users may separately interact with the system, while the same system is processing the sensors, detecting the gestures and propagating commands to the target features (Figure 8.2).

8.4.4. Integration of Gesturing

As outlined before, gesture recognition approaches can be classified either as categorical or spatio-temporal [56, 83]. In contrast, our proposed gesture recognition approaches in Chapter 7 has characteristics of both categorical and spatio-temporal gesture recognition methods. One of the main objectives of this project is to bring such novel gesturing approaches into the OR. Hence, the envisioned framework in this chapter should provide the required functionalities.

8.5. Framework Architecture

This section provides an overview of the design and implementation details of the Signal.NET framework. First we explain the concept of the component model and continue by introducing the data pipeline design pattern. We believe that such an architecture enables rapid development with heterogeneous components and can be used in conjunction with high level visual programming interface, enabling the customization at runtime. Finally, we demonstrate how the configuration of the component pipelines can be stored and managed using a visual editor.

8.5.1. Heterogeneous Components

In order to achieve the desired extensibility and customizability, we propose a framework with a component-based architecture. In this model, components encapsulate elements of an interactive system. Components are unaware of the framework in which they are running. A component is only characterized by its functionalities and is defined as a reusable and independent software unit with a defined interface. Components expose their functionalities via these interfaces to the framework as a set of signals, an abstraction of typical properties, methods and events. Each signal has a specific data type and is labeled with an access attribute. Currently supported data types are boolean, integer and real values and valid access attributes are read-only, write-only and read-write. This design is intentionally wide enough to encapsulate a large number of software components and hardware devices

when implementing an interactive multimodal solution for the OR. The components can then be easily reused in any solution developed using Signal.NET framework in a plug and play fashion by defining the pipelines, presented in the following section. To summarize, components should be defined by the following attributes:

- *Signal Interface:* List of signals that a component provides to communicate with other components and the framework.
- *Documentation:* The component must be well documented to enhance reusability.
- *No dependencies:* A component must not be dependent to the framework or any other component. All the required features should be included within the component itself.

8.5.2. Bindings and Data Pipeline

In order to flexibly set up the connections between recognized gestures and features of the target components, we further extend the proposed component-based architecture with the pipeline data flow model, as shown in Figure 8.2. Since the components and their properties are known to the framework, we developed a binding mechanism to create a connection between properties exposed by different components at runtime (late binding). A binding can be either directional or bi-directional. Once such a binding is defined, the framework propagates values of all the changed signals in the defined directions. This guarantees that if a property value changes in one component, all the dependent properties in the other components will be updated accordingly. The framework synchronizes the update process using an internal clock. Moreover, the framework internally handles required data type conversion. Since the data transmission in the system is happening

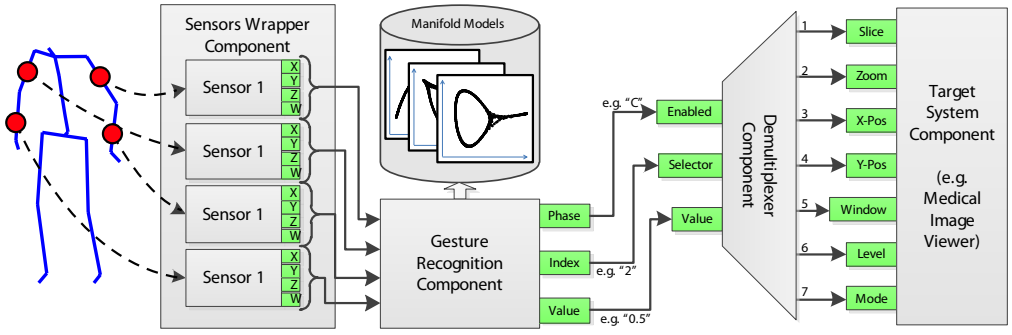


Figure 8.2.: An exemplary gesturing pipeline composed of several components.

within these connections, it is possible to transmit the data over network, extending the application scope to distributed platforms. With this pattern, end-users can manipulate the user interface response to detected gestures by altering the underlying pipeline graph. A pipeline also supports dynamic reconfiguration of connections at runtime. The pipeline model implemented in the Signal.NET framework provides simple embedded dataflow controls, such as direct function calls and asynchronous calls. The pipeline also enables isolating component in separate processes or distributing the execution of an assembly over a set of computers running an instance of Signal.NET framework.

8.5.3. Configuration Management

The pipeline model, internally is stored as a list of components and their interconnections, as shown in Figure 8.3. This configuration also can be serialized in an XML configuration file, shown in Figure 8.4. Figure 8.5 demonstrates schema of the configuration XML file. Parsing the configuration file at runtime, the framework creates the required set of components, as well as the defined bindings between them.

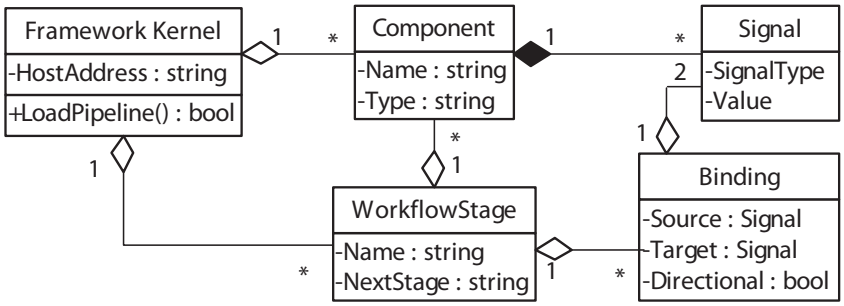


Figure 8.3.: A class diagram demonstrating the framework architecture.

Using this approach, the behavior of the system can be completely customized, as required for the specific usage scenario, without extra programming. For instance, one can specify which gesture changes the current slice number property of the medical image viewer application. This is possible by specifying a proper binding between the gesture recognition component and the medical image viewer component. Having such a persistent pipeline model, the behavior of the system can be defined and retrieved, as required for a specific usage scenario, without extra programming.

```
<InteractionPipeline>
  <Components>
    <Component ZIndex="79" Y="35" X="713" GUID="a2817461" TypeName="TargetProxy" AssemblyName="BasicComponents, Version=1.0" />
    <Component ZIndex="82" Y="94" X="200" GUID="ebbc03d6" TypeName="ManifoldPredictor" AssemblyName="ManifoldPredictorManagedWrapper, Version=1.0" />
    <Component ZIndex="109" Y="99" X="392" GUID="a4308e9a" TypeName="DeMultiplexer" AssemblyName="BasicComponents, Version=1.0" />
    <Component ZIndex="103" Y="36" X="18" GUID="615bf054" TypeName="KinectDevice" AssemblyName="Kinect, Version=1.0" />
  </Components>
  <Stages>
    <Stage Name="Localisation">
      <Bindings>
        <Binding IsDirectional="True" TargetSignal="Dim" TargetComponent="a2817461" SourceSignal="Output4" SourceComponent="a4308e9a" />
        <Binding IsDirectional="True" TargetSignal="Frequency" TargetComponent="615bf054" SourceSignal="Output1" SourceComponent="a2817461" />
      </Bindings>
    </Stage>
    <Stage Name="Insertion">
      <Bindings>
        <Binding IsDirectional="True" TargetSignal="Index" TargetComponent="a4308e9a" SourceSignal="Index" SourceComponent="615bf054" />
        <Binding IsDirectional="True" TargetSignal="Value" TargetComponent="ebbc03d6" SourceSignal="Value" SourceComponent="615bf054" />
        <Binding IsDirectional="True" TargetSignal="Activity" TargetComponent="615bf054" SourceSignal="Is Started" SourceComponent="a2817461" />
      </Bindings>
    </Stage>
  </Stages>
</InteractionPipeline>
```

Figure 8.4.: An exemplary pipeline configuration XML file.

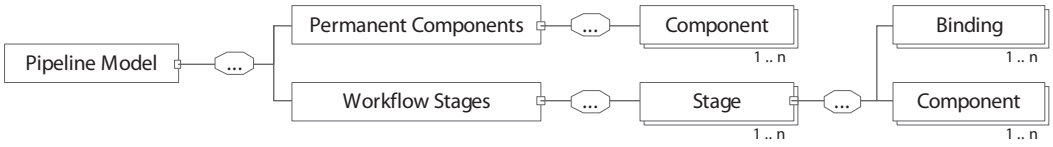


Figure 8.5.: The XML schema of the configuration file.

8.5.4. Visual Programming Interface

The interaction pipeline, including all the hosted components and their bindings, defines the behavior of the system in each stage of the surgical workflow. The interaction pipeline can be visually presented to the user as a graph, as shown in Figure 8.2. Components can be presented as the nodes of the graph and bindings as edges. Such a graphical representation can provide the end users with a visual programming interface, allowing them to customize the system behavior at runtime. By utilizing this technique in the proposed architecture, new combination of components and their binding could be formed for each specific intra-operative scenario. Using such notion, during requirement analysis stage of the OR-Use project, we prototyped a visual editor for customization of the interaction pipelines, as shown in Figure 8.6. This prototype is created in a UI prototyping tool.

Facing the need to customize interaction pipelines for specific surgeries, later, we developed a completely functional version of such visual programming interface, as was defined in our requirement analysis stage. Figure 8.7 shows the final user interface of the visual programming interface. Users can add any of the available components, presented in the left panel, to the interaction pipeline and create connections by drag and dropping properties (signals). The top right panel includes a list of workflow stages, in which users can manage the surgical workflow. This list can be customized for any given surgical routine.

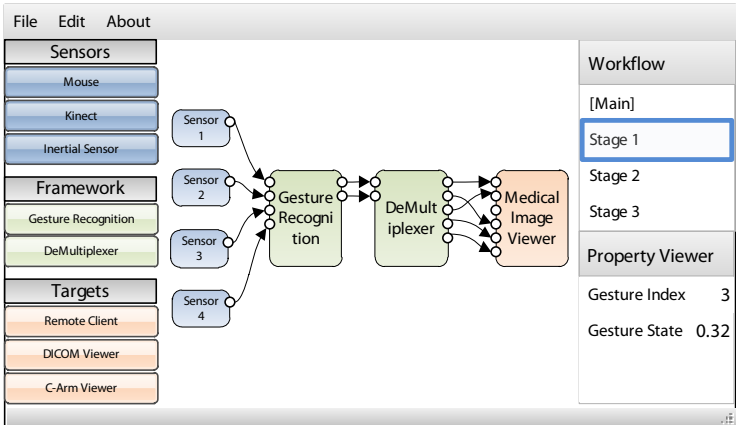


Figure 8.6.: An early prototype of a pipeline editing application.

Additionally, the editor can serialize and restore the pipeline from an XML file. This is important since the design and actual usage may happen at different times. Developers can use this graphical interface to dynamically or statically combine components, and generate a running application.

8.6. Gesturing Components

Here, we discuss practical aspects and possibilities about integration of the extracted information about the gestures, the gesture type and state, into the Signal.NET framework. As mentioned in Chapter 7, categorical gestures are typically used for functional features (e.g. loading data, flipping an image). Alternatively, spatio-temporal gestures are more suitable for parametric features where a value is required (e.g. scale factor). However, the interface of a target system usually contains both functional and parametric features. For example, a zoom tool should be activated (functional feature) before modifying the zoom factor in a view (parametric feature). Utilizing the proposed gesture recognition technique, the extracted gesture type and gesture state can be used to set multiple parametric features of a target system without any need for activating a tool or controller in between. As an example, two gestures can be defined as horizontal and vertical movement of the right hand in front of the body. The first gesture can be assigned to control scale while the second gesture is used to change the brightness. In this scenario, a user can fine-tune these parameters simply by moving his hand accordingly and based on the gesture type the application can update the required changes using the gesture state. This is similar to the way that in multi-touch interfaces, users can zoom and pan without explicitly activating the relevant

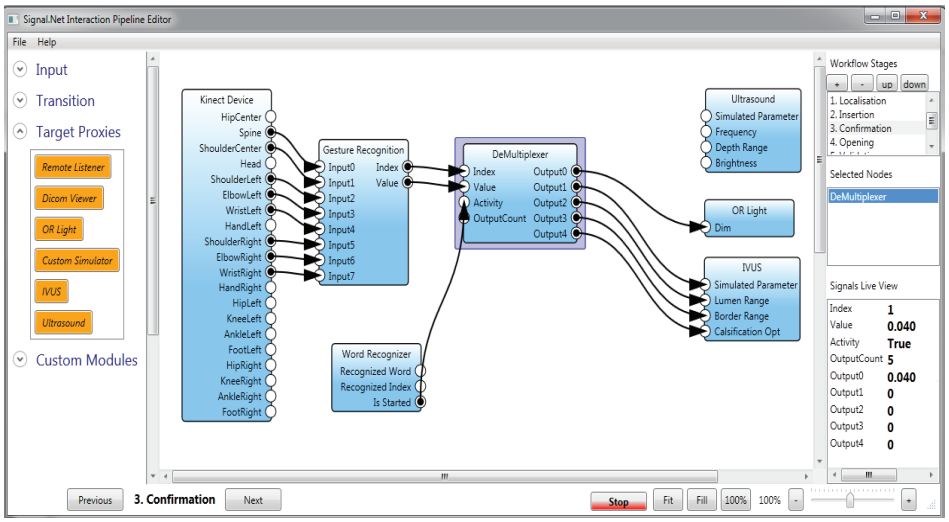


Figure 8.7.: Visual programming interface of the Signal.NET framework.

controller before. The integration of our gesturing techniques, requires at least two stages of training and interaction, as shown in Figure 8.8. Later in this chapter we propose a methodological approach which fulfils this requirement.

The proposed gesture recognition methods in Chapter 7 or in more general the categorical and spatio-temporal gesturing techniques can be integrated into the Signal.NET framework by defining four new types of components. This enables users to utilize visual programming interfaces for customizing the system behavior through binding the gestures to the target features. As shown in Figure 8.9, we have created separate components for the input sensors, the gesture recognition method and a medical image viewer as an exemplary target device. The following sections explain these component types in more detail.

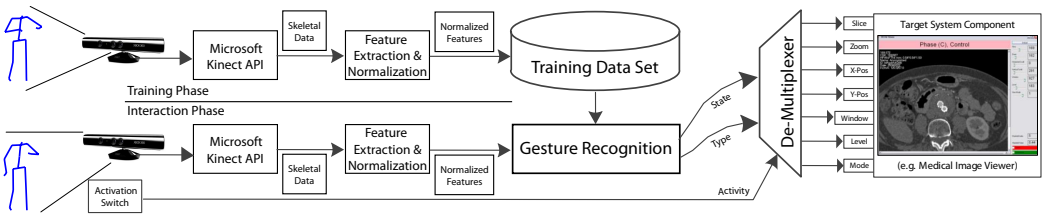


Figure 8.8.: Exemplary data flow diagram for integration of the proposed gesturing technique in the training and interaction phases.

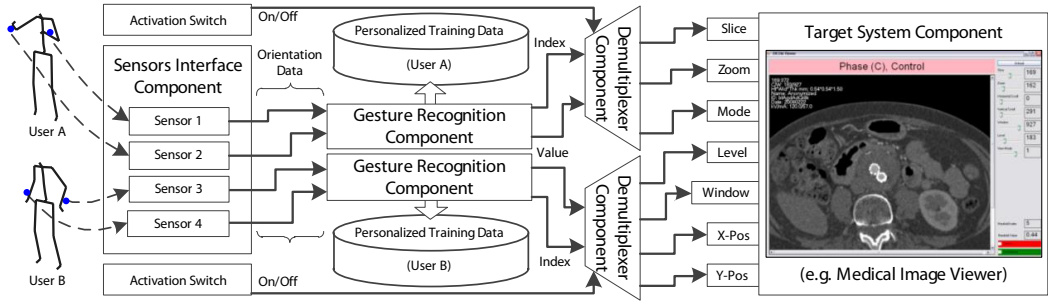


Figure 8.9.: A component pipeline containing four main types of gesturing components of the Signal.NET framework.

8.6.1. Sensor Components

These components encapsulate a specific type of input sensors. This can be Kinect or inertial sensors. Each instance of these components contains signals to read data from an input device. All the required code to communicate with the device application programming interface is implemented inside this components.

8.6.2. Target Components

Components of this type wrap the interface required to control a specific intra-operative target application. In order to control a new application with the proposed gesture-based interaction framework, a specific target component needs to be implemented. Applications could be software or hardware systems which provide a proper programming interface. Using the proposed model, the framework can access both, functions and parameters of the target application. Functions (e.g. load, save, close, etc.) can be implemented as a method within the target component and can be triggered with an event. Parameters can be encapsulated as a component property which stores a numeric or logical value and reflects a specific setting of the target system (e.g. visualization mode, zoom level, horizontal scroll, etc.). As an example, an intra-operative imaging device could be wrapped within a component containing properties like zoom, brightness, contrast etc. Such a modular design pattern satisfies extendibility requirement of a generic intra-operative HCI designing framework and therefore a wider range of computer-based equipments could be combined and controlled with the proposed gesture based user interface.

8.6.3. Recognition Components

The gesture recognition component hosts the implementation of the gesture recognition approaches presented in Chapter 7. This component has two sets of properties, input data and the recognized output. The value of the input data properties are updated directly based on recent sensor values. The number of input properties and the used dimensionality reduction technique are specified in a configuration file which can be used to customize the framework. The recognized output properties are gesture type and gesture state. The gesture type is the index of the current detected gesture and introduced in Chapter 8 by $\hat{c}_t$. The gesture state is a relative pose within the current gesture and is defined as $\hat{x}_t \in [0, 1]$.

8.6.4. Demultiplexer Components

These components act as a switch between the gesture recognition components and target components. They forward the recognized gesture value to the relevant property of the target component. As an input, a demultiplexer component receives the recognized gesture index ($\hat{c}_t$) and the current relative pose ($\hat{x}_t$) from a gesture recognition component. For each of the trained gestures, the demultiplexer component has a separate output channel. The current gesture state ($\hat{x}_t \in [0, 1]$) is routed to the relevant output channel, based on the recognized gesture type ($\hat{c}_t$). This removes the dependency on other interaction techniques, such as mouse or voice commands, for switching between different properties to control. Furthermore, using its activity property, the data flow can be enabled or disabled.

8.6.5. Activation Control

Activation of the gesture-based interfaces is a common problem in this field [66]. In order to prevent that movements similar to the trained gestures be considered as commands, an activation switch should be exploited. We solve this problem by using an additional switch which controls the activity of the gesture-based interface. The activation switch can be implemented as an input component and the data flow within the gesture recognition components may be interrupted in the demultiplexer component, as shown in the figure 8.9. This switch can be used to enable and disable the activity of the demultiplexer component. It can be physically implemented using different techniques. External switches, such as pedals or voice commands, are among the most common options. In our studies, we examined two different implementations, one using a voice recognition module and the other using a wireless handheld switch. Note that these activation modes are not the focus of our work and the selected options are thus straightforward. Furthermore, once the user finds

the proper value of a continuous parameter using its corresponding gesture, the gesture-based interface should provide a method to store the selected value. The aforementioned two exemplary implementations were also used for this functionality.

8.7. Component Database

Table 8.1 lists a selection of components currently developed in the Signal.NET framework. The components are designed to be highly reusable and originate from different projects and experiments. The database is categorized to input components, target components, gesturing components, and transition components. Many of these components have been already tested and can be utilized in order to define other interaction solutions.

Category	Component Name	Description
Input Components	MouseKeyboard	Captures events from the mouse and keyboard.
	Kinect	Retrieves skeletal information from Microsoft Kinect.
	InertialSensor	Receives orientation data from inertial sensors.
	VoiceCommand	Recognizes voice commands from the user.
Target Components	ORLight	Simulator for an intra-operative light dimmer.
	IVUSViewer	Simulator for viewing intra-vascular ultrasound data.
	Ultrasound	Simulator of intra-operative ultrasound device.
	DICOMViewer	Viewer of medical images with DICOM format.
Gesturing Components	PCARecognition	Gesture recognition component using PCA.
	ManifoldRecognition	Gesture recognition component using manifold learning.
Transition Components	Demultiplexer	Switch for forwarding gesture commands.
	RangeMapper	Maps the data from an input range to an output range.
	TypeConvertor	Converts output signal data to the type of input signal.

Table 8.1.: Selection of components available in the Signal.NET framework.

8.7.1. Implementation

The Signal.NET framework is developed using Microsoft .NET framework. The official Microsoft Kinect SDK was used to wrap the Kinect in an input component in the framework. The gesture recognition method was developed in Matlab and the compiled binary wrapped as a new component using C. A voice recognition module was developed using Microsoft voice API. In order to develop an interaction method based on inertial sensors, hardware related issues, such as noise and drift, need to be considered. We reduced the noise effect by applying a mean filter with a window of 10 frames (equivalent to 200ms) to the output of the gesture recognition method. This introduces a short delay; however, the

effect is ignorable. Drift causes an incremental inaccuracy in the sensor measurements. To compensate this effect the boresight direction of the sensors should be reset periodically. In our experiments, we reset the sensors while they are attached to the user's body and he is standing with released hands on two sides of his body. The DICOM image viewer application was developed using an open source component with functionalities such as zoom, pan, brightness and contrast.

8.8. Context-Awareness

As explained before, OR is a highly complex environment where surgical staff often use various types of assistive computerized medical systems, for instance to visualize patient data. In such situation context awareness is an important attribute of a user interface. Here, we demonstrate how a gesture-based interface, developed using the Signal.NET framework, can be customized in a context-aware manner based on different complexity aspects of the OR, introduced in Chapter 2. An example is given in Figure 8.10.

- **Workflow awareness:** At any given time during the usage workflow, typically only a small subset of features of a target application is practically needed. With proper understanding of the workflow, a separate pipeline configuration can be defined for each workflow stage, hence the same set of gestures can be used for different purposes. Figure 8.10 demonstrates a specific pipeline, defined for the highlighted workflow stage.

- **Multiple targets:** Our modular architecture enables simultaneous control of different target applications. By encapsulating each target application in a corresponding

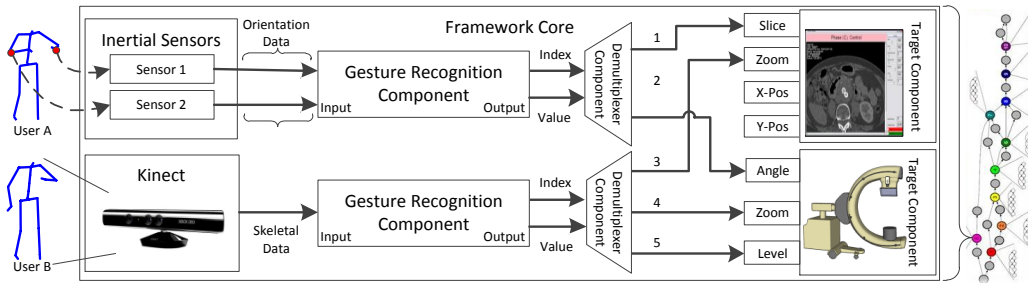


Figure 8.10.: An exemplary collaborative domain scenario, where two users controlling two different medical application. This pipeline defines the system behavior in the highlighted stage of the surgical workflow.

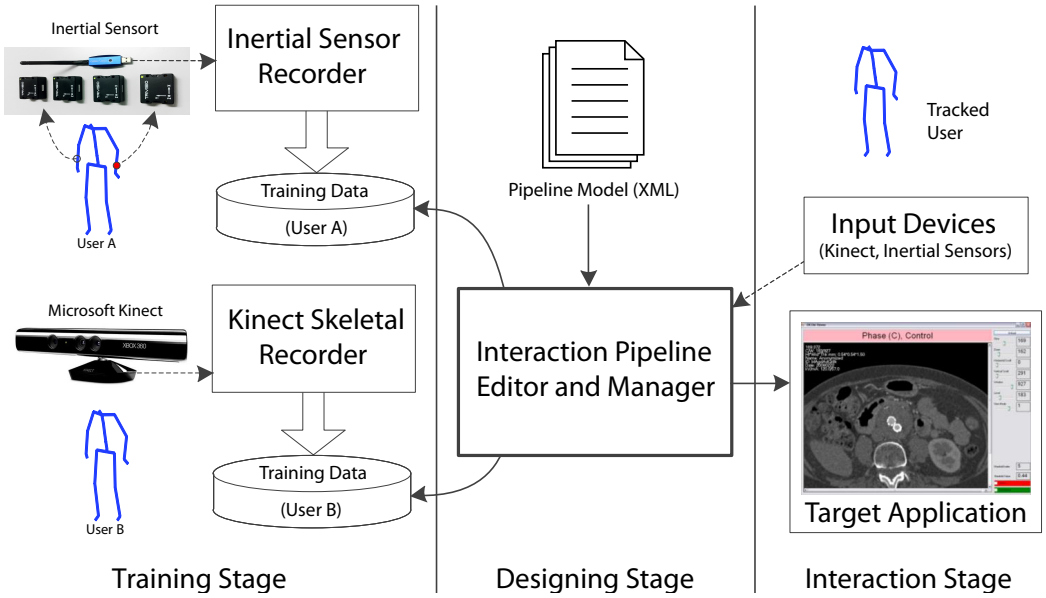


Figure 8.11.: Practical stages for deploying a gesture-based interface in the OR.

target component, several of such targets can be included in a pipeline, possibly running on a separate processes or machines. Figure 8.10 shows an interaction pipeline to control two different target applications.

- **Collaborative environment:** In a collaborative domain, such as the OR, the interaction with a target application may be distributed between multiple human roles. Since the input sensors are uniquely identified each with a separate component, the underlying data pipeline can be rewired properly to provide the required control for each individual user. In other words, with a proper configuration of the interaction pipeline a customized interface can be defined for each user in the OR, who is equipped with a separate set of sensors, as shown in Figure 8.10.

8.9. Methodological Approach

In order to utilize the Signal.NET framework in practice in the OR, here we propose a methodological approach, which consist of four stages: planing, training, designing and interaction. Figure 8.11 pictures the relation of the training, designing and interaction stages. Up to our knowledge, there is no previous methodology for defining and customizing a unified gesture-based interface for the OR.

Stage	Role	Surgeon	Assistant	Nurse
	1. Localisation	DICOMSlice,Zoom	US.Frequency, Depth,IVUS.Slice	USTissue,ORLight.Dim
	2. Confirmation	DICOMSlice,Zoom	US.Frequency	ORLight.Dim
	3. Opening	N/A	N/A	ORLight.Dim
	4. Validation	DICOMSlice,Zoom	N/A	ORLight.Dim

Figure 8.12.: Interaction requirements form, used during planing stage.

8.9.1. Planing Stage

In this phase, surgeons are asked to fill a form to define the interaction requirements of their upcoming surgery. The structure of this form consists of the main aspects of the OR domain model, Figure 8.12. Surgical workflow stages are in rows, OR human roles are in columns and the cells contain a list of required features of different target devices. Filling this form, the head surgeon can distribute the task of interaction among his team within each stage of the surgical workflow.

8.9.2. Training Stage

During the training stage, a recorder application is used to collect the training data, consisting of one example per gesture. This step enables users to customize movements which our recognition techniques identify as gestures. Although, this can be done with surgeon if needed, in most cases once the system is trained there is no need for extra refinements.

8.9.3. Designing Stage

After planing, the visual editor, presented in Figure 8.7, is used to define the interaction pipelines and fulfill the expected requirements. This can be done either directly by surgeons or by an OR technician who is more familiar with the underlying concepts. Once such an interaction pipeline is defined, it can be reused for similar surgeries. This can also be done by user interface experts of the manufacturer upon the deployment of a new OR in a hospital, based on the type of operations performed in that hospital.

8.9.4. Interaction Stage

During surgery, the system loads the defined configuration and starts updating the pipeline. Following the current surgical workflow stage, a required subset of features would be available to the users as planned before. The framework continuously reads the sensor

values, runs the gesture recognition method and updates the target devices based on the recognized gestures and the defined pipeline. The current workflow stages can be provided either manually or using an automated activity recognition technique [16]. The system behaviour can be further customized at runtime, using the visual editor or by changing the current workflow stage.

8.10. Summery and Conclusion

In this chapter, we proposed Signal.NET framework for integration and customization of intra-operative interfaces, within the context of the OR domain. Although, there are several platforms to facilitate the implementation of multimodal gesture-based interfaces in other domains, this concept is novel for the Operating Room. Signal.NET framework enables reuse of existing components to iteratively design and build a multimodal system with minimal programming effort. Moreover, the used architecture allows it to be interfaced with any kind of front-end application which can run Signal.NET framework. We also presented our visual programming interface, built on top of the Signal.NET framework. The proposed categorical and spatio-temporal gesture recognition approaches can be combined into the Signal.NET framework utilizing four types of component. We demonstrated that the proposed software framework enables a dynamic association of learned gestures to arbitrary target systems. This enables the application to customize its behaviour in a context-aware manner. The proposed methodological approach provides the required foundations in order to smoothly exploit such frameworks in real life OR scenarios. We believe that such flexible and customizable UI designing approaches are necessary to bring many solutions proposed within the medical community into final application in the OR. The extensibility of the proposed architecture makes it also applicable to a wide range of similar scenarios.

CHAPTER 9

Case Studies and Evaluations

SEVERAL quantitative experiments have been performed to evaluate the performance and applicability of the proposed gesturing techniques. Additionally, we have conducted several user studies, through a real-life scenario with biomedical computing students, surgeons and user interface experts. These user studies are conducted in order to evaluate the usability and applicability of the the proposed HCI designing framework. This chapter presents the results of these experiments and user studies.

9.1. Introduction

As mentioned before, the OR is a highly advanced environment where surgical staff often work under high pressure. Many types of computerized medical systems are used in this domain for assistance. Interacting with such devices during surgery is a well-known problem in this domain [81]. This issue inspired us to propose our gesture recognition approaches and the interaction designing framework, aiming at introducing an intuitive user interfaces for this context. In order to evaluate the proposed gesture recognition methods and Signal.NET framework, in this chapter we report from several quantitative experiments. We also report of the conducted user studies with experienced surgeons and experts of the human computer interactions community. The proposed ideas are general and can be applied in many different medical cases. However, in this work an exemplary case of Stent implantation has been the basis of our examinations.

Figure 9.1.: Sample images of a trainer subject performing one of the considered gestures.

9.2. Quantitative Experiments for Recognition Techniques

This section focuses on quantitative experiments conducted to evaluate the functionality of the proposed gesture recognition techniques. First, we evaluate the performance of the Manifold-based method using inertial sensors, then we evaluate and compare PCA-based method using inertial sensors. Finally, PCA-based method is evaluated with the Kinect data. Our implementations for this section all has been done in Matlab and runs on a 2.5 GHz Intel Core 2 Duo machine with 4 GB of memory.

9.2.1. Inertial Sensors with Manifold

In order to evaluate the gesture recognition method based on Manifold learning technique a systematic experiment was conducted on a dataset of 18 different gestures captured from inertial sensors, each recorded four times with one person. We created labeled sequences of multiple gestures in a row, each between 1000 and 5000 frames. Figure 9.1 demonstrates a user while recording the training dataset.

9.2.1.1. Gesture Recognition Rates

The gesture recognition accuracy of the proposed method was evaluated using the recorded data set. This experiment was performed in a cross-validation manner, each time using one of the sequences for training and one of the others for testing. While measuring the percentage of frames with correctly recognized gestures, we varied the number of simultaneously trained gestures and the number of inertial sensors. As shown in Figure 9.2.a, best gesture recognition rates were achieved with six inertial sensors. In this case, 95% of all frames were correctly recognized with a set of four gestures, and 88% when the method was trained on 18 gestures. Although the recognition rates decrease for less than six sensors, even only two sensors (one on each arm) yield correct recognition rates above 80% for all considered numbers of gestures. Our method thus scales well with respect to the

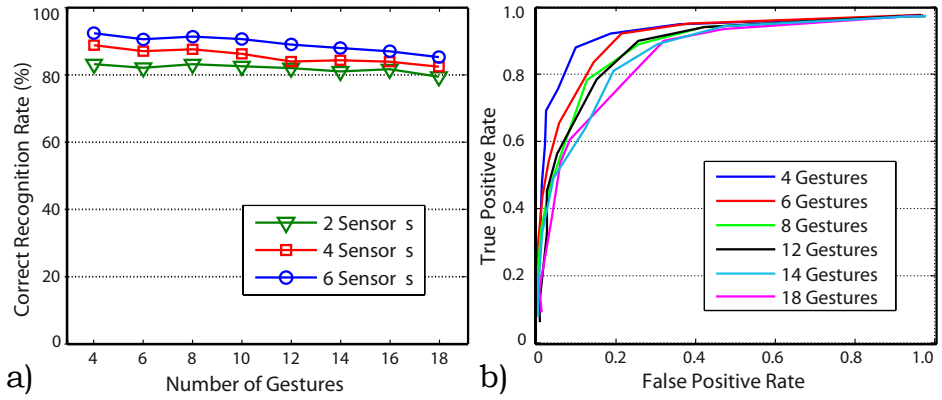


Figure 9.2.: (a) Correct gesture classification rates for various settings. (b) ROC curves for automatic differentiation of learned gestures from non-gesture movements.

number of gestures to be recognized simultaneously. However, we expect that less than 10 gestures need to be distinguishable in practical applications.

9.2.1.2. Gesture Segmentation Rates

Besides correctly recognizing gestures, it is crucial for a gesture-based interaction method to reliably distinguish instances of learned gestures from arbitrary other movements. As explained in Chapter 7, we use the confidence measure $\lambda_t^{\emptyset}$ for this purpose. In a series of experiments, we varied the associated threshold and measured the percentage of frames with gesture movements recognized as such (true positive rate, TPR) and the percentage of non-gesture frames wrongly identified as gestures (false positive rate, FPR). We trained the method on different numbers of gestures and randomly created sequences where only one out of multiple gestures was known to the method. In all cases, six inertial sensors were used. Figure 9.2.b shows the resulting ROC curves. As expected, the best combination of high TPR and low FPR was achieved in the setting with four gestures. In this case, above 90% of gestures were detected with less than 10% of false positives. Obviously, if more gestures are known to the system, distinguishing non-gesture movements becomes more difficult. However, even for the 18 gestures setting, detection results are still reasonable.

9.2.2. Inertial Sensors with PCA

For a quantitative assessment of the gesture recognition method based on PCA and comparing it with the Manifold-based technique, we recorded a larger dataset of 21 arm gestures

using inertial sensors, performed by one person in 12 repetitions per gesture. Considered gestures included movements of one or both arms, such as dragging an imaginary slider in front of the body, tracing out circles or waving. The experiments were performed on the recorded dataset of 10^6 frames in a batch mode to allow for systematic cross-validation. For each run, we used three of the 12 repetitions per gesture for training and three other repetitions for testing.

9.2.2.1. Gesture Recognition Rates

In order to quantify how precisely the PCA-based method is able to recognize gestures, we determined the percentage of correctly classified frames for different settings. We varied the total number of gestures the method is trained and tested on and used different variations of the method, as discussed previously. Results are shown in Figure 9.3. For every number $n_g \in \{4, 6, 8, 10, 12, 14, 16, 18\}$ of considered gestures, 10 automatic experiments were performed, each time randomly selecting subsets of n_g gestures for training and testing out of the available 21 gestures. For each choice of n_g, the average classification rate over the 10 experiments thus indicates the performance to be expected with a random selection of gestures. The maximum and minimum for each value of n_g can be achieved with an optimal and inappropriate set of gestures, respectively. Optimality in this case refers to a minimal overlap between gestures to be recognized simultaneously. As can be seen in Figure 9.3.a, using Laplacian Eigenmaps (Manifold) for dimensionality reduction results in a similar recognition behavior to using PCA. In both cases, 95% of frames are correctly classified when 4 gestures are considered, with a decrease to 85% for 18 simultaneously learned gestures. For any value of n_g, accuracy decreases by 2-3$\%$ without the smoothness modification (Equation 7.3).

9.2.2.2. Influence of Training Dataset Size

As a short training phase is desirable for the users [83], we evaluated the behavior of our method with respect to a varying number of repetitions per gesture in the training data. We recorded a dataset of 8 gestures with an increasing number of repetitions, ranging from 1 to 8. The sequence with 8 repetitions was taken for testing, while training on the other sequences, one by one. Apparently, training on a single repetition per gesture cannot sufficiently account for typical variation when performing a gesture multiple times. However, gesture recognition rates matched those displayed in Figure 9.3 for two repetitions per gesture, without additional improvement up to the case of 7 repetitions per gesture. We argue that performing two repetitions per gesture for training is not a significant burden,

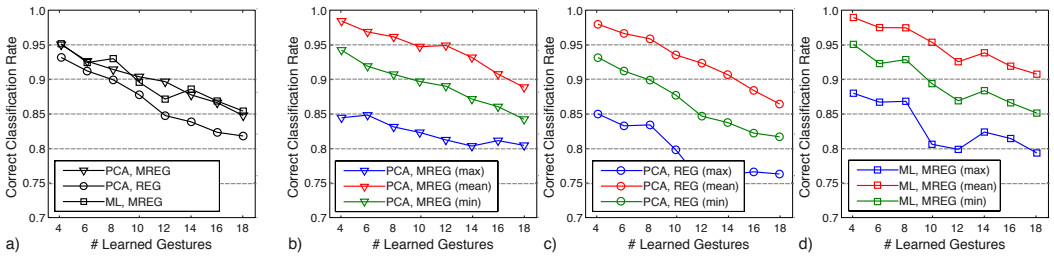


Figure 9.3.: Correct gesture classification rates for various numbers of simultaneously learned gestures (n_g) and different method options (PCA: principal component analysis, ML: manifold learning, MREG: regression with smoothness modification, REG: plain kernel regression). (a) Average rates for all experiments and 3 method settings. (b)-(d) Maximal, average and minimal rates. Maxima indicate rates for an optimal, non-overlapping choice of gestures.

considering that the system only rarely needs to be re-trained.

9.2.2.3. Processing Speed

Figure 9.4.a illustrates the computational performance of the gesture recognition method for all settings of n_g. The choice of PCA or Laplacian Eigenmaps for dimenisonality reduction does not have an impact on processing speed, as these methods are applied offline. Most computationally intensive is the kernel regression step that, in each frame, passes through all frames of the training dataset. The training dataset grows from 1,000 (for $n_g = 4$) to 8,000 frames (for $n_g = 18$). On our machine, the method allows for very high frame rates above 1,000 frames per second, even for 18 gestures. Performance is slightly lower when the smoothness modification is enabled. In practice, processing speed is thus only limited by the acquisition rates of the inertial sensors, typically 60-120 Hz. These results highlight the feasibility of deploying our method on portable devices with limited processing capabilities, such as smartphones.

9.2.2.4. Sensitivity to Sensor Placement

In order to evaluate how sensitive our method is to an exact placement of the sensors, we recorded an additional dataset with a subset of 8 gestures, varying the position of the sensors on the person's arms. Starting with a training sequence of all 8 gestures and the sensors in a reference position, we tested on sequences with incremental rotational and translational deviation of the sensors. Translation was changed to 4, 8 and 12 cm

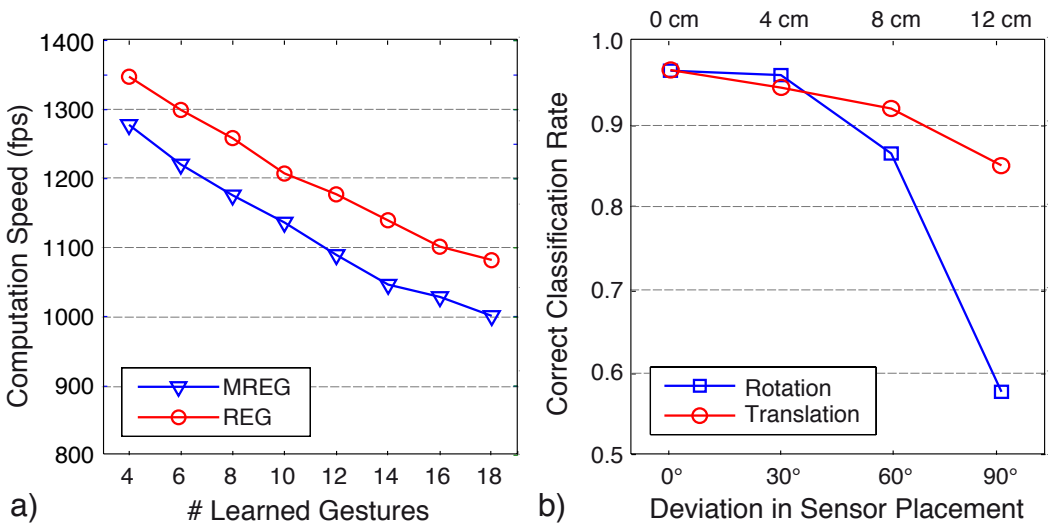


Figure 9.4.: (a) Processing speed in frames per second (fps) for different settings of n_g using kernel regression with smoothness modification (MREG) and plain kernel regression (REG). (b) Sensitivity of gesture recognition to incremental deviations in sensor placement.

(along the arms) from the reference position, rotation (around the arm axis) was changed to approximately 30, 60 and 90 degrees. Figure 9.4.b shows that gesture recognition is more sensitive to rotational deviation than to translations. However, rotations up to $30°$ and translations up to 4 cm almost do not affect recognition rates. We argue that rotational deviations close to $90°$, where recognition rates drop below 60%, are not to be expected under typical usage conditions. The results also imply that the sensors can be taken off and on, e.g. to continue work on another day, without a new training.

9.2.3. Kinect with PCA

Based on the above mentioned experiments, we noticed that PCA is capable of recognizing gestures with the same accuracy as Manifold based method for inertial sensors. As developing PCA-based method requires less tuning and several open source implementations are available in different languages, we decided to focus on PCA-based recognition technique for the remaining experiments. For a quantitative assessment of the PCA-based gesture recognition method using Kinect, we recorded two datasets of arm gestures. One dataset consists of 16 different gestures, performed by three persons in 6 repetitions per gesture. We use this dataset to evaluate the scalability of our method with respect to larger

numbers of gestures. The second dataset consists of 6 gestures, performed by 12 persons, also in 6 repetitions per gesture. This dataset was recorded alongside with the user study that will be described in the Section 9.4. The gestures consist of simple hand movements such as moving one hand up and down (as in moving a virtual vertical slider in front of the body), opening and closing both hands in front of the body (similar to opening and closing a window), moving the left hand to the left and right above of the head (like waving), etc. The focus here was to assess the variability of results for different persons. To allow for systematic cross-validation, we performed the quantitative experiments in a batch mode. For each run, we used 3 of the 6 repetitions per gesture for training and the other 3 repetitions for testing. Our experiments focus on the two main outputs generated by the gesture recognition method: the gesture type and the gesture state.

9.2.3.1. Gesture Recognition Rates

The gesture type is determined in each time step t by the value $\hat{c}_t \in \{1, \ldots, N\}$ generated by the gesture classification component. To evaluate the performance of this classification task providing the Kinect data and PCA-based recognition method, we measured the percentage of correctly classified frames for different settings. Using the dataset with 16 gestures and 3 persons, we varied the total number of trained and tested gestures in four steps from 4 to 16. The gesture classification problem obviously becomes more complex, the more gestures are available in the training dataset. In addition, we evaluated two scenarios regarding the training data: in a first series of experiments, we kept the person identical for training and testing (personalized training), and in a second series, we used different persons for training and testing (common training). All combinations of persons in the 3 person dataset were considered and the outcome was averaged. The results are shown in Figure 9.5.a, including separate values for each of the nine feature and normalization types described in Chapter 7.

A general observation is that using personalized training data results in better gesture recognition rates. This effect increases with the number of considered gestures. However, for reasonable numbers of gestures, such as eight, classification rates only deviate by less than 10%. In the case of four gestures, the common training data (i.e. training data of one particular person, used for all other persons) accounts for as high recognition rates as the personalized training data. Another general observation is that maximum recognition rates decrease with an increasing total number of gestures. While this behavior is expected, the rates only drop slightly from above 90% in the case of 4 gestures to a reasonable 85% for the case of 16 gestures. In terms of feature selection, the distance features clearly perform

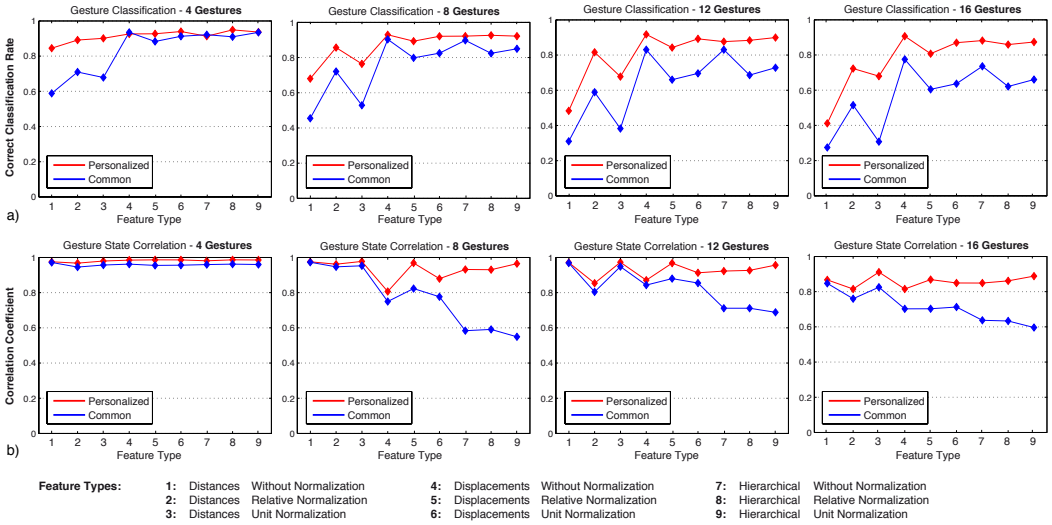


Figure 9.5.: (a) Correct gesture classification rates for different numbers of gestures and different input features, using personalized training data or one common training dataset for all testing persons. (b) Correlation coefficients as a measure of prediction quality for the gesture states.

worst in all settings (features 1-3). In combination with a relative normalization, there is an improvement (feature 2). The best performance is achieved by the displacement features without normalization (feature 4) or with unit normalization (feature 6). Hierarchical features almost match the results of the displacement features, without any significant difference between the normalization strategies (features 7-9).

9.2.3.2. Gesture State Prediction Quality

The second output computed by our gesture recognition component is the gesture state, given by the number $\hat{x}_t \in [0, 1]$. This value allows translating arbitrary spatial movements of the user to parameters of the target system. It is important that this value varies smoothly and that movements within each gesture translate to the full value range between 0 and 1. In order to evaluate the prediction quality, we measured the correlation between the computed signal of $\hat{x}_t$ over time with an ideal signal obtained from the ground truth data $\mathbf{X}^c$. Figure 9.6.b illustrates the meaning of this correlation measure. A maximum correlation of 1 is achieved if the change in $\hat{x}_t$ is aligned in time with that of the ground truth and if the amplitudes (i.e. the extent of $\hat{x}_t$ between 0 and 1) match exactly. Figure 9.5.b shows the resulting correlation coefficients for the series of experiments as described in the previous

section.

In the case of 4 gestures considered simultaneously, the correlation is close to the maximum of 1, with no difference between the feature types and with an almost equal performance with personalized and with common training data. Overall, the correlation between the reconstructed gesture state signal and the ground truth decreases with an increasing number of gestures. As expected, using personalized training data yields better results than when one training set is used for all testing persons. However, the correlation coefficients for the personalized case never drop below 0.8, even when the system is trained on 16 gestures. In terms of features, the distance features (features 1-3) perform better than in the classification experiments. Best correlation overall is achieved by the displacement

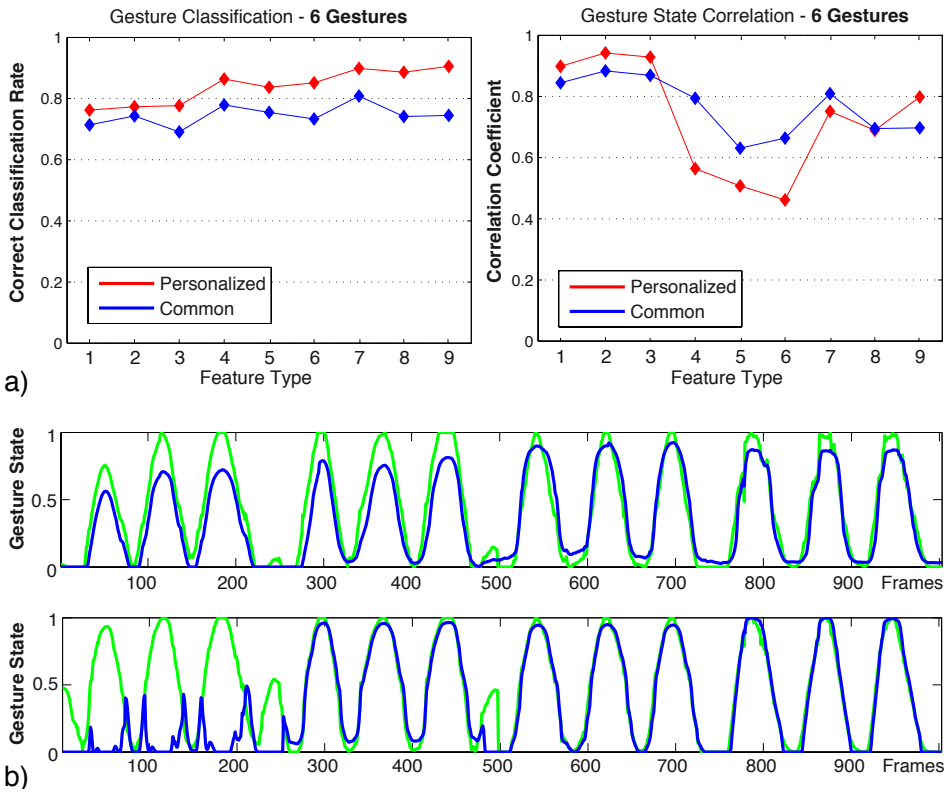


Figure 9.6.: (a) Correct gesture classification rates (*left*) and correlation coefficients for the gesture state (*right*); experiments on 12 testing persons and 6 gestures. Feature types see Figure 9.5. (b) Examples of gesture state sequences with prediction (*blue*) and ground truth (*green*). The top example: coefficient of 0.95; bottom example: coefficient of 0.68, as the first three movements are badly predicted.

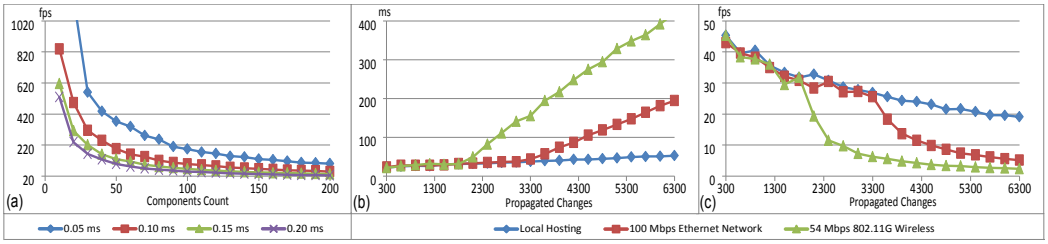


Figure 9.7.: (a) The effect of the hosted components count on the update rate. (b) The effect of the change propagation rate on update time, (c) frame rate.

features in combination with relative normalization (feature 5). Combined with the results on gesture classification, we conclude that the displacement features are the best choice among all presented feature types. For this reason, we used displacement features with relative normalization in our user study. Figure 9.6.a shows results for gesture type classification and gesture state correlation for the second dataset with 6 gestures and 12 persons. Classification rates are around 90% for personalized training and slightly below 80% for common training. These results are similar to those on the 3-person dataset, indicating that our method generalizes to multiple persons, even with a shared training dataset.

9.3. Framework Performance Evaluation

In order to evaluate the performance of the Signal.NET framework we also conducted several quantitative evaluations. We set up our framework on a machine with a 2.5 GHz Intel Core 2 Duo processor and 4 GB of memory. In order to evaluate the overall update rate of the system in regard to number of components, we used automatically generated pipelines with different number of components and let the system run for one minute using each configuration. The update rate is computed by averaging the recorded results. In order to simulate the processing time within each component, we used an internal timer for blocking the components for 0.05, 0.1, 0.15 and 0.2 milliseconds on their update call. The results are shown in Figure 9.7.

As shown in Figure 9.7.a the plots are performing as a hyperbolic function which highlights the linear dependency of the update rate to the number of components. Additionally, Figure 9.7.b and Figure 9.7.c show that update time and the update rate of the system in relation to the number of the propagated values over different network connections. Based on these plots, before reaching the upper boundary of the bandwidth, computational limitations are the only affecting parameters. However, the connection speed considerably

slows down the performance once its boundary is exceeded. Overall the system performs in realtime with both PCA and Manifold recognition techniques and using Kinect and inertial sensors. The overall framerate is bounded only by the frame rate of the Kinect skeleton tracker and inertial sensor data retrieval APIs.

9.4. Gesturing User Studies with Biomedical Students

To evaluate the proposed gesture-based interaction methods for the Operating Room, we first conducted qualitative experiments with biomedical computing students. The test subjects all were pursuing or hold a degree in biomedical computing field. Thus, the subjects were educated to develop intra-operative systems and had a very good understanding about the OR and medical scenarios. The interaction pipelines were hosted in the Signal.NET framework, where the gesture recognition components was a wrapped version of a Matlab implementation. This solution runs at 30 frames per second. In this section we report about the result of user studies conducted with different gesture recognition techniques.

All the experiments conducted in this section consisted of six phases.

1. Each user was given a short introduction about the purpose of the study, available functionalities of the medical image viewer application, the available gestures and the activation mechanism.

2. The participants were given several minutes of free exploration time, to get a feeling of the system.

3. We then explained our predefined test task. From a medical perspective, the goal of the test task was to locate an aortic stent in a loaded CT dataset and navigate to its bifurcation. This task is based on a real activity within aortic stent implantation surgery, where the surgeon needs to localize the implanted stent for validation. The used dataset has been acquired at our university hospital, Klinikum Rechts der Isat, and was anonymized accordingly. To support the users without any medical background we further provided the expected correct values for each of the 6 parameters (slice number, zoom ratio, view position in x and y, brightness and contrast).

4. Parameters were reset to initial values and each participant was asked to perform the test task using a common training dataset. This step was only performed using Kinect and PCA-based recognition method.

5. Personalized gestures were recorded and users were asked to perform the test task again with their personalized training data.

6. Finally, for the study with the Kinect, each participant filled a questionnaire designed based on AttrakDiff [61], once for each of the explored methods in their test session. AttrakDiff is a testing support tool to evaluate human-computer interfaces in terms of hedonic and pragmatic qualities. It consists of 28 word pairs selected to qualify the system as well as some demographic questions (age, profession, education level). For experiments with inertial sensors, users were asked to fill a questionnaire, designed to capture different usability aspects of the proposed gesturing techniques.

9.4.1. Inertial Sensors with Manifold

In order to evaluate the manifold-based gesture recognition technique with inertial secsors, we used four wireless inertial orientation sensors attached to the arms of our testing subjects. Gestures were defined for evaluation included movements with one or both arms, such as moving an arm horizontally or vertically, or tracing out a circle.

9.4.1.1. Study Design

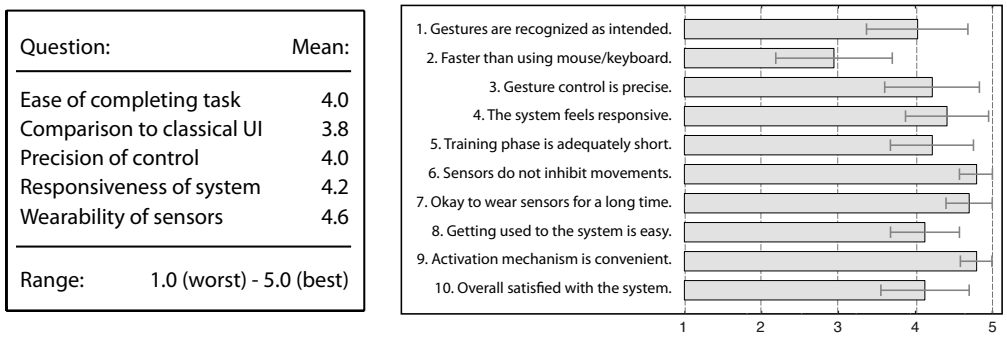
We conducted this user study with 10 biomedical students as test subjects to assess the usability of the proposed manifold-based gesture recognition method with using the inertial sensors. We used four Trivisio Colibri Wireless inertial orientation sensors, placing two on each arm of our test subjects, as suggested in [132]. The system was implemented in C++ and tested on a 2.5 GHz Intel Core 2 Duo machine with 4 GB of memory.

9.4.1.2. Qualitative Results

Average user answers in a questionnaire consisting of five questions are given in Figure 9.8.a. The questions were requested to be answered by selecting a number between 1 (worst) and 5 (best). Our evaluation shows promising results of using this method in our experimental setup. Very positive feedback was given to the wearability of the sensors and the responsiveness of the system. The achieved precision of gesture-based control and the overall ease of completing the given task were also acknowledged.

9.4.2. Inertial Sensors with PCA

We conducted another user study to evaluate the usability of the PCA-based gesture recognition technique providing the inertial sensor data. we used a medical image viewer ap-



(a) (b)

Figure 9.8.: Average questionnaire results after qualitative user study for inertial sensors. The values indicate subjective consent to the phrases given on the left, on a scale from 1 (*disagree*) to 5 (*agree*). (a) With 10 test subjects and Manifold gesture recognition technique. (b) With 10 test subjects and PCA gesture recognition technique. Standard deviations for each item are shown.

plication as the target system. This application lets the user load and browse medical volumetric image data. We used a subset of 8 different gestures, each assigned to one of the main features of the image viewer, including slice number, zoom factor, view position, brightness, contrast, etc. For allowing users to activate and deactivate gesture control, in this study, we implemented two exemplary techniques: a voice recognition module reacting to the commands ”start” and ”stop”, and a hand-held wireless switch. These solutions were designed using the Signal.NET framework.

9.4.2.1. Study Design

Ten biomedical students as test subjects participated in our study with an average age of 28 years, three of them female. Among them, 4 were senior master students, 4 PhD students and 2 post-docs. After providing basic instructions to each participant, we recorded training data for every gesture with three repetitions. To reach the target values, the users had to use the trained gestures. In this user study, the procedure was performed twice per participant, once using voice recognition and once the hand-held switch to activate and deactivate gesture control. We recorded the resulting values that users set using each of the methods and computed the deviation of each of the parameters from the target values. At the end, the users were asked to fill a questionnaire.

9.4.2.2. Qualitative Results

As illustrated in Figure 9.8.b, our gesture-based interface using PCA method also has received positive feedback. The results show that the sensors are not inhibiting the usual movement of the users and could be worn for extended periods of time. Furthermore, most of the users agree on the responsiveness and precision of the proposed system. The more skeptical response to the speed comparison with respect to using keyboard and mouse (question 2) can be explained with the fact that the participants have a bias towards using familiar classical interfaces in WIMP environments. During our study, we noticed that users liked especially being able to move freely within the room. Regarding the activation, eight of the ten participants preferred the hand-held wireless switch over the voice recognition system. This might be due to the fact that users feel faster response and more direct control over the system, using a switch compared to the voice command. In both approaches, the participants achieved an acceptable accuracy in setting the target parameter values. Using the voice command method, users reached the expected parameter values with a deviation of 2% of the respective parameter range. For the hand-held switch, precision was within 4% of the parameter range. This gave enough freedom to the users to accurately control the target application. Moreover, initial experiments in a real Operating Room environment provided encouraging results and show that the proposed gesture-based interface has good usability factors for an applicability in practice.

9.4.3. Kinect with PCA

In order to evaluate the usability of our PCA-based gesture recognition technique with Kinect, we conducted a user study, explained in this section. Gestures were defined similar to those which were used in studies using inertial sensors, including movements with one or both arms, such as moving an arm horizontally or vertically, or tracing out a circle.

9.4.3.1. Study Design

We conducted a user study with 12 biomedical students as participants (9 males and 3 females with different levels of computer experience) aged from 23 to 35. No additional selection criteria were considered for selecting users. None of the subjects had experience with 3D gesture-based interfaces, except few hours on Xbox using Kinect for three persons. We used a subset of 6 different gestures, each assigned to one of the features of the image viewer. For gesture activation, we used a voice recognition system which activates the system on ”control” and stops it on ”set”.

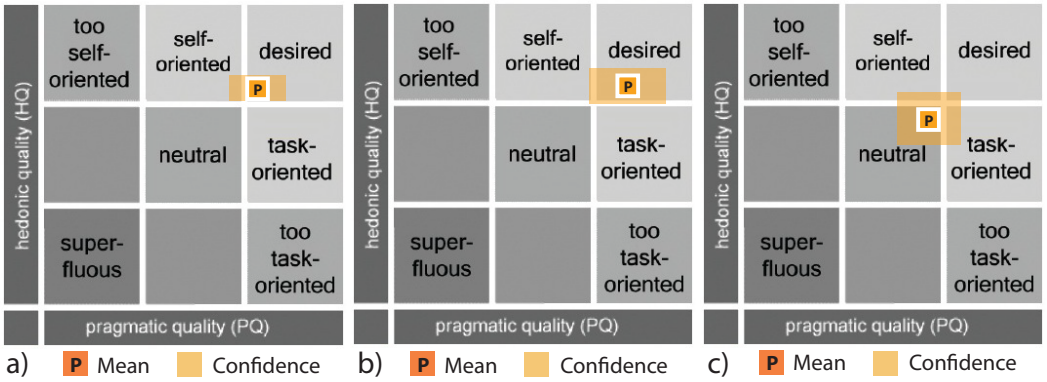


Figure 9.9.: (a) Results of the user study for the gesture-based interface using the common training data, Kinect and PCA-based recognition technique. (b) Results using personalized training data. (c) Results using the method based on inertial sensors.

9.4.3.2. Qualitative Results

All the test subjects were able to successfully complete the test tasks. We generated three graphs, shown in Figure 9.9, to illustrate the results of the user tests with and without personalized training, and compare the user experience with inertial sensor-based method. These diagrams place the systems in terms of pragmatic quality (PQ) and hedonic quality (HQ). PQ addresses different aspect of human needs related to control, learnability and ease of use [62]. In other words, PQ demonstrates the softwares usefulness, expressed by product characteristics such as ”clear”, ”supporting”, and ”controllable”. Furthermore, HQ deals with human desires for excitement including novelty and satisfaction. This refers to the softwares quality aspects such as ”innovative”, ”exciting”, and ”exclusive”.

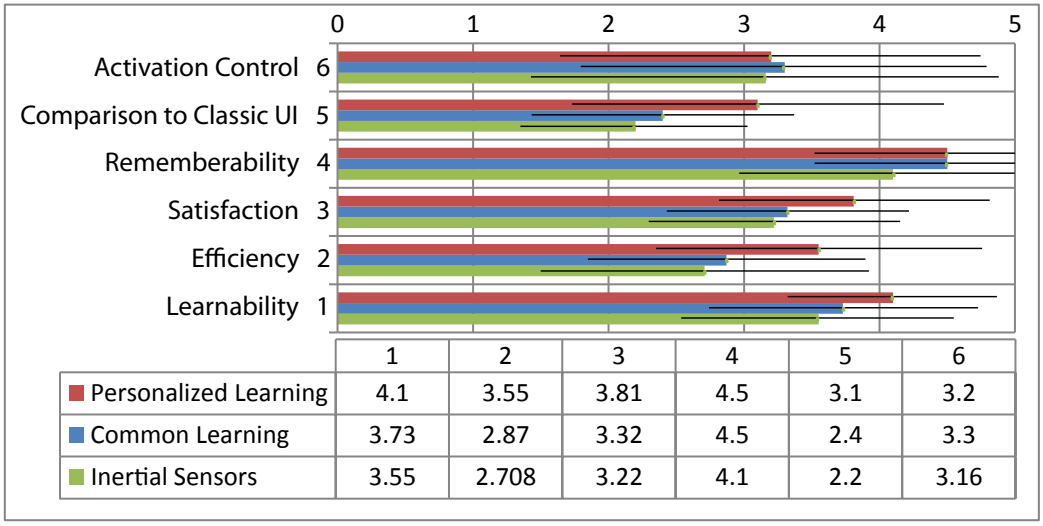
The gesture-based interface without any personalized training was categorized as ”desired”. This means that, on average, the system performs acceptable both in terms of pragmatic and hedonic qualities. The confidence rectangle is larger for the pragmatic dimension, which means that the participants opinion was more diverse about the level of usability. On the other hand, the overall impression is that this system is attractive and inspiring and most of the users agree about this. According to our users, the system is ”manageable”. The results for our gesture-based interface using personalized training data has been also categorized as ”desired”. Compared to the common training data approach, the hedonic quality is equal but the pragmatic quality is higher. Based on the overall impressions of the users, the system is attractive. Furthermore the confidence area is larger

for both PQ and HQ. This means that users have more different ideas about usability when an additional training is required. Positive points about the system are ”manageable” and ”inviting” and the negative point is ”cumbersome”, which based on our observations during the studies is related to the additional required time for training. The users also liked the fact that they could freely move around without any boundaries (in the Kinect field of view). The gesturing method using inertial sensors is categorized as ”natural”, meaning that the experience was acceptable for the users. However, both PQ and HQ aspects are rated less than the methods using Kinect as it requires users to wear inertial sensors in advance.

In addition, we asked users to fill a questionnaire including several questions related to different usability components such as learnability, efficiency, satisfaction and rememberability. Two more questions were also added about the used activation control and a comparison of this gesture-based method with classical user interfaces. Users were asked to answer with a value between 1 to 5, corresponding to disagree and agree. Figure 9.10 shows the average and standard deviation of these answers. According to the users, the system is more efficient, learnable and satisfactory when using personalized training data. Moreover, the presence of personalized training data makes it more comparable to classical user interfaces. Overall, large standard deviations highlight the fact that the users have different opinions about these factors. Highlights include the high scores for learnability and rememberability, as well as an overall better rating for the proposed gesturing interfaces utilizing Kinect, as compared to the sensor-based gesturing approach. In general, voice based activation was not desirable for some of our users, due to required additional effort and time.

9.5. HCI Design User Studies with Surgeons and UI Experts

In order to evaluate the applicability of the proposed gesturing method and customization concepts using Signal.NET framework, we have conducted user studies with two groups of potential users, namely surgeons and experts in the field of human computer interactions. The test setup was prepared to simulate an exemplary case of the stent implantation surgery. The required intra-operative devices for this operation were available running on simulators, including an IVUS image viewer, Ultra Sound device, 3D DICOM image viewer and a dim controller for the OR light. All these simulators were connected to the Signal.NET framework, using their target components. The framework was equipped with Kinect and the gesture recognition technique based on PCA. The displacement features



	1	2	3	4	5	6
Personalized Learning	4.1	3.55	3.81	4.5	3.1	3.2
Common Learning	3.73	2.87	3.32	4.5	2.4	3.3
Inertial Sensors	3.55	2.708	3.22	4.1	2.2	3.16

Figure 9.10.: Usability results collected in form of questionnaire, using Kinect and PCA-based gesture recognition technique.

with relative normalization was used as feature extraction technique. The test task consisted of three parts, representing phases one, three and four of the proposed methodology in Chapter 8. First, surgeons were asked to plan the interaction requirements for an aortic stent implantation surgery, by filling a form similar to Figure 8.12. In the second part, users were asked to design the interaction pipeline as planned. In the third part, users were asked to interact within the simulated OR, evaluating the configured gesturing interface. The second and third steps were performed both with surgeons and UI experts. After completing these steps, users were asked to fill a questionnaire about each part. Figure 9.11, contains some of the main questions included in our questionnaires. For the second and third parts, two additional AttrakDiff [61] questionnaires were also filled by the participants. AttrakDiff is a standard testing support tool to evaluate human computer interfaces in terms of hedonic (HQ) and pragmatic (PQ) qualities. It consists of 28 word pairs, some shown in Figure 9.12.a. The two graphs, shown in Figure 9.12.b, illustrate the results of these surveys.

9.5.1. User Study with UI Experts

The proposed visual editor can be used by OR technician or deployment team of medical device manufacturers. Such users are familiar with the main concepts of HCI. To evaluate the proposed ideas from the perspective of these users we conducted a user study in the

context of a demonstration in ACM international Conference on Intelligent User Interfaces, took place between 14th and 17th of February 2012 in Lisabon, Portugal. This conference, each year brings together hundreds of UI experts. Within the 3 hours of demonstration, 18 experts participated in our study (age mean:35.5, σ:8.5) and (mean of HCI experience in years:9.1, σ:5.3). The generated AttrakDiff reports categorized the visual editor and the gesturing interface as "desired" for UI experts. This means that, on average, both interfaces perform acceptable equally in terms of pragmatic and hedonic qualities. The confidence rectangle is larger for gesturing interfaces, which means that the participants opinion was more diverse about it. As shown in Figure 9.12.a, UI experts have the impression that the customization concept is practical and straightforward. Based on the collected data in questionnaires, Figure 9.11, they further highlight that the system is fun to use. Additionally, the learnability and rememberability of both interfaces are rated more than 80 % for experts of the HCI field. The interface for customization of user interactions also mentioned to be applicable in collaborative environments with presence of workflow and multiple target systems as the OR domain. Analyzing the feedbacks collected from these participants, we noticed that such environments to customize the UI are also interesting for other domains.

9.5.2. User Study with Surgeons

This user study was conducted inside a real OR at the department of vascular surgery of Klinikum Pasing in Munich, Germany. Three surgeons with different experience levels participated in our study (one assistant physician with 3 years of experience, one senior

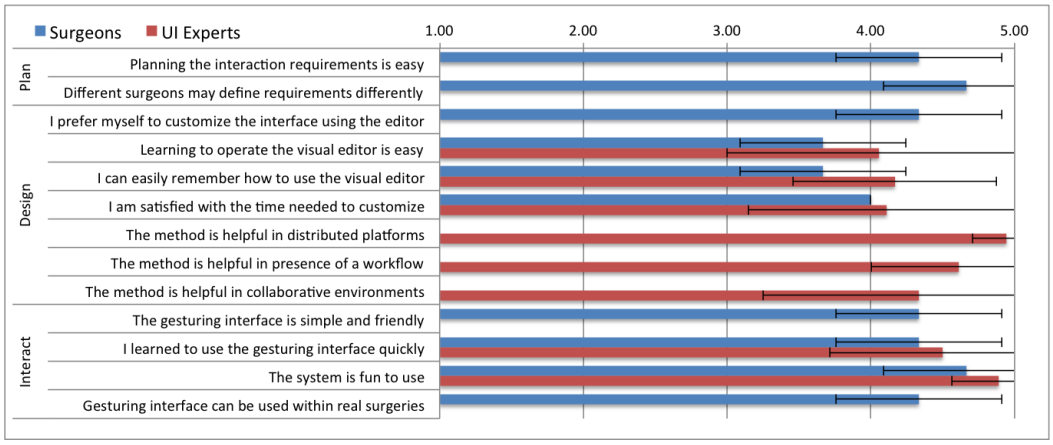


Figure 9.11.: Selected questions from the questionnaires: (1: Disagree) - (5: Agree).

physician with 13 years of experience and one chief physician with 25 years of experience). The aim of this user study was to evaluate the proposed concepts from the perspective of its real end-users. Based on the AttrakDiff reports, the gesturing interface is categorized as "desired" for surgeons. The small confidence area for gesturing interface means all the surgeons agree about the applicability of the gesturing interface in the OR. However, lower pragmatic quality for the visual editing environment shows that the customization interface is hard to learn and use for surgeons. As shown in Figure 9.12.a the gesturing interface is novel and presentable from surgeons opinion. Based on the questionnaires, Figure 9.11, the planing phase was not challenging, however different surgeons may define the requirements differently. This further proves the need for a customizable interface for the OR. Although surgeons mentioned that they prefer to use the visual editor themselves, Figure 9.12.b, learning and remembering the details of the visual editor is harder for them compared to the users familiar with the HCI concepts. This suggests that customization is better to be performed with OR technicians or by the manufacturer, upon deployment of a new OR in a hospital.

9.6. Discussion

Several points can be discussed regarding the aforementioned quantitative evaluations. The experiments shows that using our customizable gesture recognition approaches, the surgeon has the freedom to define gestures as desired. Our experiments show that these approaches are able to robustly recognize learned gestures and can simultaneously recognize several gestures to a high accuracy. While novel depth imaging modalities, such as the Kinect, are a straightforward option for gesture-based interaction, we argue that iner-

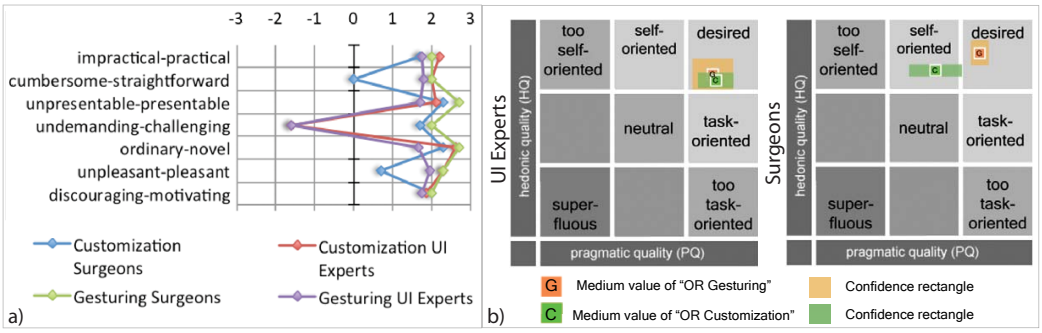


Figure 9.12.: AttrakDiff reports: (a) Selected word pairs. (b) Generated HQ-PQ diagrams.

tial sensors are more suitable for the OR scenario. Surgeons typically alternate interaction among multiple devices distributed in the OR, e.g. patient monitoring systems or imaging devices, such as ultra-sound or X-ray. At each moment, the surgeon looks at a different display based on personal requirements. With a Kinect solution, one device would have to be added per display to capture gestures. The presented concept can be exploited to interact with different devices without any line-of-sight problems. Inertial sensors also do not limit the surgeon in moving within the OR. In addition, each sensor has a unique ID, allowing an easy distribution of sensors over multiple persons. In terms wearing inertial sensors, it is important to note that, during their preparation, surgeons have to undress completely and wear specific sterile OR clothing. Adding ergonomic, small sensors will thus not be an additional burden in general. This might be different in cases such as radiology, where doctors only wear a cover on top of their usual clothing. Regarding the drop of recognition rate while increasing number of trained gestures, we argue that this problem is tolerable in practice. Based on the results, the recognition quality for up to 8 gestures, is proper for interaction purposes. Furthermore, this set of well-recognized gestures can be dynamically bound to different target features based on the usage context. Considering the quantitative evaluations results using Kinect, among the examined features, the displacements perform best. The normalization approaches do not have a significant effect on gesture recognition rates but account for some improvement in terms of gesture state correlation. We therefore used the displacement features with relative normalization in our user study with Kinect. This configuration is invariant with respect to the user's location in front of the Kinect and invariant with respect to different users' body proportions.

Regarding the performance, all the proposed recognition methods run in realtime on a standard computer for up to 18 gestures using Signal.NET framework. The computational complexity is constant and since the recognition method performs separately on each frame, it is further independent of the complexity of the movements. The dimensionality reduction step in the gesture recognition methods serves the purpose of creating an internal representation of body poses, captured by the inertial sensors or Microsoft Kinect. An end-user can think of this process as a translation from body poses to a single value in the range [0,1] that is general enough for being mapped to arbitrary parameters of a user interface. Dimensionality reduction simply compresses the full pose information to this one-dimensional representation. Moreover, the recognition algorithm and its implementation can be deployed in the application logic in such a way that users can utilize the gesturing interface, simply after recording an instance of each gesture. This step can be omitted as well, using common training data for Kinect. Overall the implemented method in this section showed that the use of the proposed gesture recognition technique, besides

decreasing the development effort required for creating a flexible 3D gesture-based interface, provided by Signal.NET framework, can produce acceptable results for the end-users.

Such gesture-based interaction in the OR has been explored in several studies, e.g. [81], where the authors point out the clear need for touch-less interaction in the OR based on interviews with surgeons. We found a similar interest from surgeons who participated in our user studies. The results of the user studies highlight that the proposed gesture-based interaction system performed acceptably. Most of the users mentioned that it is an interesting way of interaction with a system and all of them accomplished the given tasks in all tests. Nevertheless, the result for the comparison with a classical UI are still negative. This can be argued with the fact that most of the participants have a bias towards using familiar classical interfaces in WIMP environments. The qualitative experiments also highlight that a higher level of performance and satisfaction is achieved using personalized training data. Although this issue is mainly rooted in the fact that different people perform the same gesture in dissimilar ways, the effect of the noise presented in the Kinect and inertial sensor data should not be underestimated, which makes the movements even more diverse. A more controlled user training phase can reduce the former issue, while enhancing the quality of the sensors and de-noising the input signals are possible solutions for the latter issue. Although the voice-based activation control gives the users the opportunity to freely use both hands for gesturing, our participants were concerned about the additional time they had to wait while pronouncing the voice commands, e.g. ”control” and ”set”. A more direct control over the activating and deactivating process was desired by some users. This suggests utilizing devices like foot pedal for activation in the OR. Such pedals are commonly used in the OR for sending command to different devices such as X-Ray C-Arm, as their sterility is not critical. In fact, activation is a common problem in the field of touch-less human-machine interaction [66]. Our comparison between Kinect and inertial sensors in terms of end-user satisfaction, shows that the Kinect-based technique is more satisfactory to the user. Several medical professionals observed and experienced our system beside our user studies, including a neurosurgeon and a trauma orthopedist from Klinikum Rechts der Isar, with 1 and 3 year(s) into their residency, as well as a minimally invasive abdominal surgeon and a vascular surgeon, both with more than 20 years of experience from the same hospital. In a qualitative feedback, these surgeons pointed out their interest in this interaction methods and their ability to see this solution as a possible UI for the OR. Additionally, the evaluations show that the customizability of the UI in regard to surgical workflow and OR roles, etc., that is brought in by Signal.NET framework, has been highly appreciated by surgeons, who otherwise are faced to predefined and static UI, offered by providers. This clearly shows the impact of this framework. These user studies

demonstrate that the proposed framework can be used in practice to define user interfaces in collaborative environments such as OR, where the behavior and the system response can be adapted based on the current workflow stage and individual user requirements. The promising results of our user studies encourage a practical application of these methods in developing of 3D gesture-based user interfaces for the OR.

9.7. Summery and Conclusion

This part of the thesis (last three chapters) proposed a novel gesture-based interaction method for the OR using wireless inertial sensors, Microsoft Kinect and Signal.NET, a specialized software framework to manage and host the interaction pipelines based on main complexity aspects of the OR. The proposed gesture recognition techniques are both categorical and spatio-temporal. During a training phase, gestures are learned for each considered gesture, based on the sensor data. This allows customization for each individual user within different workflow phases. Using a proper feature extraction method, the Kinect training data can be shared among different users. This eliminates the additional training phase per user if it is not desired. Simultaneous recognition of gesture type and gesture state can facilitate the development of flexible 3D gesture-based interfaces. By integrating this techniques into the Signal.NET framework, a high level of customizability can be achieved without any hard-coding. Having such a dynamic mapping between the gestures and the functionalities of a target system raises new possibilities for developing context-aware user interfaces for the OR. Our qualitative and quantitative evaluations, presented in this chapter, show promising results of using these methods in our experimental setups. While the automatic differentiation between learned gestures and other movements achieves true positive rates above 90%, an additional tool, e.g. a pedal or voice command, could be used to activate and deactivate automatic gesture-based control within critical workflow stages. The conducted user studies, further highlight the need for such approaches from surgeons perspective and prove the applicability of the proposed concepts based on the UI experts. In this work, we have considered the OR as the main application scenario. However, we believe that this approach, with minor modifications, can be also applied to other collaborative workspaces, which can be characterized by a workflow model.

Part IV.

Final Conclusions

CHAPTER 10

Conclusions

THE present chapter summarizes the main contributions and findings of the research conducted in this work. Finally, it is concluded by discussing the remaining challenges, future directions and ideas worth investigating.

10.1. Summery

The main objective of this work is to improve human computer interactions inside the OR. Our literature study, revealed the fact that this is a well known problem and has a considerable effect on the patient outcome and the overall success of medical interventions. We argued that, this problem is rooted in the lack of usability of intra-operative solutions. As a solution, we closely studied the OR domain in close collaboration of surgeons and domain experts. Our investigations demonstrate that the OR is a complex domain, having several key characteristics of such domains. To best of our knowledge, this is the first work closely studying the OR as a complex domain. Such complexity adversely affects the development of usable intra-operative devices, thus a novel and generic modeling technique for complex domains is introduced. This work shows how such a model can be specifically defined for the OR to capture main complexity aspects within this domain. The proposed model contains detailed information about the target system and its environment. Surgical workflow, collaborative environments with various human roles and compound intra-operative solutions are introduced as the main complexity sources in the OR. As the facets of these different views are tightly bound to each other, the concept of mapping tables is intro-

duced to capture such correlations. Based on the proposed modeling technique, a surgery specific domain model can be constructed for each type of medical intervention, in close collaboration of domain experts. We hypothesized that such a conceptual OR domain model can provide a common understanding for the development team. As a proof of concept, we demonstrated how the proposed OR domain model can be utilized in the design and development of two supportive frameworks, namely OR-Use and Signal.NET. These frameworks lay the foundation for further studies in the field of multi-center usability studies for intra-operative solutions and unified multi-modal user interface development in the OR. To best of our knowledge these are the first frameworks in their type which have been constructed, specifically for the OR domain. The following sections further summarize the outcomes of these projects.

10.1.1. Usability Evaluations

Usability is seen as a crucial factor for the commercial success of many products and even it is more important for intra-operative solutions as usability deficiencies in such equipments can cause fatal errors. Evaluating the usability in complex domains, such as OR, is more challenging compared to many other domains, e.g. websites and mobile applications. This is due to the fact that the usability specialists require deep domain knowledge, which they usually cannot acquire. Consequently, usability professionals face a variety of challenges during usability test conduction, acquisition and management of usability data, as well as analysis of huge amount of data. Having a tool for supporting different aspects of usability testing can increase testing efficiency and improve the usability of intra-operative devices. In order to define the requirements of a usability testing support tool, we performed a requirement analysis by studying the similar solutions in other domains. This study was performed in close cooperation with domain experts and manufacturers of intra-operative devices. Ideally, usability testing support tools should capture a wide range of inputs, manage and organize the collected data and support analysis of the usability data via a proper set of data retrieval and visualization interfaces.

Based on the introduced requirements, we proposed an architecture, which is used in development of the OR-Use framework. To facilitate the data acquisition stage of the usability testing, the design of a portable annotation tool is introduced, which has been defined based on the real activities that evaluators perform in the OR. Such tool should be highly usable itself, since any interruption of the surgery by the evaluators might be disturbing for the OR team. To achieve the high usability in this tool, the introduced OR domain model is used to dynamically generate the UI elements, specifically for a given

type of surgery. Our user studies revealed the usability of this approach. Furthermore, user studies showed that voice-based annotation is more desired for the evaluators. However, using this feature requires surgeons' permission to avoid any disturbance for the OR team. In absence of the voice-based documentation, traditional on-screen keyboard is shown to be equally practical. Similar to other medical instruments, patient anonymization and data security are very important in such a tool. Required security is achieved by storing only patient id and encrypting the collected data.

Other usability data such as device specific logs and recorded videos can be also collected using the proposed architecture. The collected data can be transferred and stored in a centralized usability data management service. Our experiments show that the data transmission bandwidth is the main bottle-neck of such a design, as usability data usually contains bulky recorded videos and supplementary files. On the server side, utilizing an extendible design pattern can facilitate addition of new type of usability data, which is necessary while using such framework for evaluating a new device. This is important as often intra-operative devices come with several internal data formats, e.g. reconstructed models, overlayed videos, tracking information, etc. Regarding data management, the proposed OR domain model can be used in order to organize huge amount of data, collected within several test sessions in multi-center studies. With the proposed data binding technique, each part of usability data is tagged with facets within the OR domain model and therefore can be traced from different perspectives captured in the model. Such highly organized data can be intuitively queried and extracted from the server using the proposed query format. As the response is generated in XML format, such queries may be executed from any web browser connected to internet, providing world-wide access to the collected usability data.

The OR domain model can be also a basis for usability data visualization. Such visualization is introduced enabling detailed navigation along with browsing within usability data. With this technique, the analysis task can be performed in fraction of time which is required in traditional heuristic usability data analysis approaches. Recorded videos during the test sessions also can be indexed using the data elements which themselves are bound to the OR domain model. With this technique an analyser can directly play all the video segments, relevant to a specific portion of the data. This concept can be considered as the next generation of the timeline-based video indexing techniques, which are currently considered as the state of art in usability data visualization. Our studies reveal that the concepts introduced in the OR-Use framework are practically applicable and additionally utilize finding of critical usability problems, hidden inside the data collected within multi-center studies. Moreover, OR-Use framework can be used to measure classical usability

components such as learnability, efficiency and subjective satisfaction. As a benefit of utilizing OR domain model, all the usability related aspects of a system can be studied and evaluated from different viewpoints of the model. The proposed framework architecture helps to overcome the challenges of usability testing in the OR domain. Signal.NET has been used within several usability studies, up until May 2012, and it successfully facilitated the evaluation process.

10.1.2. Human Computer Interactions

Usage of traditional touch-based interaction techniques are limited in the OR, due to sterility regulations. As a solution, in this work, we suggest gesture-based interfaces. Initially, we proposed the novel concept of categorical and spatio-temporal characteristics of a gesturing interface. Then, we introduced two methods with these properties which allow surgeons to define a personalized set of gestures in a training phase that best fits their particular requirements. Apart from automatically recognizing performed gestures, these methods simultaneously track the relative pose within each gesture. The conducted quantitative experiments demonstrate that these approaches practically recognize both gesture type and gesture state in real-time with a recognition accuracy up to 90% for 8 user-defined gestures. These gesture recognition techniques are based on dimensionality reduction methods namely, PCA and Manifold learning. Our experiments show that there is no considerable difference in correct recognition rate between PCA-based method and the technique based on Manifold learning. Furthermore, our presented smoothness modification for kernel regression decreases the sensitivity to outlier sensor data and thus improves gesture recognition rates. Different input modalities can be used for tracking users such as inertial sensors and Kinect. We argue that although Kinect is more pleasant to users as they do not have to wear any additional sensors on their body. However, due to uneven lighting condition and line of sight problem, inertial sensors are a better choice for the Operating Rooms. Using a proper feature extraction technique, we demonstrated that the system can perform independent of the sensors placement. Moreover, the very same training data can be shared among several users, to avoid additional training.

In order to utilize these gesture recognition techniques, we also described the design of Signal.NET, a specialized multi-modal HCI designing framework. The Signal.NET framework enables a dynamic association of learned gestures to arbitrary target systems, based on the main aspects introduced in the OR domain model. The used component-based design pattern enables fusion of intra-operative interfaces in a unified gesture-based interface. Thanks to the pipeline model, which enables data transition between different components,

a separate gesturing interface can be defined for each user within each stage of the surgical workflow. Moreover, a binding can be implemented as a remote data connection over the network, enabling the inclusion of several applications running on different machines. A graph-based representation of the interaction pipeline can provide the end users with a high level visual programming interface for customization purposes without any programming knowledge. This visual editing interface can utilize the defined workflow stages and human roles, as defined in the OR domain model, in order to enable the design of context aware interfaces. The quantitative evaluations performed on the Signal.NET revealed that the proposed architecture performs in real time, without any noticeable latency, using Ethernet or wireless networks.

We additionally proposed a methodological approach to bring such customizable gesturing interfaces into the OR. To best of our knowledge, this methodology is the first of its type for customization of intra-operative user interface, based on the real requirements of a specific surgery and surgeons decision. In order to prove the applicability of the proposed ideas, we performed two user studies based on a real scenario with Surgeons and HCI experts. The collected qualitative results demonstrate that the idea of customized gesturing interface for the Operating Room is interesting for the surgeons and they can imagine it as the future of the HCI in the OR. The user studies also demonstrate that our prototype is feasible and robust enough to be used in practice. Regarding the planning stage in the proposed methodological approach, surgeons mention that it is easy for them to plan the interactions requirements in advance. However, different surgeons may define it differently, mainly due to different levels of experience. Moreover, surgeons prefer to perform the designing stage on their own. However, the results of user studies demonstrated that such visual programming interface is much easier to learn and remember for people with more experience in the field of HCI. The UI experts also point out that this approach is a practical way of defining context-aware gesturing interfaces for a complex domain such as the OR. The promising results of our user studies encourage a practical application of our method in the Operating Room.

10.2. Future Work

The proposed modeling technique can be improved by investigating and considering other aspects influencing the OR domain, such as patient background. In our current setup, a surgery specific domain model should be created manually, providing an XML file with a given schema. This process can be improved by utilizing available semi-automatic tools

designed for annotating surgical interventions [104] or with fully automatic surgical workflow mining approaches such as [16]. The OR-Use framework should be evaluated in further details, within more usability studies in the OR with different types of devices. Another very important step toward supporting the analysis of usability testing in the Operating Rooms and more generally in complex domains is to study different data mining techniques. Such solutions possibly can prioritize the usability comments in order to provide an automatic or semi-automatic decision support interface for analysers. Theoretically, this is possible as all sections of the acquired usability data within the OR-Use framework are tagged with the relevant aspects of the OR.

In our current implementation of gesture recognition methods, gestures that are too similar cannot be distinguished by the system. Within this work, we asked the users to perform gestures that have a minimal mutual overlap. However, future work will include an automated method which measures the quality of the training dataset and alerts the surgeons in case of gestures that are too similar to be differentiated. Beside the PCA and Manifold learning, other dimensionality reduction techniques should be investigated, in order to improve the correct gesture recognition rates. Improving the activation control and testing the recognition method with other feature representations are also among the future work. In the Signal.NET framework, the interface of different intra-operative devices can be included, such as C-Arm and SPECT device. This framework should be also evaluated in more detail within the practical usage inside the OR. Additionally, Signal.NET framework can be improved by integrating a usability assessment and input device performance evaluation tools like [121]. One option would be to integrate Signal.NET inside the OR-Use framework. Future work can also focus on embedding interaction design techniques like [59] and also recommender systems for interaction pipelines, based on different aspects of the OR specific domain model including the type of surgery or experience of the OR crew. At the end, making these projects available as open-source, can provide several possibilities for multi-disciplinary researches with participation of other groups.

CHAPTER 11

Glossary and Acronyms

Glossary

Activation Control. A mechanism to prevent movements similar to the trained gestures be considered as commands.

Adverse Event. Injuries caused to patients as the result of a medical intervention rather than the underlying medical condition.

Demultiplexer. Acts as a switch between the recognition components and target components, forwarding the recognized gesture state to the relevant property of the target component.

Displacement Feature Representation. Represents each joint by its displacement vector from the center of the human body.

Distance Feature Representation. Represents each joint by its distance to the center of the human body.

Domain Model. A conceptual model of a family of systems which describes the various entities involved in those systems and their relationships.

Efficiency. One of the usability components which allows a system to be used with high level of productivity.

Feature Extraction. A technique to represent the acquired input sensor data inside the proposed gesture recognition approaches.

Feature Normalization. Makes the features independent of the body styles and proportions.

Feature Representation. Extracts data in a specific way from the original sensor data

Gesture. User body movements in order to interact with a computerized solution.

Hedonic Quality. Deals with human desires for excitement including novelty and satisfaction. This refers to the softwares quality aspects such as ”innovative”, ”exciting”, and ”exclusive”.

Hierarchical Feature Representation. Represent each joint by its displacement vector from the parent joint.

Human Role. Distinct roles that can be assigned to each member of the OR-team.

Inertial Sensors. A sensor type that can measure its orientation with respect to the earth reference frame.

Information Visualization. A technique that provides insight about the underlying patterns in presented data.

Interaction Pipeline. A graph including all the hosted components and their bindings, which defines the behavior of the system in each stage of the surgical workflow.

Learnability. One of the usability components which makes a system easy to learn for novice users.

Mapping Table. Models the correlations between various views in the constructed domain model in complex domains.

Memorability. One of the usability components which makes it possible for casual users to return to the system after some period of not having used it, without additional training.

Multicenter Study. A clinical trial that is conducted in several clinical centres by different evaluators.

OR Specific Domain Model. A domain model which is based on the surgical workflow, human roles and intra-operative devices of a specific type of medical intervention.

Performance Log. Recorded users interactions with different features of the interface.

Pivot. A data visualization toolkit which enables viewing and browsing large amounts of data, published by Microsoft.

Pragmatic Quality. Demonstrates the softwares usefulness, expressed by product characteristics such as ”clear”, ”supporting”, and ”controllable”.

Recognition Component. Hosts the implementation of the gesture recognition approaches proposed in this work.

Relative Normalization. A normalization technique which makes the features independent from the height of the users.

Sensor Component. Encapsulates a specific type of input sensors such as Kinect or inertial sensors.

Sensor Data. In this work, we refer to the data acquired with an input device from users interactions as sensor data.

Service Contract. A collection of operations that a web service exposes in the internet.

Signal. An abstraction of typical properties, methods and events composed by components.

Sociotechnical Domain. A domain which is composed of a technical system, surrounded by the workers interacting with it. Furthermore, these type of domains are influenced by the organizational or management structures, and their environmental context.

Subjective Satisfaction. One of the usability components which makes a system pleasant to use.

Surgical Workflow. The workflow of an intervention encompasses the common steps of one type of surgery.

Target Component. Wraps the interface required to control a specific intra-operative software or hardware.

Testing Methods. All usability evaluation methods which involve test users working with the target system within a controlled environment.

Thinking Aloud. A testing technique in which test subjects continuously verbalize their thoughts, while interacting with the system.

Video Indexing. Provides a mean to retrieve the corresponding video segments for each piece of usability data such as user comments or performance logs.

View. The first entity of the proposed domain modeling technique in this work which captures a single source of complexity in the domain.

Unit Normalization. A normalization technique which makes the features independent from the height and body proportion of the users.

Usability. acceptability of a system to its end users which can be measured with several factors including learnability, memorability, efficiency and subjective satisfaction.

Usability Data. In this work we refer to all the data which has been collected during usability testing as usability data.

Usability Engineering. A methodological process to improve the user interface of the system under the test.

Acronyms

2D Two Dimensional

3D Three Dimensional

CT Computed Tomography

CTA Computed Tomography Angiography

CXML Pivot Collection eXtensible Mark-up Language

DB Database

DBMS Database Management System

DICOM Digital Imaging and Communications in Medicine

DMS Data Management Service

DTW Dynamic Time Warping

FDA Food and Drug Administration

FK Foreign Key

FPR False Positive Rate

fps Frame per Second

GHz Gigahertz

HCI Human Computer Interaction

HQ Hedonic Quality

HTTP Hypertext Transfer (or Transport) Protocol

HUA HUIA

IIS Internet Information Service

IR Infra Red light

Isomap Isometric Feature Mapping

IVUS Intra-Vascular Ultrasound

JIT Just In Time (Generator)

JNI Java Native Interface

kNN k-Nearest Neighbours

MB Megabyte

MHP Model Human Processor

ML Manifold Learning

MREG regression with smoothness modification

MRI Magnet Resonance Imaging

MVC Model View Controller

OR Operating Room

PCA Principle Component Analysis

PK Primary Key

PQ Pragmatic Quality

RDBMS Relational Database Management System

REG plain kernel regression

ROC Rate of Classification

SAV Save Tool

SLN Sentinal Lymph Node

SLNB Sentinel Lymph Node Biopsy

SPECT Single-Photon Emission Computerized Tomography

SQL Structured Query Language

TCP Transmission Control Protocol

ToF Time of Flight

TPR True Positive Rate

UI User Interface

URI Uniform Resource Identifier

US Ultrasound

WBT Web-based Tool

WBX Web Exact

WCF Windows Communication Foundation

WIMP Windows, Icons, Menu, Pointer

WSDL Web Services Description Language

WUP Web Usability Probe

XML eXtensible Mark-up Language

XSD XML Schema Definition

CHAPTER 12

Domain Model of SLNB Using the SPECT Imaging Device

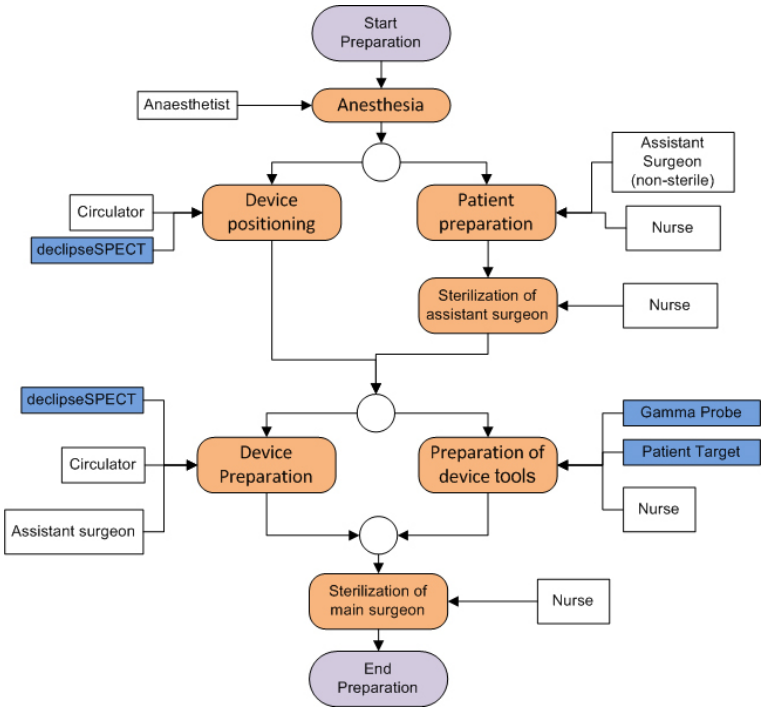


Figure 12.1.: SLNB Workflow using the SPECT device - Preparation

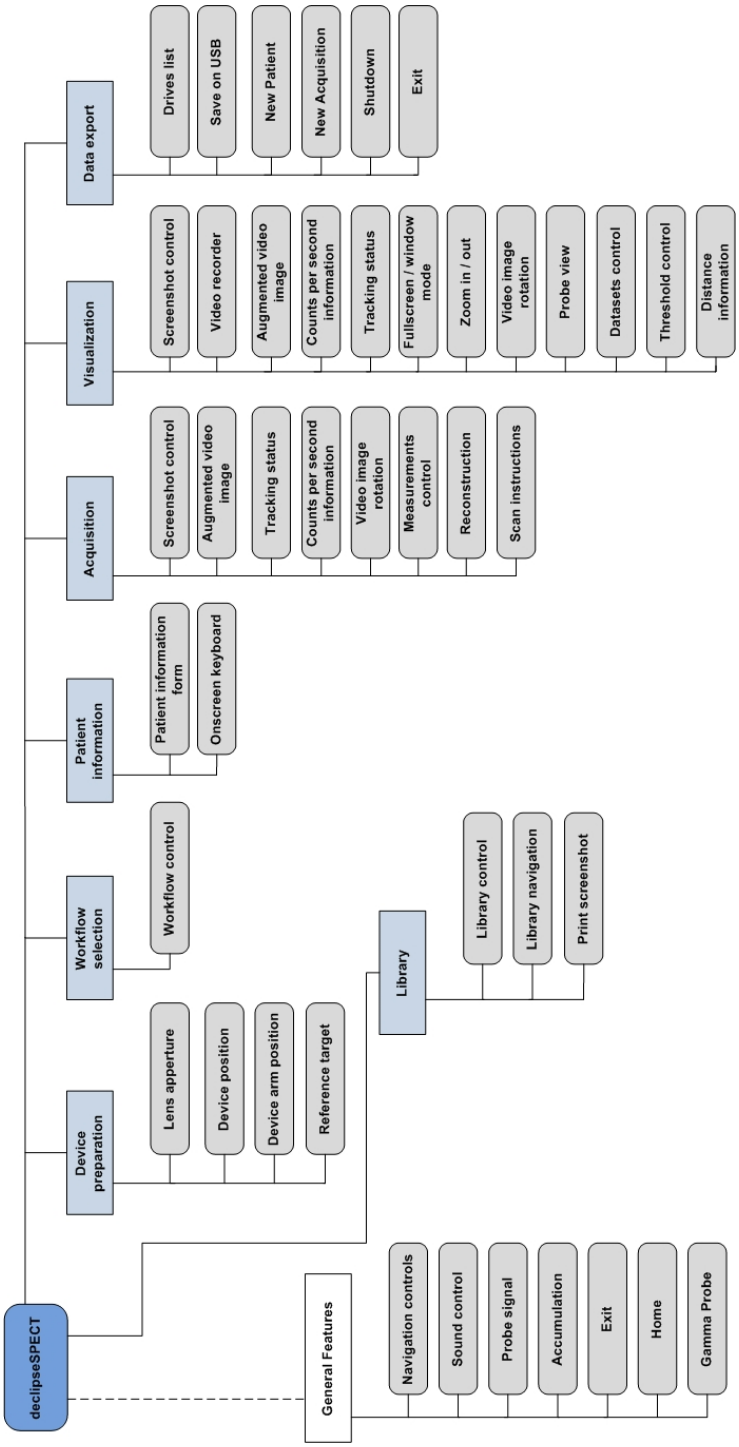


Figure 12.2.: Feature tree diagram of the SPECT device

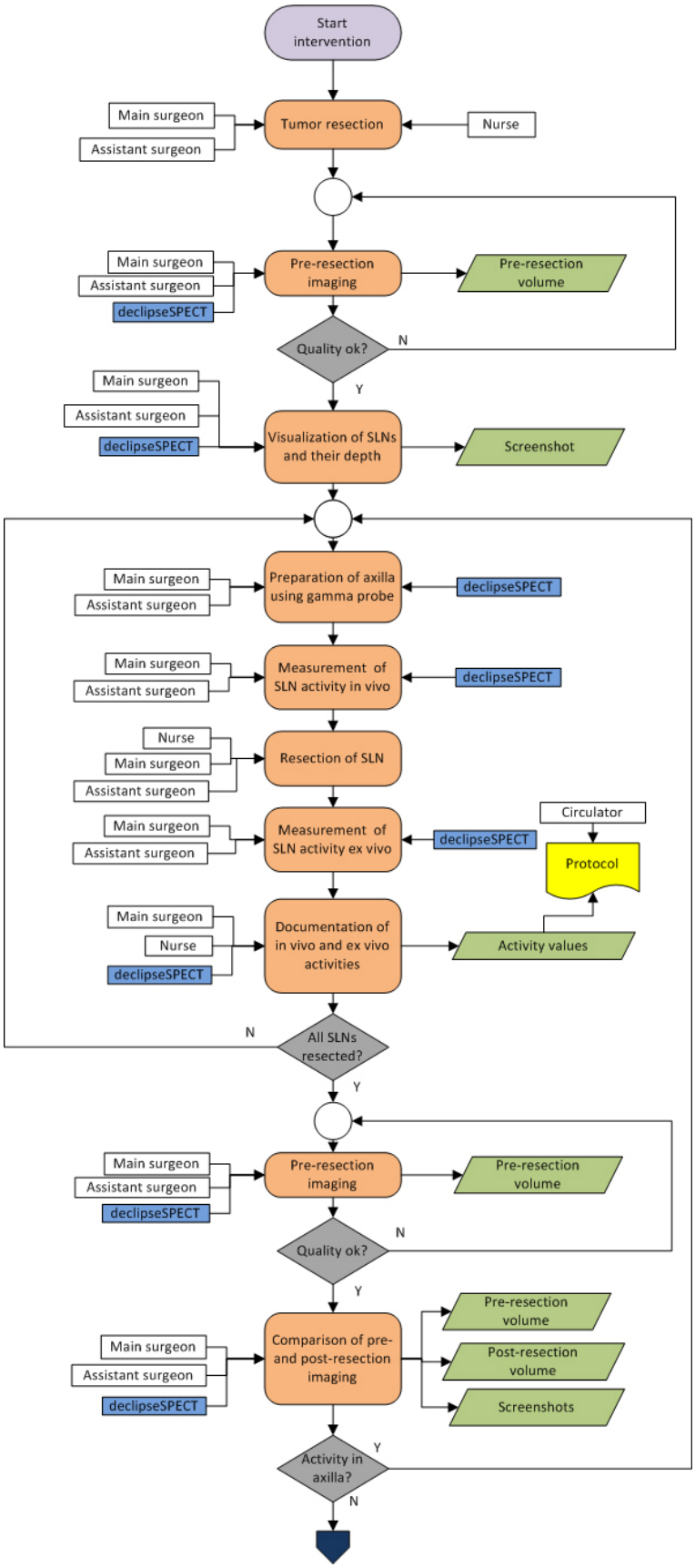


Figure 12.3.: SLNB Workflow using the SPECT device - Intervention part 1

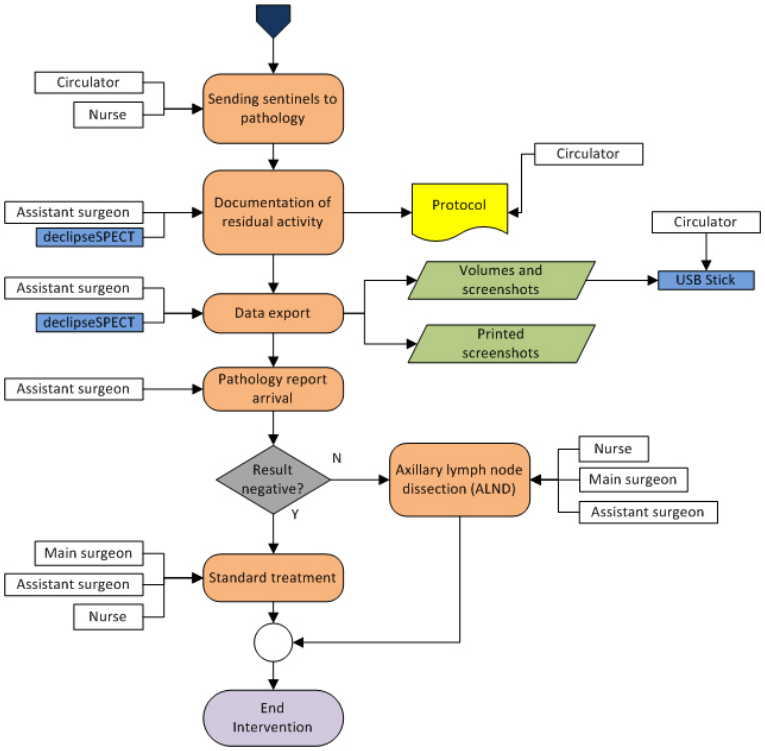
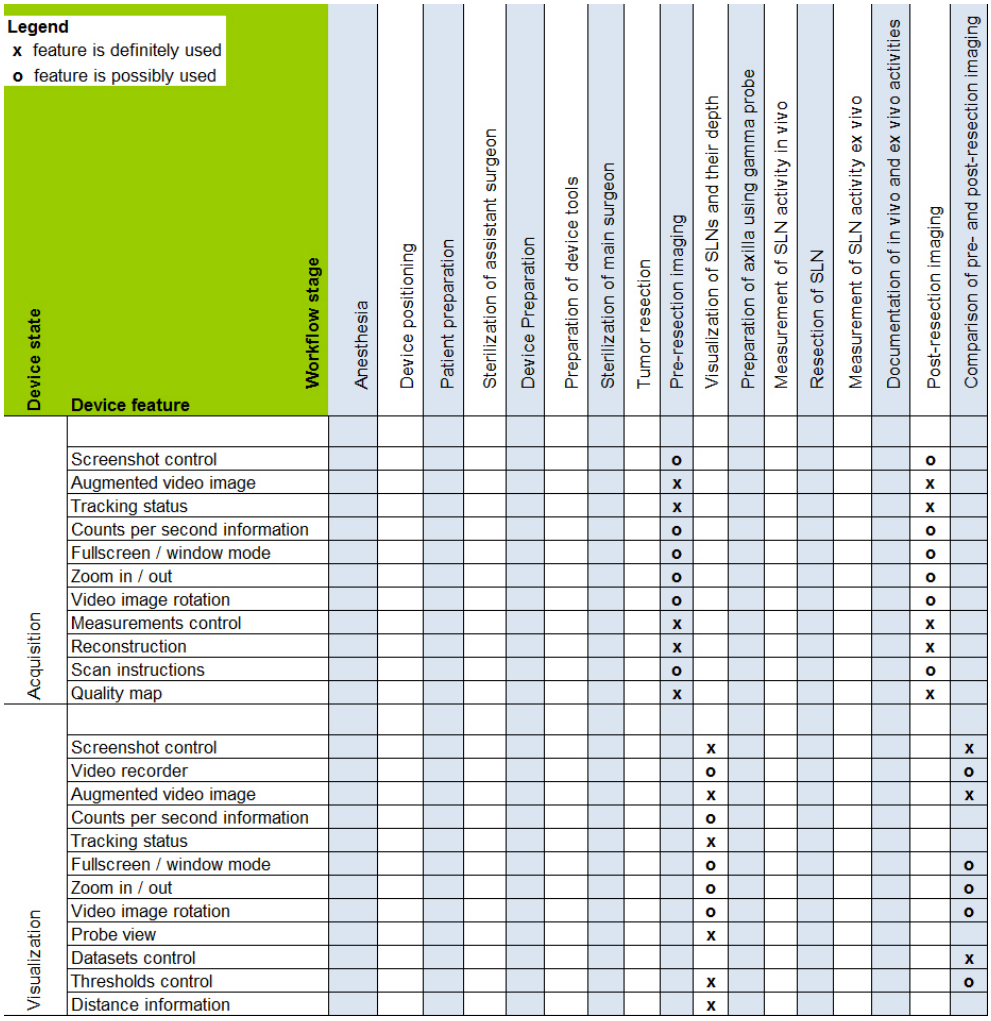


Figure 12.4.: SLNB Workflow using the SPECT device - Intervention part 2



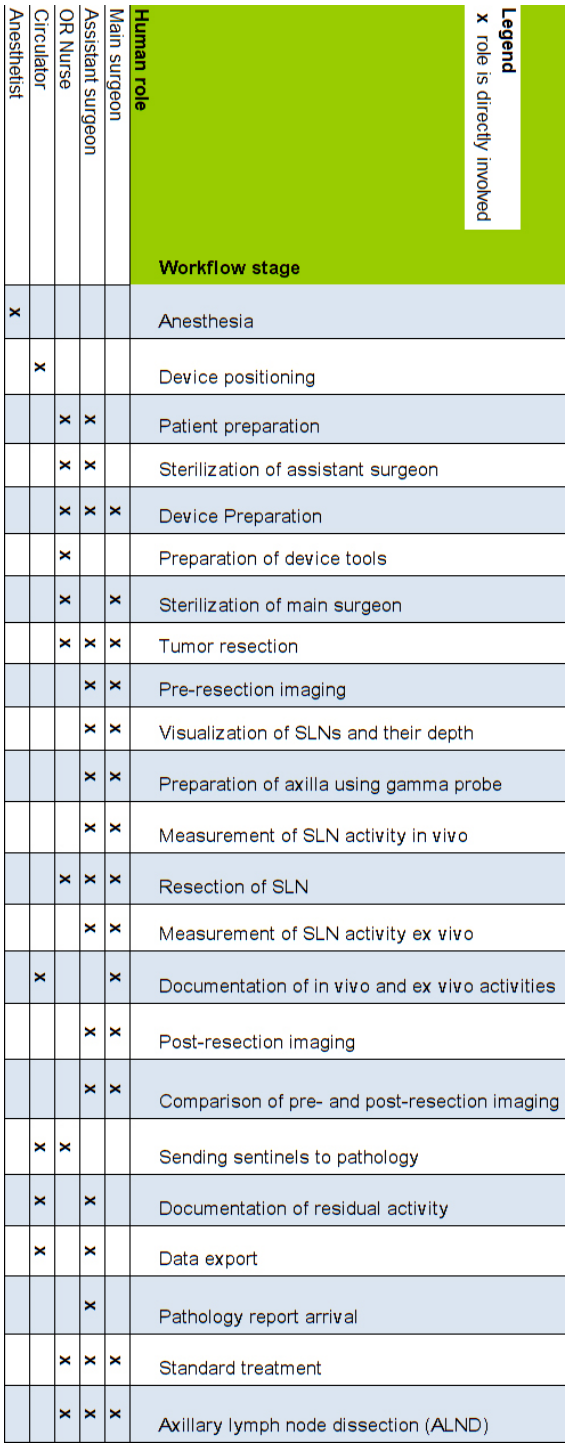
Legend
x feature is definitely used
o feature is possibly used

Device state	Device feature / Workflow stage	Anesthesia	Device positioning	Patient preparation	Sterilization of assistant surgeon	Device Preparation	Preparation of device tools	Sterilization of main surgeon	Tumor resection	Pre-resection imaging	Visualization of SLNs and their depth	Preparation of axilla using gamma probe	Measurement of SLN activity in vivo	Resection of SLN	Measurement of SLN activity ex vivo	Documentation of in vivo and ex vivo activities	Post-resection imaging	Comparison of pre- and post-resection imaging
Acquisition																		
	Screenshot control									o							o	
	Augmented video image									x							x	
	Tracking status									x							x	
	Counts per second information									o							o	
	Fullscreen / window mode									o							o	
	Zoom in / out									o							o	
	Video image rotation									o							o	
	Measurements control									x							x	
	Reconstruction									x							x	
	Scan instructions									o							o	
	Quality map									x							x	
Visualization																		
	Screenshot control										x							x
	Video recorder										o							o
	Augmented video image										x							x
	Counts per second information										o							
	Tracking status										x							
	Fullscreen / window mode										o							o
	Zoom in / out										o							o
	Video image rotation										o							o
	Probe view										x							
	Datasets control																	x
	Thresholds control										x							o
	Distance information										x							

Figure 12.5.: Excerpt of the mapping table between the device view and the workflow view, created during the SPECT device evaluation.

Legend x feature is intended to be used o feature is possibly used		Human role	Main surgeon	Assistant surgeon	OR Nurse	Circulator	Anesthetist
Device state	Device feature						
Acquisition							
	Screenshot control			x			
	Augmented video image		x	x			
	Tracking status		x	o			
	Counts per second information		x	o			
	Fullscreen / window mode			x			
	Zoom in / out			x			
	Video image rotation			x			
	Measurements control			x			
	Reconstruction			x			
	Scan instructions		x				
	Quality map		x	x			
Visualization							
	Screenshot control			x			
	Video Recorder			x			
	Augmented video image		x	x			
	Counts per second information		x	o			
	Tracking status		x	o			
	Fullscreen / window mode			x			
	Zoom in / out			x			
	Video image rotation			x			
	Probe view		x	x	o		
	Datasets control			x			
	Thresholds control			x			
	Distance information		x	o			

Figure 12.6.: Excerpt of the mapping table between the device view and the human roles view, created during the SPECT device evaluation.



Anesthetist	Circulator	OR Nurse	Assistant surgeon	Main surgeon	Human role / Workflow stage
x					Anesthesia
	x				Device positioning
		x	x		Patient preparation
		x	x		Sterilization of assistant surgeon
		x	x	x	Device Preparation
		x			Preparation of device tools
		x		x	Sterilization of main surgeon
		x	x	x	Tumor resection
			x	x	Pre-resection imaging
			x	x	Visualization of SLNs and their depth
			x	x	Preparation of axilla using gamma probe
			x	x	Measurement of SLN activity in vivo
		x	x	x	Resection of SLN
			x	x	Measurement of SLN activity ex vivo
	x			x	Documentation of in vivo and ex vivo activities
			x	x	Post-resection imaging
			x	x	Comparison of pre- and post-resection imaging
	x	x			Sending sentinels to pathology
	x		x		Documentation of residual activity
	x		x		Data export
			x		Pathology report arrival
		x	x	x	Standard treatment
		x	x	x	Axillary lymph node dissection (ALND)

Legend
x role is directly involved

Figure 12.7.: Excerpt of the mapping table between the human roles view and the workflow view, created during the SPECT device evaluation.

Bibliography

[1] Sentinel lymph node biopsy. Available online at http://www.cancer.gov/cancertopics/factsheet/therapy/sentinel-node-biopsy; visited on October 12th 2010.

[2] ISO 9241-11. *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability*, 1998.

[3] O. Amft and G. Tröster. Recognition of dietary activity events using on-body sensors. *Artificial Intelligence in Medicine*, 42(2):121–136, 2008.

[4] C. Ardito, F. Costabile, M. De Marsico, R. Lanzilotti, S. Levialdi, T. Roselli, and V. Rossano. An approach to usability evaluation of e-learning applications. *Univers. Access Inf. Soc.*, 4:270–283, 2006.

[5] D. E. Attarian. What is a preventable adverse event. Available online at http://www.aaos.org/news/aaosnow/may08/managing6.asp; visited on September 8th 2010.

[6] AttrakDiff. http://www.attrakdiff.de/en/Home/.

[7] Fiora T. W. Au, Simon Baker, Ian Warren, and Gillian Dobbie. Automated usability testing framework. In *Ninth Australasian User Interface Conference*, volume 76, pages 55–64, Wollongong, NSW, Australia, 2008. ACS.

[8] C. Backhaus. *Usability-Engineering in der Medizintechnik: Grundlagen - Methoden - Beispiele*. Springer-Verlag, 2010.

[9] Albert N. Badre, Scott E. Hudson, and Paulo J. Santos. Sychronizing video and event logs for usability studies. In *Proceedings of the workshop on Advanced visual interfaces*, AVI '94, pages 222–224, New York, NY, USA, 1994. ACM.

[10] R. Bailey and K. Bailey. Expediting the usability testing process. In *13th Annual Conference of the Usability Professionals Association*, 2003.

[11] Rafael Ballagas, Meredith Ringel, Maureen Stone, and Jan Borchers. istuff: A physical user interface toolkit for ubiquitous computing environments. pages 537–544. Press, 2003.

[12] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373 – 1396, Feb 2003.

[13] Ali Bigdelou, Tobias Sterner, Stefan Wiesner, Thomas Wendler, Florian Matthes, and Nassir Navab. OR Specific Domain Model for Usability Evaluations of Intra-operative Systems. In Russell Taylor and Guang-Zhong Yang, editors, *IPCAI 2011*, volume 6689 of *LNCS*, pages 25–35. Springer, Heidelberg, 2011.

[14] Andreas Bischof, Holger Stark, Reinhard Blumenstein, Thomas Wagner, Andre Brechmann, and Henning Scheich. A semiautomatic method for adding behavioral data to videotape records in combination with the noldus observer system. *Behavior Research Methods*, 33:549–555, 2001. 10.3758/BF03195415.

[15] S. Bleyer. Medizinisch-technische Zwischenfälle in Krankenhäusern und ihre Verhinderung. *Mitteilungen des Instituts für Biomedizinische Technik und Krankenhaustechnik der Medizinischen Hochschule Hannover. Hannover: Fachverlag für Krankenhaustechnik*, 1992.

[16] T. Blum, N. Padoy, H. Feussner, and N. Navab. Workflow mining for visualization and analysis of surgeries. pages 134–135, Barcelona, Spain, June 2008.

[17] Cristian Bogdan, Hermann Kaindl, Jürgen Falb, and Roman Popp. Modeling of interaction design by end users through discourse modeling. In *Proceedings of the 13th international conference on Intelligent user interfaces*, IUI '08, pages 305–308, New York, NY, USA, 2008. ACM.

[18] M.S. Bogner. *Human error in medicine*. L. Erlbaum Associates, 1994.

[19] L. Bouarfa, L. Stassen, P. Jonker, and J. Dankelman. Discovery of surgical high-level activities in the operating. In *ASCI 2010*, 2010.

[20] Jullien Bouchet. Icare software components for rapidly developing multimodal interfaces. pages 251–258. ACM Press, 2004.

[21] M. N. Boulos, B. J. Blanchard, C. Walker, J. Montero, A. Tripathy, and R. Gutierrez-Osuna. Web gis in practice x: a microsoft kinect natural user interface for google earth navigation. *Int. Journal of Health*.

[22] Marie-Luce Bourguet. A toolkit for creating and testing multimodal interface designs. In *Proceedings of UIST'02*, 2002.

[23] Andreas K. Buck, Alexandra Ehlerding, Jakob Saeckl, Ken Herrmann, Thomas Wendler, Marc E. Martignoni, Andreas Schnelzer, and Markus Schwaiger. Evaluation of feasibility of freehand spect in the or. In *Proceedings of the EANM 2010*, Vienna, Austria, Oct. 2010. EANM.

[24] M.L. Bustamante. *Learning WCF*. O'Reilly Media, 2007.

[25] K. A. Butler. Usability engineering turns 10. *interactions*, 3(1):58–75, 1996.

[26] M. C. Cabral, C. H. Morimoto, and M. K. Zuffo. On the usability of gesture interfaces in virtual reality environments. *Latin American conference on Human-computer interaction*, 2005.

[27] S. K. Card, T. P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, 1986.

[28] M. A. Carreira-Perpinán and Z. Lu. The laplacian eigenmaps latent variable model. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

[29] M. Chavez and A. Kyaw. A gesture-based interface and active cinema. In Sidney D Mello, Arthur Graesser, Björn Schuller, and Jean-Claude Martin, editors, *Affective Computing and Intelligent Interaction*, volume 6975 of *Lecture Notes in Computer Science*, pages 309–310. Springer Berlin, 2011.

[30] Min Chen, David Ebert, Hans Hagen, Robert S. Laramee, Robert van Liere, Kwan-Liu Ma, William Ribarsky, Gerik Scheuermann, and Deborah Silver. Data, information, and knowledge in visualization. *IEEE Comput. Graph. Appl.*, 29(1):12–19, January 2009.

[31] U. H. Chi. Formal specification of user interfaces: a comparison and evaluation of four axiomatic approaches. *IEEE Transactions on Software Engineering*, pages 671–685, 1985.

[32] P. K. Chilana, J. O. Wobbrock, and A. J. Ko. Understanding usability practices in complex domains. In *CHI '10: Proceedings of the 28th international conference on Human factors in computing systems*, pages 2337–2346. ACM, 2010.

[33] E. F. Codd. *The relational model for database management: version 2*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[34] S. Demirci, F. Manstad-Hulaas, and N. Navab. Quantification of aortic deformation after EVAR. *SPIE Medical Imaging*, 2009.

[35] Pierre Dragicevic, Jean-Daniel Fekete, Rue Alfred Kastler, and La Chantrerie. Input device selection and interaction configuration with icon, 2001.

[36] Bruno Dumas, Denis Lalanne, and Rolf Ingold. Hephaistk: a toolkit for rapid prototyping of multimodal interfaces. In *Proceedings of the 2009 International Conference on Multimodal Interfaces*, pages 231–232, New York, USA, 2009. ACM.

[37] A. Elgammal and C. Lee. The role of manifold learning in human motion analysis. *Human Motion Understanding, Modeling, Capture and Animation*, 2008.

[38] A Scientific Computing Problem Solving Environment. `http://www.scirun.org/`.

[39] E. Evans. *Domain-Driven Design - Tackling Complexity in the Heart of Software*. Addison-Wesley, 2004.

[40] Jean-Daniel Fekete, Jarke J. Wijk, John T. Stasko, and Chris North. Information visualization. chapter The Value of Information Visualization, pages 1–18. Springer-Verlag, Berlin, Heidelberg, 2008.

[41] A. Ferscha, S. Resmerita, C. Holzmann, and M. Reichör. Orientation sensing for gesture-based interaction with smart artifacts. *Computer communications*, 28(13):1552–1563, 2005.

[42] Markus Fluckiger and Michael Richter. *Usability Engineering kompakt. Benutzbare Software gezielt entwickeln*. Spektrum, 2 edition, 2010.

[43] G. Forestier, L. Riffaud, B. Trelhu, and P. Jannin. Classification of surgical processes using dynamic time warping. In *Journal of Biomedical Informatics*, pages 255–264, 2012.

[44] A. Freudenthal, T. Studeli, P. Lamata, and E. Samset. Collaborative co-design of emerging multi-technologies for surgery. *Journal of Biomedical Informatics*, 44:198–215, 2011.

[45] L.M. Friedman, C.D. Furberg, and D.L. Demets. *Fundamentals of Clinical Trials*. Springer, 2010.

[46] Kevin J. Fish und Steven K. Rall Marcus Gaba, David M. *Zwischenfälle in der Anästhesie: Pravention und Management*. 1998.

[47] L Gallo. A glove-based interface for 3d medical image visualization. *Intelligent interact. multimedia sys. and services*, 6:221–230, 2010.

[48] L. N. Gallo, A. P. Placitelli, and M. Ciampi. Controller-free exploration of medical image data: Experiencing the kinect. In *International Symposium on Computer-Based Medical Systems*, 2011.

[49] T. Gonzalez-Sanchez and D. Puig. Real-time body gesture recognition using depth camera. *Electronics Letters*, 47:697–698, 2011.

[50] C. Graetzel, T. Fong, S. Grange, and C. Baur. A non-contact mouse for surgeon-computer interaction. *Technology and Health Care*, 12(3):245–257, 2004.

[51] M. Gray, A. Badre, and M. Guzdial. Visualizing usability log data. In *Proceedings of the 1996 IEEE Symposium on Information Visualization (INFOVIS '96)*, INFOVIS '96, Washington, DC, USA, 1996. IEEE Computer Society.

[52] P. Gray, A. Ramsay, and M. Serrano. A demonstration of the openinterface interaction development environment. In *Symposium on User Interface Software and Technology (UIST)*, UIST '07, 2009.

[53] François Guimbretiére, Morgan Dixon, and Ken Hinckley. Experiscope: an analysis tool for interaction data. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 1333–1342, New York, NY, USA, 2007. ACM.

[54] François Guimbretiére and Terry Winograd. Flowmenu: combining command, text, and data entry. In *Proc. of the 13th annual ACM symposium on UIST*, UIST, pages 213–216, New York, NY, USA, 2000. ACM.

[55] D. Gulbransen. *Using XML*. Special Edition Series. Que, 2002.

[56] S. Gustafson, D. Bierwirth, and P. Baudisch. Imaginary interfaces: spatial interaction with empty hands and without visual feedback. *Symposium on User Interface Software and Technology*, 2010.

[57] Beverly Harrison, Russel Owen, and Ronald Baecker. Timelines an interactive system for the collection and visualization of temporal data, 1994.

[58] B. Hartmann and N. Link. Gesture recognition with inertial sensors and optimized dtw prototypes. *IEEE International Conference on Systems Man and Cybernetics (SMC)*, pages 2102–2109, 2010.

[59] Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 145–154, New York, NY, USA, 2007. ACM.

[60] A.M.J. Hass. *Configuration management principles and practice*. Addison-Wesley, 2003.

[61] M. Hassenzahl, M. Burmester, and F. Koller. Attrakdiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität. In G. Szwillus and J. Ziegler, editors, *Mensch und Computer 2003: Interaktion in Bewegung*, pages 187–196. B. G. Teubner, 2003.

[62] M. Hassenzahl, R. Kekez, and M. Burmester. The importance of a software's pragmatic quality depends on usage modes. Work with display units - world wide work : proceedings of the 6th interntional Scientific Conference on Work with Display units, WWDU 2002 - <6, 2002, Berchtesgaden. Hrsg.: H. Luczak ... . - Berlin, Ergonomic, inst. f. Arbeits- und Sozialforschung, 2002, January 2002.

[63] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: Data Mining, Inference and Prediction*. Springer, 2001.

[64] P. Hodgson. Usability for medical devices: A new international standard. Available online at `http://www.blueprintusability.com/topics/articleiec62366.html`; visited on October 4th 2010.

[65] R. Holm, M. Priglinger, E. Stauder, J. Volkert, and R. Wagner. Automatic data acquisition and visualization for usability evaluation of virtual reality systems. In *EUROGRAPHICS 2002*. The Eurographics Association, 2002.

[66] M. Hopmann, P. Salamin, N. Chauvin, F. Vexo, and D. Thalmann. Natural activation for gesture recognition systems. In *extended abstracts on Human factors in computing systems*. ACM, 2011.

[67] IIBA. *A Guide to the Business Analysis Body of Knowledge (BABOK Guide)*. International Institute of Business Analysis, 2009.

[68] Mangold INTERACT. `http://www.mangold-international.com/software/interact`.

[69] D. Ionescu, B. Ionescu, C. Gadea, and S. Islam. Controller-free exploration of medical image data: Experiencing the kinect. In *Int. Conf. on Computer Communications and Networks (ICCCN)*, 2011.

[70] M. Isard and A. Blake. Condensation—conditional density propagation for visual tracking. *International Journal of Computer Vision*, Jan 1998.

[71] Michael Isard and Andrew Blake. A mixed-state condensation tracker with automatic model-switching. *IEEE International Conference on Computer Vision (ICCV)*, pages 107–112, Jan 1998.

[72] ISO IEC 62366. *Medical devices – Application of usability engineering to medical devices*. 2007.

[73] M. Y. Ivory and M. A. Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM COMPUTING SURVEYS*, 33:470–516, 2001.

[74] T. Jaeggli, E. Koller-Meier, and L. Van Gool. Learning generative models for multi-activity body pose estimation. *Int J Comput Vis*, 83(2):121–134, 2009.

[75] A. Jaimes and N. Sebe. Multimodal human-computer interaction: A survey. *Computer Vision and Image Understanding*, 2007.

[76] P. Jannin and W. Korb. Assessment in image guided interventions. In *MICCAI*. Springer-Verlag, 2008.

[77] P. Jannin, M. Raimbault, X. Morandi, and B. Gibaud. Modeling surgical procedures for multimodal image-guided neurosurgery. In *MICCAI*, pages 565–572. Lecture Notes in Computer Science, 2001.

[78] P. Jannin, M. Raimbault, X. Morandi, L. Riffaud, and B. Gibaud. Modeling surgical procedures for multimodal image-guided neurosurgery. In *Journal of Computer Aided Surgery*, pages 98–106, 2003.

[79] P. Jannin, M. Raimbault, X. Morandi, E. Seigneuret, and B. Gibaud. Design of a neurosurgical gestures model for multimodal image guided surgery. In *Computer Assisted Radiology and Surgery*, pages 102–107. Elsevier Science, 2001.

[80] B. E. John and D. E. Kieras. The goms family of user interface analysis techniques: comparison and contrast. *ACM Trans. Comput.-Hum. Interact.*, 3(4):320–351, 1996.

[81] R. Johnson, K. O'Hara, A. Sellen, C. Cousins, and A. Criminisi. Exploring the potential for touchless interaction in image-guided interventional radiology. *ACM Conference on Human Factors in Computing Systems*, 2011.

[82] D. Achacon Jr, D. Carlos, and M. Puyaoan.... Realism: Real-time hand gesture interface for surgeons and medical experts. *Citeseer*.

[83] J. Kela, P. Korpipää, J. Mäntyjärvi, S. Kallio, G. Savino, L. Jozzo, and S.D. Marca. Accelerometer-based gesture control for a design environment. *Pers Ubiquit Comput*, 10(5):285–299, 2006.

[84] T. Kipshagen, M. Graw, V. Tronnier, M. Bonsanto, and U.G. Hofmann. Touch-and marker-free interaction with medical software. *Medical Physics and Biomedical Engineering*, pages 75–78, 2009.

[85] J. Kjeldskov, M. Skov, and J. Stage. A longitudinal study of usability in health care: Does time heal. *Int. J. of Medical Informatics*, 79(6):e135–e143, 2010.

[86] Jesper Kjeldskov, Mikael B. Skov, and Jan Stage. Instant data analysis: conducting usability evaluations in a day. In *Proceedings of the third Nordic conference on Human-computer interaction*, NordiCHI '04, pages 233–240, New York, NY, USA, 2004. ACM.

[87] R. A. Kockro, A. Stadie, E. Schwandt, R. Reisch, C. Charalampaki, I. Ng, T. T. Yeo, P. Hwang, L. Serra, and A. Perneczky. A collaborative virtual reality environment for neurosurgical planning and training. *Neurosurgery*, 61:379–391, 2007.

[88] L. T. Kohn, J. Corrigan, and M. S. Donaldson. *To err is human: building a safer health system*. National Academy Press, 1999.

[89] J. Kotz, R. Haban, and S. Steckermeier. *Dot Net 3.0*. Addison-Wesley, 2007.

[90] A. W. Kushniruk and V. L. Patel. Cognitive and usability eng. methods for the evaluation of clinical info. system. *J. of Biomedical Informatics*, 37(1):56–76, 2004.

[91] Jean-Yves Lionel Lawson, Ahmad-Amr Al-Akkad, Jean Vanderdonckt, and Benoit Macq. An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, EICS '09, pages 245–254, New York, NY, USA, 2009. ACM.

[92] E. Liljegren. Usability in a medical technology context assessment of methods for usability evaluation of medical equipment. *International Journal of Industrial Ergonomics*, 36(4):345–352, 2006.

[93] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[94] Ivo Maly and Pavel Slavík. Towards visual analysis of usability test logs using task models. In *Proceedings of the 5th international conference on Task models and diagrams for users interface design*, TAMODIA'06, pages 24–38, Berlin, Heidelberg, 2007. Springer-Verlag.

[95] M.J. McAuliffe, F.M. Lalonde, D McGarry, K Csaky, and B.L. Trus. Medical image processing, analysis and visualization in clinical research. In *Proceedings. 14th IEEE Symposium on Computer-Based Medical Systems*, pages 381–386. IEEE, 2001.

[96] MeVisLab. `http://www.mevislab.de/`.

[97] Microsoft. System-provided bindings. Available online at `http://msdn.microsoft.com/en-us/library/ms730879.aspx`; visited on September 9th 2011.

[98] Microsoft Corporation. XBox Kinect. `www.xbox.com/kinect`.

[99] Microsoft Live Labs Pivot. *Pivot Viewer*. Available online at http://www.microsoft.com/silverlight/pivotviewer/, 2010.

[100] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(3):311–324, 2007.

[101] Morae. `http://www.techsmith.com/morae.html`.

[102] T. P. Moran. The command language grammar: a representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies*, 15(1):3–50, 1981.

[103] H. Mosemann and M. Kose. *Android*. Hanser, 2009.

[104] T. Neumuth, N. Durstewitz, M. Fischer, G. Strauss, A. Dietz, J. Meixensberger, P. Jannin, K. Cleary, H. U. Lemke, and O. Burgert. Structured recording of intraoperative surgical workflows. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 6145, pages 54–65, 2006.

[105] T. Neumuth, P. Jannin, G. Strauss, Meixensberger, and O. Burgert. Validation of knowledge acquisition for surgical process models. In *Journal of the American Medical Informatics Association*, pages 72–80, 2009.

[106] T. Neumuth, G. Strauß, J. Meixensberger, H. Lemke, and O. Burgert. Acquisition of process descriptions from surgical interventions. In *Database and Expert Systems Applications*, pages 602–611, 2006.

[107] N. Nguyen-Duc-Thanh, D. Stonier, S. Lee, and D. Kim. A new approach for human-robot interaction using human body language. In *Proceedings of the 5th international conference on Convergence and hybrid information technology*, ICHIT'11, pages 762–769, Berlin, Heidelberg, 2011. Springer-Verlag.

[108] J. Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers, 1994.

[109] K. Ohnuma, K. Masamune, K. Yoshimitsu, T. Sadahiro, J. Vain, Y. Fukui, and F. Miyawaki. Timed-automata-based model for laparoscopic surgery and intraoperative motion recognition of a surgeon as the interface connecting the signal scenario. *Int J Com Assist Radiol Surg*, 1(1):442–445, 2006.

[110] Asli Okur, Seyed-Ahmad Ahmadi, Ali Bigdelou, Thomas Wendler, and Nassir Navab. MR in OR: First analysis of AR/VR visualization in 100 intra-operative Freehand SPECT acquisitions. In *Proceedings of the 10th International Symposium*

on Mixed and Augmented Reality (ISMAR), pages 211–218, Basel, Switzerland, October 2011.

[111] S. Pieper, M. Halle, and R. Kinkinis. 3d slicer. In *Proceedings of the 1st IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 632–635. IEEE, 2004.

[112] Stuart Pook, Eric Lecolinet Guy Vaysseix, and Emmanuel Barillot. Control menus: Execution and control in a single interactor. In *ACM CHI Conf. on Human Factors in Computing Systems*, pages 263–264. ACM Press, 2000.

[113] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey. *Human-Computer Interaction: Concepts And Design (ICS)*. Addison Wesley, 1994.

[114] Pardha S. Pyla, Jonathan R. Howarth, Chris Catanzaro, and Chris North. Vizability: a tool for usability engineering process improvement through the visualization of usability problem data. In *Proceedings of the 44th annual Southeast regional conference*, ACM-SE 44, pages 620–625, New York, NY, USA, 2006. ACM.

[115] J. Redish. Expanding usability testing to evaluate complex systems, 2006.

[116] W. A. Rehmann and S. A. Wagner. *Medizinproduktegesetz (MPG): Mit Erläuterungen*. Beck, 2005.

[117] L. Riffaud, T. Neumuth, X. Morandi, C. Trantakis, J. Meixensberger, O. Burgert, B. Trelhu, and P. Jannin. Recording of surgical processes: a study comparing senior and junior neurosurgeons during lumbar disc herniation surgery. In *Neurosurgery*, pages 325–332, 2010.

[118] J. Rosen, J.D. Brown, L. Chang, M.N. Sinanan, and B. Hannaford. Generalized approach for modeling minimally invasive surgery as a stochastic process using a discrete markov model. *IEEE Trans. on Biomedical Eng.*, 53(3):399–413, 2006.

[119] J. Rubin. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[120] Ivan Santi, Lorenzo Fantini, Ken Herrmann, Chiara Fuccio, Thomas Wendler, Paola Caroli, Paolo Castelucci, Andreas K. Buck, Markus Schwaiger, and Stefano Fanti. Freehand spect for sentinel lymph node biopsy: clinical experience. In *EANM 2010*, Vienna, Austria, Oct. 2010. EANM.

[121] Martin J. Schedlbauer. An extensible platform for the interactive exploration of fitts' law and related movement time models. In *CHI '07 extended abstracts on Human factors in computing systems*, CHI EA '07, pages 2633–2638, New York, NY, USA, 2007. ACM.

[122] M. Schubert. *FMEA - Fehlermöglichkeits- und Einflušanalyse: Leitfaden*. Beuth, 1993.

[123] L Schwarz, A Bigdelou, and N Navab. Learning gestures for customizable human-computer interaction in the operating room. In *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2011.

[124] L. A. Schwarz, D. Mateus, and N. Navab. Multiple-activity human body tracking in unconstrained environments. *AMDO*, pages 192–202, Mar 2010.

[125] A. Seffah, M. Donyaee, R. B. Kline, and H. Padda. Usability measurement and metrics: A consolidated model. *Software Quality Journal*, 14(2):159–178, June 2006.

[126] Marcos Serrano, David Juras, and Laurence Nigay. A three-dimensional characterization space of software components for rapidly developing multimodal interfaces. In *Proceedings of the 10th international conference on Multimodal interfaces*, ICMI '08, pages 149–156, New York, NY, USA, 2008. ACM.

[127] Anoop K. Sinha and James A. Landay. Capturing user tests in a multimodal, multi-device informal prototyping tool. In *Proceedings of the 5th international conference on Multimodal interfaces*, ICMI '03, pages 117–124, New York, NY, USA, 2003. ACM.

[128] Richard Socher and Matthias Hein. *Manifold Learning and Dimensionality Reduction with Diffusion Maps*. Tech. report, 2008.

[129] S. Soutschek, J. Penne, J. Hornegger, and J. Kornhuber. 3-d gesture-based scene navigation in medical imaging applications using time-of-flight cameras. *Computer Vision and Pattern Recognition Workshops*, Apr 2008.

[130] Gero Strauss, Kirill Koulechov, Stefan Rottger, Jenny Bahner, Christos Trantakis, Mathias Hofer, Werner Korb, Oliver Burgert, Juergen Meixensberger, Dietrich Manzey, Andreas Dietz, and Tim Luth. Evaluation of a navigation system for ENT with surgical efficiency criteria. *The Laryngoscope*, 116(4):564–572, 2006.

[131] Watchfire: Accessibility Testing. `http://www.watchfire.com/products/webxm/bobb`.

[132] M. Urban, P. Bajcsy, R. Kooper, and J.C. Lementec. Recognition of arm gestures using multiple orientation sensors: Repeatability assessment. *Intelligent Transportation Systems*, pages 553–558, 2004.

[133] K. J. Vicente. *Cognitive Work Analysis : Toward Safe, Productive, and Healthy Computer-Based Work*. CRC Press, 1999.

[134] Kitware VolView. `http://www.kitware.com/products/volview.html`.

[135] Ulrike von Luxburg. A tutorial on spectral clustering. *Stat Comput, Springer Science and Business Media*, 17(5):395–416, 2007.

[136] W3C. Extensible markup language (xml) 1.0 (fifth edition). Available online at `http://www.w3.org/TR/xml/`; visited on September 4th 2011.

[137] W3C. Web services architecture requirements. Available online at `http://www.w3.org/TR/2002/WD-wsa-reqs-20020429#N100CB`; visited on September 8th 2011.

[138] W3C. Xml essentials. Available online at `http://www.w3.org/standards/xml/core`; visited on September 4th 2011.

[139] J. P. Wachs, H. Stern, Y. Edan, M. Gillam, C. Feied, M. Smith, and J. Handler. A real-time hand gesture interface for medical visualization applications. *Applications of Soft Computing*, pages 153–162, 2006.

[140] G. Welch and G. Bishop. An introduction to the kalman filter, 1995.

[141] T. Vidal Wendler. 3d intraoperative functional imaging with navigated probes, 2010.

[142] Janet Wesson and Darelle Van Greunen. Visualisation of usability data: measuring task efficiency. In *Proceedings of the 2002 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*, SAICSIT '02, pages 11–18, Republic of South Africa, 2002. South African Institute for Computer Scientists and Information Technologists.

[143] M. Wiklund and S. Wilcox. *Designing Usability Into Medical Products*. CRC, 2004.

[144] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. *Symposium on User Interface Software and Technology*, 2007.

[145] Ivo Wolf, Marcus Vetter, Ingmar Wegner, Thomas Bottger, Marco Nolden, Max Schobinger, Mark Hastenteufel, Tobias Kunert, and Hans-Peter Meinzer. The medical imaging interaction toolkit. *Medical Image Analysis*, 9(6):594 – 604, 2005.

[146] Kebomix Wordpress. Android system architecture. Available online at `http://kebomix.wordpress.com/2010/08/17/android-system-architecture/`; visited on August 29th 2011.

[147] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 379–385, 1992.

[148] Lei Yan, Xiaodong Zhou, Lu Lu, Arthur G. Centeno, Leonard Kuan, Michael Hawrylycz, Monica Larimer, Glenn D. Rosen, and Robert W. Williams. Global exploratory analysis of massive neuroimaging collections using microsoft silverlight pivotviewer. In *Biomedical Science and Engineering Conference Annual ORNL*, 2011.

[149] B. Yoo, J. J. Han, C. Choi, H. Ryu, D. S. Park, and K. C. Yeong. 3d remote interface for smart displays. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, pages 551–560, New York, NY, USA, 2011. ACM.

[150] Xu Zhang, Xiang Chen, Wen-hui Wang, Ji-hai Yang, Vuokko Lantz, and Kong-qiao Wang. Hand gesture recognition and virtual game control based on 3d accelerometer and emg sensors. In *Proceedings of the 14th international conference on Intelligent user interfaces*, IUI '09, pages 401–406, New York, NY, USA, 2009. ACM.