

Daniel Lachmann

Streamlining the Process of creating and viewing
Illustrated Technical Manuals: Development of an
ergonomic Java Application

Bachelorthesis

Daniel Lachmann

Streamlining the Process of creating and viewing
Illustrated Technical Manuals: Development of an
ergonomic Java Application

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Mechatronik
an der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.Ing. Thomas Frischgesell
Zweitgutachter : Prof. Dr.Ing. Randolph Isenberg

Abgegeben am 2. September 2016

Daniel Lachmann

Thema der Bachelorthesis

Entwicklung einer Java Anwendung zur Vereinfachung des Erstellens und der Verwendung von illustrierten technischen Anleitungen

Stichworte

Technische Dokumentation, Montageanleitung, Inbetriebnahme, Wartung

Kurzzusammenfassung

Thema

Das Ziel der Arbeit ist es, das Erstellen strukturierter Dokumentation zu technischen Vorgängen, wie z.B. Montage, Inbetriebnahme oder Wartungsanleitungen zu erleichtern. Es wird eine Betriebssystem-übergreifend verwendbare Software entwickelt, welche mit Hilfe einer Webcam zunächst serielle technische Abläufe in Bild, Text und Reihenfolge dokumentieren kann. Das Format entspricht einer digitalen Bildergalerie, in welcher zu jedem Bild zusätzliche Informationen wie z.B. ein beschreibender Text hinzugefügt werden kann.

Umfang der Arbeit

Der Stand der Technik bzgl. der Erstellung von Anleitungen soll auf Schwachstellen untersucht werden. Aufbauend auf den der Entwickleridee zugrundeliegenden Basisfunktionen (Aufnahmen einer Bilderserie mit Text zu jedem Bild) werden unter Berücksichtigung der vorangegangenen Analyse sinnvolle Features erarbeitet, um das Produkt der Arbeit über den Stand der Technik zu heben. Die erarbeiteten Funktionen werden bzgl. des voraussichtlichen Umsetzungsaufwands und ihres Anwendernutzens bewertet, und ein im Zeitrahmen realistisches Feature-Set für die Version 1.0 der Anwendung ausgewählt. Anschliessend werden die ausgewählten Features in die Software integriert, und das fertige Programm anhand praktischer Beispiele getestet. Die Anwendungsbereiche sollen sowohl mechanische als auch elektrotechnische Montage, sowie Wartung und Inbetriebnahme technischer Systeme umfassen. Insbesondere soll mindestens ein Anwendungsfall die Inbetriebnahme eines vom PC angesteuerten Gerätes beinhalten, wobei neben den physikalischen Anschlüssen auch die Verbindungsherstellung mit der steuernden Software dokumentiert wird. Die aus den Tests resultierenden Anleitungen werden ausgewertet und mit dem zuvor charakterisierten Stand der Technik bezüglich ihrer Stärken und Schwächen verglichen. Die zur Herstellung der Anwendung verwendeten Software-Tools und Ressourcen werden beschrieben, und der Prozess der Softwareentwicklung dokumentiert.

Einschränkungen

Bei der Entwicklung im Rahmen der Bachelorarbeit liegt der Fokus auf der Dokumentation von Gegenständen, welche auf einem oder im nahen Umfeld eines PC-Arbeitsplatzes untergebracht werden können. Weiterhin werden sollen Testgegenstände ausreichend gross sein, um ohne Vergrösserungsobjektive von einer Webcam erfasst werden zu können. Es ist vorgesehen, dass mit der entwickelten Anwendung vornehmlich papierlose Dokumentation erstellt wird. Trotzdem wird eine export-Funktion in ein gängiges Format (PDF oder HTML) bei der Feature-Auswahl berücksichtigt werden.

Daniel Lachmann

Title of the paper

Streamlining the Process of creating and viewing Illustrated Technical Manuals: Development of an ergonomic Java Application

Keywords

Technical Documentation, Assembly Manual, Commissioning, Maintenance

Abstract

Topic

The aim of this thesis is to make creating documentation for assembly, commissioning and maintenance of technical devices less time consuming and more ergonomical. To achieve this, an open-source, platform-independent software application will be developed. The basic functionality of the software will cover the capture of images via webcam and their arrangement in an image gallery format, with a text description attached to each image. The software will also have a read-only mode of operation to allow the viewing of documentation files.

Scope

The state of the art regarding the creation of relevant types of technical documentation will be analysed and its strengths and weaknesses identified. Using the results of this analysis, new features and changes to the initial concept shall be derived with the goal of matching the established processes' strengths, while eliminating as many weaknesses as possible. The features and changes identified will be evaluated regarding their usefulness for the documenter or reader vs. their predicted implementation time-cost. Based on the evaluation and existing time constraints, a feature set will be selected for integration into the base application, which will define the capabilities of the first published version of the software.

Following the implementation of the new feature set, the application will undergo practical testing. All three use-cases described above shall be integrated into the tests. At least one test shall document the procedure of commissioning a PC-controlled device, including the control-software setup.

The process of creating documentation for the testcases shall be documented and critically compared to the state of the art. The documentation created during the tests shall be analyzed from the documentation-users perspective, and compared to manuals created through traditional means. A rough outline for future development of the software shall be presented.

The tools and resources used in the creation of the software will be documented and the process of the software development summarized.

Constraints

During this thesis the focus shall be documentation of items which can fit onto or closely beside a PC-equipped worktop, and are sufficiently large to be captured by a webcam without the need for additional optics. The intention is to keep the documentation process paper-free wherever possible. Nevertheless, an export functionality (e.g. to .PDF or HTML) to enable printing of the documentation shall be considered during the feature-selection process.

Contents

List of Tables	9
List of Figures	10
1 Introduction	11
2 State of the Art	14
2.1 Selection of Sources	14
2.2 Literature Review	14
3 Analysis	20
3.1 Viewpoint: Creator of Illustrated Technical Documentation	20
3.1.1 Evaluation of the State of the Art	20
3.1.2 The Documentation Process to consider during the Design Phase	22
3.1.3 Conclusion	24
3.2 Viewpoint: User of Illustrated Technical Documentation	24
3.2.1 Evaluation of the State of the Art	24
3.2.2 Using the Documentation	26
3.2.3 Conclusion	26
4 Developing the Application Concept	28
4.1 The Initial Concept	28
4.1.1 Envisioned Use-Cases	28
4.1.2 Software	28
4.1.3 Hardware	30
4.2 Proposed Additional Features	32
4.2.1 Additional Features gained from Reviewed Literature	32
4.2.2 Additional Features conceived during the Prototype Design Phase	32
4.2.3 Evaluating the Features	35
4.2.4 Composing the Feature-Sets	36
4.3 Selection of a Feature-Set for Implementation	41
4.4 The Software-Engineering Process: some points of interest	41
4.4.1 Designing the GUI	41

4.4.2	Designing the Gallery	43
4.4.3	Capturing Images with the Camera	46
4.4.4	Designing the Stop Motion Pictures	47
4.4.5	Saving the Gallery	48
4.4.6	Making the Project- and Application Settings Available to all Classes	50
5	Software Development	51
5.1	Language and Tools	51
5.1.1	The Java Programming Language	51
5.1.2	GUI Framework	52
5.1.3	Setting up Eclipse JDT with e(fx)clipse Plugin	53
5.1.4	SceneBuilder 2.0	53
5.2	Software Libraries	54
5.2.1	Criteria	54
5.2.2	Webcam Capture API	54
5.2.3	Apache Commons Library	55
5.3	Summary of the implementation Process	56
5.3.1	Successfully implemented Features	56
5.3.2	Discarded Features	56
6	Testing and Evaluation	59
6.1	Objective & Outline	59
6.2	Summary of the Test	59
7	Conclusion	62
7.1	Personal Development	62
7.2	Plans for the Future	63
	Bibliography	65
	Appendix	67
	Glossary	135

List of Tables

5.1 Comparison of Webcam Capture APIs	55
---	----

List of Figures

4.1	The first detailed conceptual sketch of the system in use.	31
4.2	Selected features are providing a balance between user- and documenter-orientation.	37
4.3	Selected features focus on information collection and structuring.	38
4.4	Ideal case for implementation without time constraints.	39
4.5	Selected features focus on features not available in other free tools, such as word-processors and graphics utilities	40
4.6	The features of Feature-Set 4, grouped in the GUI structure given by the perspectives	42
4.7	A small example subset of the GUI specification	43
4.8	The inheritance structure of the gallery items.	45
4.9	The interaction between main application thread (represented by the gallery browser controller) and camera thread.	47
4.10	The hierarchy of gallery items during the saving procedure.	49
5.1	Overview of the implemented and the discarded features	57
7.1	The initial design of the draft mode's layout	82
7.2	The sub controller operating within the draft mode	83
7.3	The initial design of the refine mode's layout	84
7.4	The sub controllers operating within the refine mode	85
7.5	The initial design of the view mode's layout	86

1 Introduction

Illustrated Technical Manuals have become a necessary attachment to new products developed in the industry as well as in the open-source and open-hardware communities. No product can be shipped without instructions on how to assemble and commission it. Yet, the process of documenting even simple procedures like the step-by-step assembly of a machine can be very time-consuming for the documenter. The reasons for this are, among others, the multitude of tools required (Camera, Notepad, Dictaphone) which all need to be operated separately and their output joined into a coherent, well structured manual.

The organisation of information has been developed to the point where larger companies contain all information about their product families and sub-variants in one Content Management System. This allows them to create a manual for a new product from existing text-blocks, which are already proofread and translated into all target languages, thereby avoiding the cost of having to write and translate each complete manual into all languages everytime some part of the text is changed. Even though these tools only provide the means for data organisation, as opposed to data creation, they provide a benefit for large multinational companies. Smaller companies and freelance individuals do usually not have the budget to acquire and operate a CMS. Also, smaller entities are less likely to need to supply their documentation in many languages, so a CMS may add more complication than improve the status quo. The end result is, that small-scale enterprises still have to create and organize their documentation manually. The same is true for the growing community of open technology contributors on the internet. Many good ideas get published online for everyone to use, but without even nearly adequate documentation. This can be attributed to the fact that making a decent-looking, readable manual from scratch is indeed a tough, time-consuming job.

When creating the draft of an instruction manual, the information for each step has to be recorded. The information then has to be grouped in a way that all components (images, text notes, audio notes, etc.) can be clearly seen to belong to a specific step as opposed to another. Retaining all details relevant to a specific step in the assembly process across different media requires strong organisational skill on the documenter's part, and manual grouping and naming of all the images, text-notes and audio snippets creates considerable overhead for a seemingly simple thing like writing an instruction manual. Anyone who productively participated in a technical project during work, studies or internships was likely

required to create documentation for the people who might use the result of their work. Usually this documentation consists of images, bullet points and unstructured text hastily thrown together in an office program and exported to pdf. Also, company-internal projects such as inhouse-developed testing facilities are at risk to be documented that way, if not at risk to remain undocumented.

Motivation

The reasons for the lack of adequate technical documentation are manifold, one of them being the time-cost and complexity of the micromanaging involved in the writing of instruction manuals. Having been on the receiving end of unsatisfactory documentation and likely having produced my share of it, I consider the documentation process a good candidate for optimisation. Using software to take over the micromanaging of files and the control of hardware tools such as camera, notepad and voice-recorder seems to be a promising and as-of-yet untaken direction for improvement of the status quo in technical documentation.

Goals

The main goal of the software shall be to provide an open-source tool for documenting technical processes with images and text. The focus of this thesis will be on the subjects of assembly, maintenance and commissioning of both hardware and software. The software aims to reduce the workload of the documenter by providing a structure for all relevant data and taking over the micromanagement of filenames and locations. Furthermore, the software will unite the functions of dedicated camera (using a USB webcam), dictaphone and possibly camcorder in one interface. The documentation will be saved in an accessible format which allows e.g. the images captured to be edited with graphic editing software if necessary. The software will also serve as a viewer for this form of multimedia documentation. Further features will be developed based on the evaluation of the current state of the art in the field of technical documentation, and a functionally sound feature set selected. At the conclusion of this thesis, the version 1.0 of the software will be published, which will serve as a basis for further development.

Restrictions

The software will be developed based on an initial prototype created during the preparation time for this thesis. The prototype enables the user to capture images via a webcam connected to the documenter's PC, and contain them in a format structurally resembling an image gallery where each image has a dedicated text annotation attached to it. If more than

one webcam is connected to the PC, the software provides the means to select a specific one. The emphasis in terms of features to be developed during this thesis is strongly on the documentation of one specific process at the time, e.g. the maintenance of a carburettor, the assembly of a robot arm, the commissioning of an industrial machine.

Not considered during this thesis are features and functions related to project-management above the lowest process level. Each documentation will be for one specific task, such as the process of disassembling a component, replacing worn out parts, and reassembling it. Reusability or sharing of documentation components between projects is explicitly not the focus of this thesis. Versioning shall also not be considered in this thesis, although saving all projects in common text and image formats would offer possibility to use versioning tools such as SVN. The created documentation-projects will only support one language (presumably english, although nothing hinders the user to use other languages for text and image titles), but the structures to support future implementation of multi-language capability shall be integrated in the design phase if there is time.

2 State of the Art

2.1 Selection of Sources

The documentation technology, literature and standards which are reviewed in this chapter qualify as relevant if they are in whole or in part focusing on the creation of step-by-step documentation for technical processes. The emphasis has to be on the content creation and structuring, not just on management of content created in third party tools. Exemplary of the latter are, e.g., Content Management Systems which only govern the text and images created with a multitude of standard tools such as rich text editors, 3D CAD applications and graphic modification tools. Such systems are outside the scope of this thesis as described in 1.

To evaluate the state of the art, the following media will be considered:

- Books about the principles of technical documentation, theoretical as well as instructional ones
- Standards and Guidelines from industrial organisations such as DIN, VDI and VDMA
- Existing manuals of state-of-the-art technology

2.2 Literature Review

Instruction manuals can be created in many different ways. Tools of arbitrary complexity exist for Arranging information as desired by the technical writer, but the actual creation of information (Text, Annotating pictures with arrows and numbers and the like) is still done in dedicated standard software like MS- or Open-Office and drawing tools from MS Paint to Adobe Photoshop.

The challenges in creating Illustrated Technical Documentation remain manifold, despite the availability of commercial Content Management Systems and ubiquity of computers at the workplace. According to (Piehl, 2002, p. 89) the most essential criterions for users of technical documentation are simplicity, structure and order of information as well as the proximity

of relevant information to the current text location. In (Juhl, 2015, p. VII) essentially the same notion is formulated from the documenters perspective; Documentation should contain as much as necessary, as little as possible and be as clearly structured as possible. The Phases of the creation of an instruction manual are outlined by Hahn (1996):

1. Gather Facts, draft
2. Integrate Facts, extend
3. Refine Facts, polish
4. Assure Quality, test

Whether or not a piece of documentation can be understood does not just depend on its factual accuracy and good structure. (Hahn, 1996, p. 49) names precision, completeness of instructions, and the careful consideration of the target audience as essential points to consider. The target audience needs to be defined at the beginning of the creation process, to make sure the information is presented in a way which is readable for them. If a technical manual contains several chapters intended for different target audiences, e.g. a chapter on how to operate the machine intended for the instructed machine operator, and a chapter detailing maintenance intended for the trained maintenance technician, each chapter should have the target audience defined at its beginning (Hahn, 1996, p. 63).

Most manuals use a combination of text and images to make the instruction more clear for the reader. In such cases where two media are used, one medium takes the lead role and the other one is used as a supplement (Piehl, 2002, p. 137). The lead medium should be recognized as such by the reader, and should ideally be placed in the location where the reader will look first, which for Europeans used to reading left-to-right and top-to-bottom is at the (upper) left in horizontally arranged documents, with the supplemental medium on the right, or at the top with the supplemental medium below in vertically arranged documents (Piehl, 2002, p. 137). (Juhl, 2015, p. 291) also points out the importance of considering the habitual reading direction. One interesting point was made in (Hahn, 1996, p. 97): The author recommends a 2-column layout for illustrated manuals, and suggests that Europeans are primarily text-readers, and as such the text information should be put on the left to be seen first, while Americans are picture-readers, who need the image as the lead-medium on the left of the page. Furthermore, he stated that he expected the European habit to slowly change to picture-reader behavior in the next 20 years. Note that Hahn (1996) was published about 20 years ago. The importance of combining instructional text with illustrations is emphasized throughout the reviewed literature. (Piehl, 2002, p. 134) explains that combinations of text and images present partially redundant information to the reader, which is encoded in two different ways (visual and textual), making lossless recollection at a later point more likely.

A critical part of creating technical documentation is to make sure there is no loss of key information between fact research (e.g. the documenter going through the motions of a machine to get a feeling for the procedure) and the writing of the manual. One way to record all information down to spontaneous ideas which the documenter only gets during the handling of the product and later forgets, is to use a personal audio recorder (Hahn, 1996, p. 54). During the familiarising phase with the product, the documenter can also record all his thoughts as a video with audio track, to be able to store all visual and verbal key information about the process quickly without having to stop and use his hands to write a note (Hahn, 1996, p. 55). Another advantage of bringing a photo or video camera during the initial fact-collection phase is according to (Hahn, 1996, p. 134), that concept pictures of the machine and its controls can be taken from the perspectives of which the documenter intends to have renderings or line drawings made by professional illustrators at a later point.

Regarding the instructional texts in technical manuals, (Juhl, 2015, p. 179) strongly recommends to use clearly structured text in the place of bigger blocks of plain text containing several instructions. Bullet-point lists or enumerations are recommended for their inherent clarity (one list item constitutes one instruction), which makes them readable as well as searchable (Juhl, 2015, p. 179). In KUKA Roboter GmbH (2015), bullet-point lists are used to present instructions hierarchically (two levels) in a readable way. The lower hierarchy is marked by differently coloured bullets and through indentation. For marking lower-level sectioning of the instructions by topic, sub-headlines or marginalia can be used (Juhl, 2015, p. 179). Marginalia are also useful for providing one-sentence summaries of a section or paragraph to allow the reader to recapitulate a section quickly (Juhl, 2015, p. 179). In KUKA Roboter GmbH (2015), marginalia are used as informal sub-headers as well as captions for some of the tables, while other tables are not captioned at all. Guidelines for textual formulations in technical manuals are provided by (Hahn, 1996, p. 40ff):

- Sentences should be as short as 13 words
- Active verbs should be used throughout the instructions
- Command form should be used for the instructions
- One comma per sentence, always a colon before a list or a summary
- Terminology should be kept consistent
- Abbreviations should be explained right away or in a glossary

Furthermore, (Juhl, 2015, p. 109) suggests that repetition of the instructions in the header above an instruction set can help the reader find sections relevant for him, and remember the instructions better. In KUKA Roboter GmbH (2015), instructions presented as bullet-points are preceded by a short header sentence, which serves as a sub-header and helps the reader to find out whether the following section is relevant for her. Tables are named in

(Juhl, 2015, p. 179) as the most structured and neat way to access and compare information. Tables are particularly useful for presenting technical specifications of a product, like exemplified in (Juhl, 2015, p. 253). One thing which can be criticised in KUKA Roboter GmbH (2015) is, that tables are not consistently captioned and at no point assigned a unique number code to help communication between readers trying to coordinate themselves. Some tables are captioned using marginalia, while some are placed under a corresponding image with the image number and caption positioned between the image and the table, which the skimming reader will see as a link between the two entities (KUKA Roboter GmbH (2015)). Yet some tables remain formally unconnected to surrounding instructions or text, leaving it up to the reader to connect relevant information.

An interesting point about optimal image size in manuals consisting of text and illustrations is made in (Piehl, 2002, p. 110): The image should be 7cm by 5cm at a viewing distance of 25cm, to enable the viewer to take in all important details at a glance. According to Piehl (2002), smaller pictures would obscure details, while much larger pictures require the eyes to wander too much to take in all important details effectively.

Linking features in the illustration to the instruction text can be achieved in different ways:

1. Components can be marked by numbers and referred to in instructional text by these numbers in brackets, whenever the concerning component is named (Juhl, 2015, p. 103). This works well in situations where one component is referenced by instructional text at times of the procedure, and does not require the numbering on the illustration to follow any particular rule.
2. In situations where an illustration is used to introduce components to the user, they can be marked by numbers and referred to by an enumerated list, with each list item's position number corresponding to a number on the illustration, followed by the component's name and ideally a short description. This works well in KUKA Roboter GmbH (2015), and requires that the components are numbered starting from 1.
3. The component names can also be written on the illustration, and e.g. linked by lines to their respective component. Then, the components can be referred to by their name only, not requiring a position number (Juhl, 2015, p. 140). In (Juhl, 2015, p. 71) it is suggested that using only numbers in the image makes it easier to find components referenced in the instruction text, because numbers are more searchable than words.

There are also several ways of linking the position numbers in an illustration to specific components:

1. Components can be marked by numbers which are directly placed on the feature, so no connecting lines are required. (Juhl, 2015, p. 85). This works very well in KUKA Roboter GmbH (2015) in situations where illustrations are not too crowded.

2. If the illustration contains adequate whitespace, position numbers can also be placed there as not to obscure the product rendering. The numbers can then be linked to the product components using thin lines (Juhl, 2015, p. 70). In KUKA Roboter GmbH (2015) this creates very readable illustrations with minimal obfuscation of the product rendering.

According to (Juhl, 2015, p. 71), numbering in images should start with the number 1 placed at the 12o' clock position in the image. The same is mentioned in (Hahn, 1996, p. 72), with the addition that numbering should continue on clockwise, and that only those components referenced in the accompanying instructions should receive a number. (Hahn, 1996, p. 72) also states that it makes sense to use different component names in the instructions than used on the BOM, as long as the naming is kept consistent throughout the manual, if parts of the target audience might not understand the internally used names. Numbered Bi-directional arrows can be used in illustrations to denote the order in which several cables need to be connected between two devices as portrayed in (Juhl, 2015, p. 129). In KUKA Roboter GmbH (2015), bi-directional arrows labeled with textual shortcuts are used to illustrate the directional loads in the robot arm, with straight arrows being used for forces along arm components and arrows connected by a curved line for torque in the robot's joints. Regarding visual annotations in illustrations, (Piehl, 2002, p. 121) points out that illustrations should not be overloaded and care has to be taken that annotation types are not confused, e.g. similar arrows should not be used to signify movement of a lever and to point out areas of danger. For creating fault-analysis instructions, both tables and flowcharts can be used effectively, according to (Juhl, 2015, p. 265)

Regarding safety, (Juhl, 2015, p. 22-24) recommends to follow the pattern provided by DIN EN 82079-1 for their completeness of relevant information. (Hahn, 1996, p. 113, p. 120) also states the importance of not just telling the user what he shouldn't do, but also specify exactly where the dangerous part is located, what could happen (worst case) and why, and through which measures it can be prevented from happening. In KUKA Roboter GmbH (2015), most warning labels are boxes embedded in the instruction text-flow, which are labeled with a warning sign (black and yellow triangle with an exclamation mark) and some unstructured text about possible hazards. This form of a warning does not make the severity very clear to the user, and the unstructured text hinders comprehension and retention severely, and does not invite the user to carefully study the content. Both DIN EN 82079-1 and ANSI Z535.6 require a chapter called Safety, which contains all general warnings about a product if it is not perfectly safe (Juhl, 2015, p. 22-24). When using warnings in text and pointing out the parts of the product the warning is about in an illustration, care should be taken to keep the colours consistent between the two media, because a warning text in grey may not intuitively get associated with red circles or arrows in the picture (Piehl, 2002, p. 105). Colours should be used consistently throughout a document; when selecting colours for any text component

(warning, information, headlines, etc.), the pre-existing habits of the target audience should be taken into consideration (Piehl, 2002, p. 126).

The quality of an instruction manual can be reviewed using checklists provided by DIN EN 82079-1 (Juhl, 2015, p. 364). If the manual contains a lot of information, or if different parts of it are intended for different audiences, it is useful to include a section at the beginning which outlines the intended way of use for the instruction manual (Juhl, 2015, p. 199). For thick printed manuals, cut-out tabs for separating chapters on the right side of the pages have proven a useful navigational aid for the reader (Juhl, 2015, p. 209). Lastly, (Juhl, 2015, p. 348) emphasizes that a manual has to be presented on an appropriate medium, meaning that an assembly manual for a new PC should be included in paper, while the manual for an office-software can be provided in digital form since the user of the software definitely has a working pc on hand when operating the product.

While reviewing the literature, there were a few instances where the future of technical documentation was discussed. (Juhl, 2015, p. 298) notes that there is no useful standard regarding the format of electronic manuals, stating the commonly used .PDF is not qualified as such because it is conceptually similar to the paper form and does not embrace the possibilities regarding flexibility of structure and multimedia content offered by modern computer technology. (Juhl, 2015, p. 355) describes the possibility for new standards of information presentation to emerge over the next years, and frames the optimisation of presentation structures and instruction creation methods and tools as a task for the near future. As mentioned before, Hahn (1996) predicted in 1996, the changing of the user habit from text-reader to image reader would occur over the next twenty years at the time of writing. In KUKA Roboter GmbH (2015), the information accompanying images is always located below the image, indicating that the image has become the lead medium.

3 Analysis

3.1 Viewpoint: Creator of Illustrated Technical Documentation

3.1.1 Evaluation of the State of the Art

If the advice from the reviewed literature was followed by every technical writer, Illustrated Technical Documentation would not be a field requiring many improvements. The concepts presented are clearly formulated and, where necessary, backed up by information from other scientific fields like e.g. psychology.

The main challenge a technical writer faces is therefore not to think of a good manual structure or how to make safety warnings stick in the reader's memory, it is to:

1. Find the available information regarding the creation of illustrated manuals
2. Find time to actually read it and reflect on it, which may be difficult if employers do not pay for the time they spend learning their job
3. Apply it in their local context
4. Defend themselves against criticism that documentation process is going too slow, because thorough documentation will cost more time

This became very apparent during the review of the practical example manual. While the layout looked good at first glance, the little things such as some tables lacking a label and any references in text and the unstructured text in warning labels are crucial when it comes to the evaluation of a manual's quality. Tables which do not have a label can hinder communication between users of the manual, e.g. saying 'The table after the second paragraph on page 5' is less precise than saying 'Table 1-3 on page 5'. Unstructured text with a warning black-and-yellow triangle beside it is not readable, because the reader gets worried about what might go wrong and has trouble focusing on the text well enough to get the actual information at the first try.

One problem which arises during the creation of Illustrated Technical Manuals is, that it requires the help of many software- and hardware tools which all need to be operated separately and their output joined together more or less manually in a meaningful way. Notes can be taken on paper or an electronic device with touchscreen or keyboard. Audio drafts can be recorded on the ubiquitous mobile phones, they can also capture image and video material if the documented product is of appropriate size and well lit.

The files generated are thus stored on the devices with an automatically generated name, often a timestamp of some sort. In some situations, the preview on the small screen of a handheld mobile device may not be sufficient to judge whether the image is suitable, so several images may have to be taken and saved for later review on a PC screen. When the documenter wishes to get the files, he needs to connect the mobile device to a pc, copy all the files manually, and then start sorting. The sorting process takes time, because all the images have to be evaluated and the most appropriate ones chosen. The image files will then have to be renamed following a pattern the documenter can recognize later, e.g. 'step5_componentXY' and the unsuitable pictures moved elsewhere as backup. The audio and video files may be fewer in number, but cannot be reviewed thoroughly at increased replay speeds and therefore also take time to sort out.

Next, the drafted text has to be cleaned up and structured in a text editing tool, and the appropriate accompanying images / video clips identified. The images may then need to be annotated to connect them to the text, e.g. with position numbers or arrows, which may require a separate tool. The text has to be double checked to ensure no labeled item in the image remains unreferenced in text and vice versa. Many of the text-structure guidelines put forth by the authors of the reviewed literature are very specific to the field of writing documentation, so established word-processing software will not provide automated means to check them. Writing in short command form may trigger an office-software's warnings that sentences are incomplete but it will not underline a sentence because it is longer than 13 words and uses more than one comma. This effectively means the writer of technical documentation in industrial context can not rely on assistive technology which has been taken for granted by school students writing their homework for over a decade. Also, while it is technically possible to use a compact device like a personal or company-issued smartphone for recording audio, pictures and video, such devices are restricted in some companies because of security concerns. In such companies, the documenter would be thrown back to dictaphone, digital photo camera and possibly a separate digital camcorder, further increasing the number of tools required by up to 2.

This rundown still does not include the possibility that the illustrations may be created by a dedicated office in the company, or even given out to external contractors which complicates things by another few steps of due process.

All reviewed theoretical literature has pointed out to some degree the problems technical writers face during their everyday work:

- Time pressure, because the documentation (which can often only be created when the product is effectively ready) is the only thing standing in the way of the launch
- Lack of support because documentation teams are considered costly, unwanted necessities in some companies
- The dependency on other parts of the company which are generally very busy with their own work (Engineering for understanding / trying out the product, e.g. marketing because they have illustrators on hand)
- If a check of their work turns up a problem, the other (busy) colleagues from engineering and illustration need to be involved again, offering great potential for working overtime

The key weakness of the current state of the art can therefore be described as a lack of convenience for the documenter, which has adverse effects on the documentation the reader receives. The challenges of the documenter can be summarised by phases as described in 2.2. Also, there is no common truly multimedia-capable format established, which lets the documenter use still images, audio recordings or video clips (e.g. animations) where it makes didactic sense, as well as allowing closely linked structured textual annotation. This could, in theory, be programmed in HTML, but manually doing so would not solve the workflow problems and add another layer of complexity.

3.1.2 The Documentation Process to consider during the Design Phase

Phase 1: Gather Facts

The fact-gathering phase is roughly the same for all products. The product is analysed by the technical writer, possibly under guidance from an engineer who developed the product, and time is limited between other responsibilities these employees have. Here, the documenter needs a toolkit that lets her capture the facts with minimum distraction and time consumption on the side.

The types of information which need to be picked up during this phase are:

- Pictures and Video of the machine or its components
- Audio, mainly for quick notes regarding points where special attention needs to be paid.

- Required components and tools for each step
- Additional parameters or restrictions

The potential for losing even small packets of information should be kept as low as possible, because any inaccuracy or wrong understanding will affect the documentation quality for the rest of the process, and if discovered late, can be very costly to correct. The toolkit should also allow immediate naming and organisation of the recorded materials to avoid the need to rename the files using the file explorer later, and to make finding specific components easy. The draft of the documentation should be saved in common file formats to ensure compatibility down the road (External editing of images, text translation, CMS).

Phase 2: Integrate Facts

In this phase, the documenter needs to add information such as warnings about possible dangers as well as the specification of the user group. Furthermore, audio notes need to be transcribed to text and video clips for the multimedia version of the manual need to be cut to size. Ergo, the documenter needs to go through the draft manual, review all drafted information and transcribe it to text where necessary. Pictures take may need to be replaced with renderings (if the company style guide requires it) of the same perspective, and annotated with e.g. position numbers, arrows, or other highlighting elements. References to other chapters have to be included where necessary, or information made redundant where referencing does not make sense. Ideally, the manual is given out to be proofread by the development staff at the end of this phase. This requires a format which they can easily read and annotate if mistakes are found.

Phase 3: Refine Facts

The refinement phase ideally produces the finished version of the documentation, as intended. Any corrections from the proof-reading round in phase 2 should be integrated. In certain cases, a quick-reference manual can be added for the frequent users who only need keywords and do not wish to flip through all the pages again and again.

Phase 4: Assure Quality

In this phase, the manual is given a final check using the checklist provided by e.g. DIN EN 82079-1. If all checks are passed, the manual should then be tested on a group of people who should meet the requirements of the target audience as closely as possible. Overqualified people are just as useless in this case as the underqualified. If any problems

turn up during the live tests, solutions can be found and integrated. If everything has been worked out and there are no more issues, the piece of technical documentation can be published.

3.1.3 Conclusion

The theoretical state of the art is well developed and the basic criteria for high quality in technical documentation have not changed much in recent times, with the notable exception of other media becoming available. Yet, the documentation quality described in the theoretical literature is not easily achieved in practice.

The first two phases as described in 3.1.2 are the ones where the technology stack of the documenter is most diverse and where the necessity to micromanage every single aspect (e.g. renaming a few dozens or hundreds of images) slows down the process considerably. Those two phases are the ones where a software tool as proposed in this thesis will be of the greatest benefit. By providing an appropriate structural template for the relevant types of documentation and assistive functions to integrate some of the otherwise more time-consuming elements in a quick and well-structured manner, the process of creating Illustrated Technical Manuals can be improved. This should lead to increased workplace-ergonomy for the documenter, as well as more concise and well-structured documentation for the user. The integration of video clips and audio will have to be carefully considered, since inclusion of such information makes the manual essentially unprintable. Nevertheless, the option to do so shows great promise considering the rising availability of computing devices at the workplace.

In essence, providing guiding functions for the documenter, so she can focus more on concise formulation instead of needing to check the handbook for good layouts of e.g. warning labels, will benefit the reader of the instruction manual.

3.2 Viewpoint: User of Illustrated Technical Documentation

3.2.1 Evaluation of the State of the Art

In most situations where a person needs instructions on how to assemble, commission or repair a technical device, a paper manual will be the easiest to handle intuitively. There are some reasons for this:

- Paper and its analogues have been the go-to medium for written words for a few millennia, ever since stone and clay plates went out of fashion
- Paper does not require a replay medium (if light is to be taken for granted)
- It is reasonably durable as long as it is not submerged in water or set alight
- It can be made to contain several types of information: Formatted or unformatted Text, Pictures, Formulas, etc.

On one hand, the combination of text and pictures on the paper medium has proven to be at least adequate in getting instructions across, although the level of adequacy depends greatly on the structure and formulations of the text and the quality of the illustration. On the other hand, paper has several shortcomings which have only become apparent with the rise of digital media:

- It can not display moving pictures, which in a time where products are increasingly complicated may be to the detriment of the readers understanding
- It can not contain sound samples, e.g. warning tones a product might sound out, samples of product health indicators (sound of a healthy engine vs. sound of crankshaft bearing damage) or spoken instructions
- The manual's structure is rigid once printed. There is no convenient way to follow references as in digital formats, where a click on a reference navigates the user to the referenced chapter, page and paragraph
- Searching can only be done by reading through all the text, or through an index which needs to be compiled by the documenter
- Paper does not allow editing, so if even a small error is detected post print, it has to be reprinted in full and the faulty prints discarded
- Paper manuals for consumer products are often misplaced because they are secondary to the user until a problem occurs. A digital file can be hosted online by the manufacturer and redownloaded whenever it's needed

Many of the shortcomings have not been apparent for long enough to be a problem for people, since most people using instruction manuals today were already using them before the ubiquity of ever-present multimedia capable devices. Also, If the text instructions do not adequately describe a complex action, it is easier to attest the writer a lack of talent than to realize that an animation could have made all the difference, but could not be included because of the distribution medium.

3.2.2 Using the Documentation

When opening the instruction manual for a specific task, e.g. maintenance of a product component, the reader needs to be confronted with the essential information first:

- Definition of the intended audience, to be able to judge if the process can be conducted by herself or if specially trained personnel needs to be called in for the job.
- General, well formatted safety instructions regarding the process
- A list of materials (tools, product components) needed to perform the operation; also, information about the number of people needed
- A register explaining the terminology used

A sort of index needs to exist, preferably in a form that can be studied at any point without leaving the current page. In digital media it is paramount that the index provides navigation links. The instructions need to be easily readable and well structured, as described in 2.2. At any point in the manual the connections of a table, chart or picture need to be clearly linked to a specific text section. Each step has to start with the requirements regarding tools and parts needed. A subsection of the manual needs to have a short, concise conclusion at the end, which summarizes what has been achieved. If referencing other parts of the manual is necessary, the reader should be supplied with a means to quickly find the concerning information, work through it, and return to the initial location where the reference was placed. The lead medium has to be positioned where the reader will see it first.

3.2.3 Conclusion

Traditional manuals made of paper are a usable form of documentation if they are well-made. They have clear advantages when it comes to long-term reliability because they do not require a power-source to be read, and high-quality paper prints can retain their information for long periods of time (decades to centuries) if stored correctly. The handling of printed information is second nature to most people working today, so as long as the structure of the manual is clear users will generally find the desired information.

Nevertheless, a multimedia manual running on a computer (mobile or stationary) can bring several advantages for the reader:

- While voice recognition for whole sentences is still tricky (see the so-called AI helpers on mobile phones), interpretation of a limited set of voice commands is already easily integrated in an offline application; this can free up the hands of the user to do her work, because she can be enabled to flip the pages of the manual through voice command

- Digital text is searchable by algorithms, the user looking for a keyword can easily list all available occurrences
- A digital manual does not need a rigid page-by-page structure, it may offer different ways to group the existing information at the click of a button, reducing the amount of redundant information
- A digital manual can contain text and images, as well as video and audio recordings where applicable.
- For multi-language manuals, the display language can be chosen right after opening the manual, and pages in languages which the reader does not need are not displayed, decreasing the amount of unnecessary information which may distract the reader.
- References to other chapters / pages can be implemented as a clickable link to navigate to the specific information; navigating back can be done via a reverse-button as it is common in modern file- as well as web-browsers.
- While paper manuals are often printed in greyscale for cost reasons, there is no reason for this restriction in digital media since displays of computers can be expected to display colours; this increases the possibilities regarding the annotations of images as well as the clarity of digital photos used as illustrations.

All of these possible improvements in terms of ergonomics for the reader come at the price of a potentially significant difference in information density and clarity between the multimedia version and the printed version of the manual. This problem can be attributed to the current time being one of shifting paradigms, where people either stick with the old, embrace the new without looking back, or arrange themselves somewhere in-between and accept having to make compromises on both fronts.

4 Developing the Application Concept

4.1 The Initial Concept

4.1.1 Envisioned Use-Cases

The problem that led to the conception of this documentation software was the lack of a suitable tool to document step-by-step processes quickly enough to warrant its use for a non-recurring task. Primary trigger was the need to completely disassemble a set of constant-pressure carburetors, clean them in an ultrasonic bath and put them back together the way they were.

For such an undertaking, using a smartphone to take pictures appeared to be a feasible solution. The drawback was, that pictures on the phone have no context attached, and they cannot easily be given filenames. Additionally, holding spring-loaded parts in position while taking a picture is a risky undertaking, as well as a potential health hazard if no eye-protection is worn.

While the task was completed without the software tool, it did confirm that applications for such a tool exist.

Further development of the idea led to the isolation of three primary use-cases:

- assembly manuals
- commissioning and operations manuals
- maintenance manuals

4.1.2 Software

The concept to capture information in a gallery-like format (essentially a bi-directionally traversable list of information items) was part of the initial conception of the project. Originally, the concept was envisioned for hobbyists and contributors of the open-source and

open-hardware communities, and use by commercial entities was not considered at all except for the most rudimentary drafting purposes. For that reason, the initial concept was more focused on features which would make the use of the application more convenient for the documenter, e.g.:

(Features which were not discarded until the selection process are labeled by their identification number in brackets which matches them to the indices in A-1 - Considered Features)

- A PDF Export function was conceived early on, as there needed to be a way to read the created documentation on any system without installing the software first. [1]
- Controlling the GUI via voice commands when reading or creating the documentation, such as: [2]
 - Navigating forward and back through the gallery by saying the respective words
 - Commanding the camera to take a snapshot or record a video
- Recording audio draft and saving it to file for replaying at a later point, enabling a manual transcription by the documenter. [3]
- Alternatively to [3], recording the audio and using a Speech-To-Text engine to create the first draft of the instruction text from it. [4]
- Add two sets of instructions to one project, so an assembly manual and its counterpart are inseparable. [5]
- Graphical annotations to the pictures were also considered as a feature. [6] This includes enumerating items on the picture using numbers, highlighting them with a coloured frame, and possibly inserting text.
 - For the integrated solution, an algorithm can make sure that there is a reference in the text for any enumerated item on the picture.
 - The graphical annotations do not need to be merged with the image, but kept on separate layers to enable editing at any time.
- Filtering webcam images in a way that made them look like line-drawings, to make the images visually simpler and improve printability.
- A tag-based search function, where each slide is annotated by the documenter with keywords about the contained information, which can then be searched by the reader. [9]
- For importing images from the filesystem, the idea of using links was considered: [13]
 - The image will be re-loaded from the specified file in the filesystem at startup, and saved in the project directory only to make the project transferable.

- If the base image is modified (e.g. re-rendered in case of a rendering, or annotated with arrows) the image in the Gallery is automatically be kept up to date.
- The image in the project directory is loaded only when the original link becomes invalid.
- Creating reversible manuals for simple assemblies, where only the assembly needs to be documented. The order of slides in the Gallery can then be switched at the click of a button, and the same documentation can be used for disassembly as well. [36]
- Feature [6] was also considered in a reduced form, where only enumerations can be added and are merged with the picture permanently. [37]
- Enabling the documenter to record video clips with the webcam and add them to a slide instead of or parallel to a still image. To preserve printability the documenter must be forced to select a suitable still picture as a printed replacement for the video [40].

4.1.3 Hardware

The initial concept also contained optional external hardware besides webcam, such as a foot-switch and a touch- or button-pad which provides access to the buttons of the gallery without the need for a keyboard or mouse. The main reason for deciding cross-platform portability to be a cornerstone of the development, was the idea that the application could be run on some of the available ARM-based one-board micro PCs. This would enable the development of a portable documentation tool, by just adding a battery, an arm-mounted touchscreen and a head-mounted webcam to the microcomputer. Such a portable documentation system was conceived to find application in the documentation of e.g. damage in large machines which need to be entered by humans to assess their state of repair, or motor vehicles. Nevertheless, the primary application was from the beginning to install the system on a desk or workbench, where e.g. a new product would be assembled for testing and documentation purposes. The process could then be captured without the needs for external notes or any delay between action and writing, where small crucial details may get lost.

The first realism-oriented concept-drawing 4.1 contains:

- The workdesk
- A Large screen mounted at the back of the desk for viewing the documentation

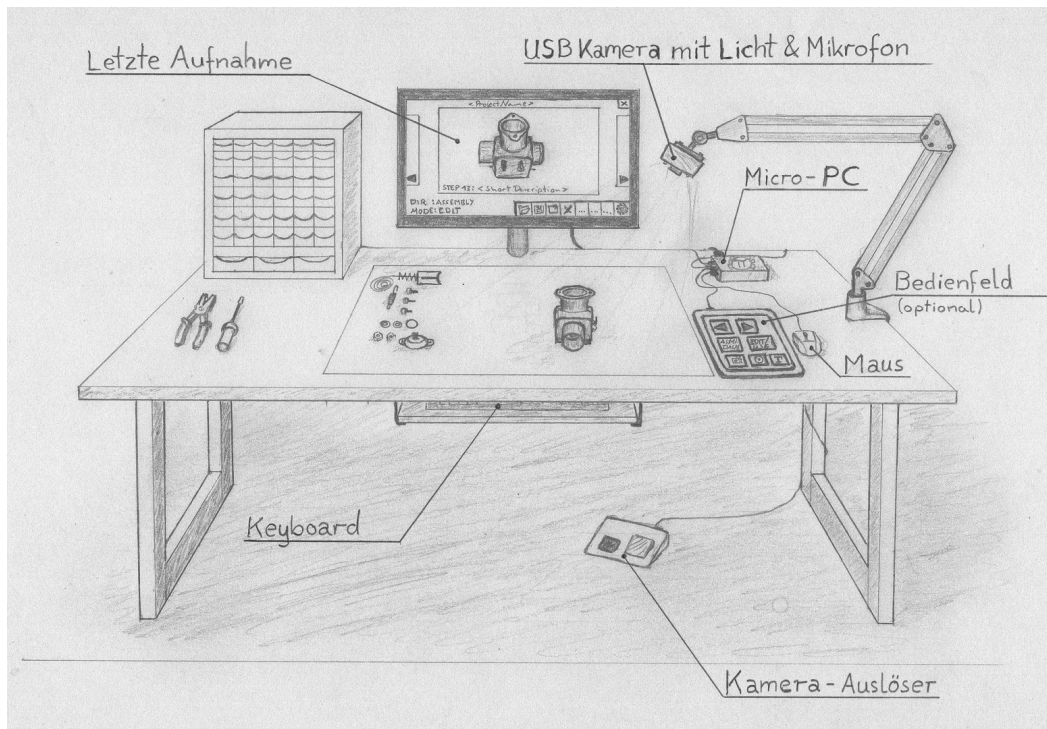


Figure 4.1: The first detailed conceptual sketch of the system in use.

- The webcam, mounted on a flexible arm like e.g. a desk lamp for flexible positioning and a steady camera (Ideally combined with working light, to have additional illumination on the subject assembly whenever needed).
- A Keyboard, used for writing the textual annotation
- A Mouse for operating the software
- A Touchpad which gives quick-access to the most important functions (e.g. forward, back, take snapshot, record audio, play audio)
- A Footswitch for taking pictures while positioning the subject with both hands in front of the camera

4.2 Proposed Additional Features

4.2.1 Additional Features gained from Reviewed Literature

Some additional feature ideas could be derived from the reviewed literature, and some existing ones expanded:

(Features are labeled by their identification number in brackets which matches them to the indices in A-1 - Considered Features)

- Importing tables (html or cvs) into the annotation text [16] was primarily inspired by (Juhl, 2015, p. 253)
- Create an automated text analysis tool which checks the text for formatting consistent with (Hahn, 1996, p. 40) [25]
- Force well-structured warnings and notices by providing templates, which cannot be inserted unless filled out [35], this was described by (Juhl, 2015, p. 37 ff) in detail
- Sub-component of [37] taken from the style suggestions by Piehl (2002), Juhl (2015), and Hahn (1996): Add position numbers with optional lines, arrange automatically starting from 12o'clock.
- Manual to the Manual [38]: Provide the reader with instructions on how to use the manual while he reads it. (Juhl, 2015, p. 199)
- Quick-access tabs for chapters [39]: Use tabbed pages for manuals with a high page-count, as found e.g. in thick catalogs and phone books. This idea was taken from (Juhl, 2015, p. 209) and applied to the software application context: Dynamically generate quick-access tabs (in the form of buttons) which lead to the beginning of each chapter. Present them at an easy-to-see location.
- Video Capture [40] was also proposed in (Hahn, 1996, p. 55) as a possibility for future, multi-media capable documentation formats.

4.2.2 Additional Features conceived during the Prototype Design Phase

- Slides for special purposes, e.g. displaying only a table, only text, or only a picture / video [7]
- Saving the project in a single file (e.g. .zip) as opposed to using a directory as a project container [8]

- Portable mode: include an option for a portable mode, where all the application data is saved in the application root to enable startup from e.g. a USB drive [10]
- Watermarks for images: include the option to impress a company logo on to every slide's image, to prevent unauthorized re-use [11]
- Product Variants: Make one manual project capable of supporting several variants of a product, by separating the shared slides from variant-specific ones and facilitate switching between variants via e.g. a dropdown menu [12]
- Project setup wizard: when a new project is started, require the user to put in the most important information about it, e.g. type of the manual, author's name, and the name of the documented product [14]
- Saving hooks: Create options in the settings manual which lets the user set requirements which will be checked before the project can be saved [15]:
 - Require all slides to have a title
 - Require all slides to have a description text
 - Require all slides to have an image
 - Require all slides to be tagged
- Facilitate the creation of a title-slide, which contains all the relevant information on the project as defined in the project settings [17]
- Facilitate the creation of chapters, by inserting special slides which contain a chapter title, chapter number, author and other relevant information [18]
- Enable slides to contain different combinations of information types, such as footnotes, advance organizer, etc., and automatically hide all unused features in view mode [19]
- HTML Generation: Compile the project into HTML-based format and store in the project directory everytime the project is saved. This makes sure the most recent version of the manual can be viewed at any given time an any webbrowser, without the need for specific software [20]
- Toolbox: Add an extra input field on every slide which lets the documenter list the tools required for each step. These can then be automatically compiled accross the project and displayed [21]
- GIF enabled slides: Provide a way to capture and replay stop-motion images such as the .GIF format. [22]
- Safety chapter: Create a dedicated chapter or slide at the beginning of a manual to present general warnings and notices before the instructions start [23]

- BOM Import: Allow importing of a BOM and enumerate all components mentioned throughout the manual according to their part number (require matching of a part-number when a component is to be enumerated, as opposed to letting the documenter choose a number freely) [24]
- Glossary: Create a glossary analogous to toolbox function [26]
- Tooltip Box: Add a dedicated box in the UI which shows context-sensitive information such as guidelines on formulating instructions while the documenter is typing instruction text, what to consider when formulating warnings, etc. [27]
- Selective features: analogous to 19, but manually switch features on and off in editing mode [28]
- Layout retention: Save relative sizes of panels (text, image, toolbox) as defined in the editing mode, to give the documenter more control over the layout of the final manual and make it possible to set an emphasis on certain components. [29]
- Perspectives: Instead of an edit and a view mode, offer three "perspectives" on the information [30]:
 - Draft Mode: For quick and complication-free recording of information, without regard for formatting
 - Refine Mode: For making the information collected in the draft phase presentable and well structured, and adding other important elements such as tables, warnings, etc.
 - View Mode: The documentation reader's perspective on the documentation. Contains the same information as defined in Refine Mode, without any editing functionality
- Navigation History: build in a backtracking-button like found in webbrowsers, which also reverses jumps through the manual done e.g. via the slide list [31]
- User annotations: Let the users write notes for every slide in view mode. Retain through save / load and display on demand [32]
- Image Frames: Automatically give an image a coloured border if the annotation text contains a warning. Set colour according to the highest warning level present, e.g. orange for "WARNING" and red for "DANGER" [33]
- Tabbed slides: Equip each slide with tabs to switch between image, tools, BOM, and other additional information [34]

- References to other chapters: Provide a mechanism to insert hyperlink-like references to an annotation text of one slide, which when clicked navigates to the referenced slide [41]

4.2.3 Evaluating the Features

A total of 41 features (see A-1 - Considered Features for the original table) were drafted for consideration, and analyzed regarding five criteria:

1. A feature's dependency on other features in the list
2. A feature's incompatibility with other features in the list
3. The expected time needed for implementing the feature (this was based on previous experiences and a quick analysis of the complexity of the required tools)
4. To signify how big of a positive impact on the application a feature affects, a relative value was assigned:
 - 1: May occasionally save a few seconds or add some convenience
 - 2: May save a few seconds regularly or add some convenience or redundant information
 - 3: Is a regular time-saver or adds valuable information or convenience in certain context
 - 4: Drastically saves time in certain contexts, adds generally valuable information or general convenience
 - 5: One instance of Level 4 combined with at least one instance of Level 3

The value for the reader of the documentation as well as the creator of the documentation were identified, and the higher one of the two was chosen. While this value was not calculated, the features with the highest impact and the ones with the lowest impact were identified, and the others categorized in-between.

5. The proposed features were given one of the following categories:
 - Documenter-Ergonomy: Functions which make life easier for the documenter
 - Documenter-Function: Features which help the documenter store information more appropriately
 - Reader-Ergonomy Functions which make life easier for the reader

- Readerformat: Functions which make information more clear for the reader, or which add (redundant) information
- Software: Affects SW internals, mainly
- Quality, Data Safety: Features not related to documentation, but to ensure a smooth documentation procedure without trapfalls
- Versatility: Increase the number of situations in which the Software can be used

4.2.4 Composing the Feature-Sets

From the list of proposed features, four feature-sets were created, each with a different focus. A combined implementation-time of around 120h per set was defined as a common baseline to keep the feature-sets comparable.

Figure 4.2 represents an attempt to create a feature set which gives equal benefits to the documenter and the reader of the documentation. This set contains all the necessary components for creating a simple production manual:

- Warnings can be specified, making is suitable for documenting most devices
- Tables can be inserted, e.g. to include a BOM or a technical specification table for a machine
- Longer manuals can be kept accessible by utilising the tag search function
- The finished manual can be exported to the easily printable .pdf format, guaranteeing the compatibility of created documentation with the outside world.

The colour-coded numbers state the priority for implementation.

Figure 4.3 provides the documenter with all the features which help giving a documentation accessible structure. Tables, warnings, videos and graphical annotations can be added, as well as special purpose title- and chapter slides. The toolbox and tags allow storing of additional information which can be searched, and the chapter-tabs feature allows quick navigation to the desired information regardless of how long the documentation is.

The third feature set (Fig. 4.4) is special in that it presents an ideal case. For this set, the time-constraint was disregarded, and a compatible selection of features put together to serve as an optimal reference.

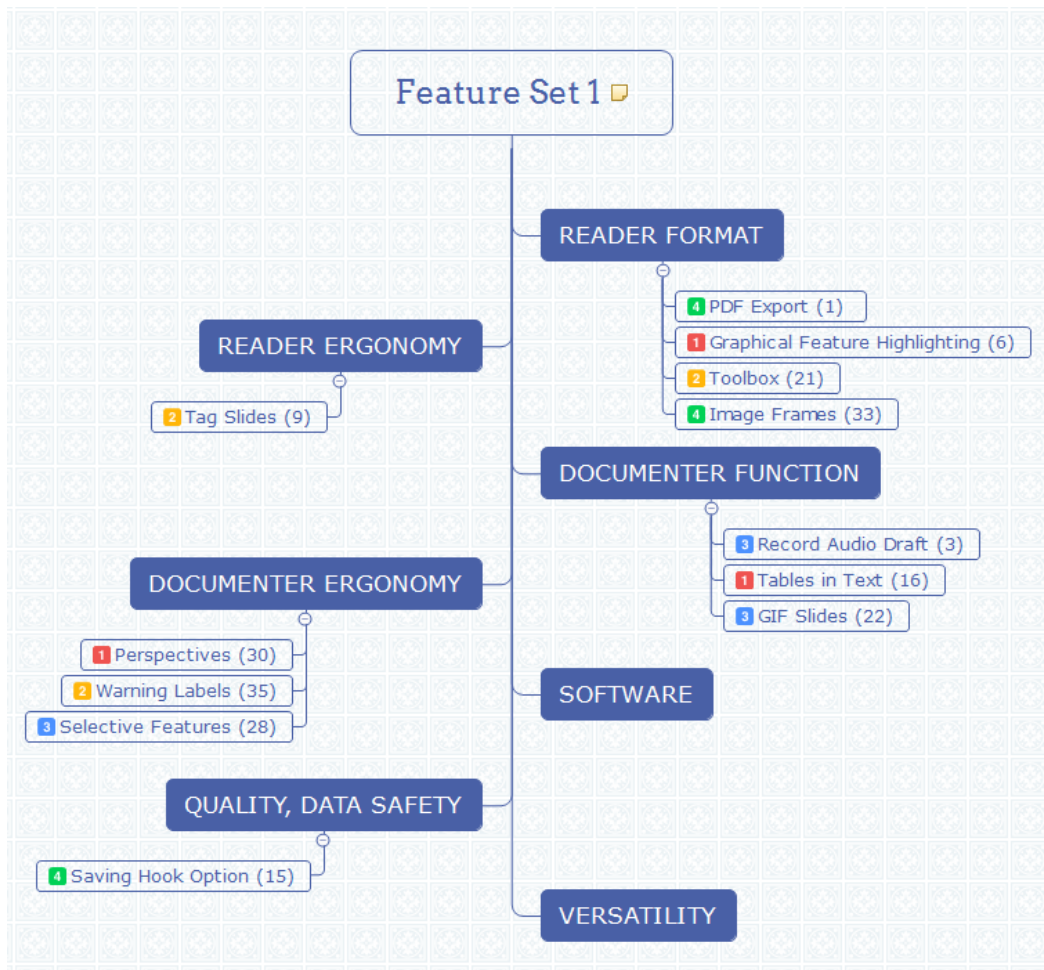


Figure 4.2: Selected features are providing a balance between user- and documenter-orientation.

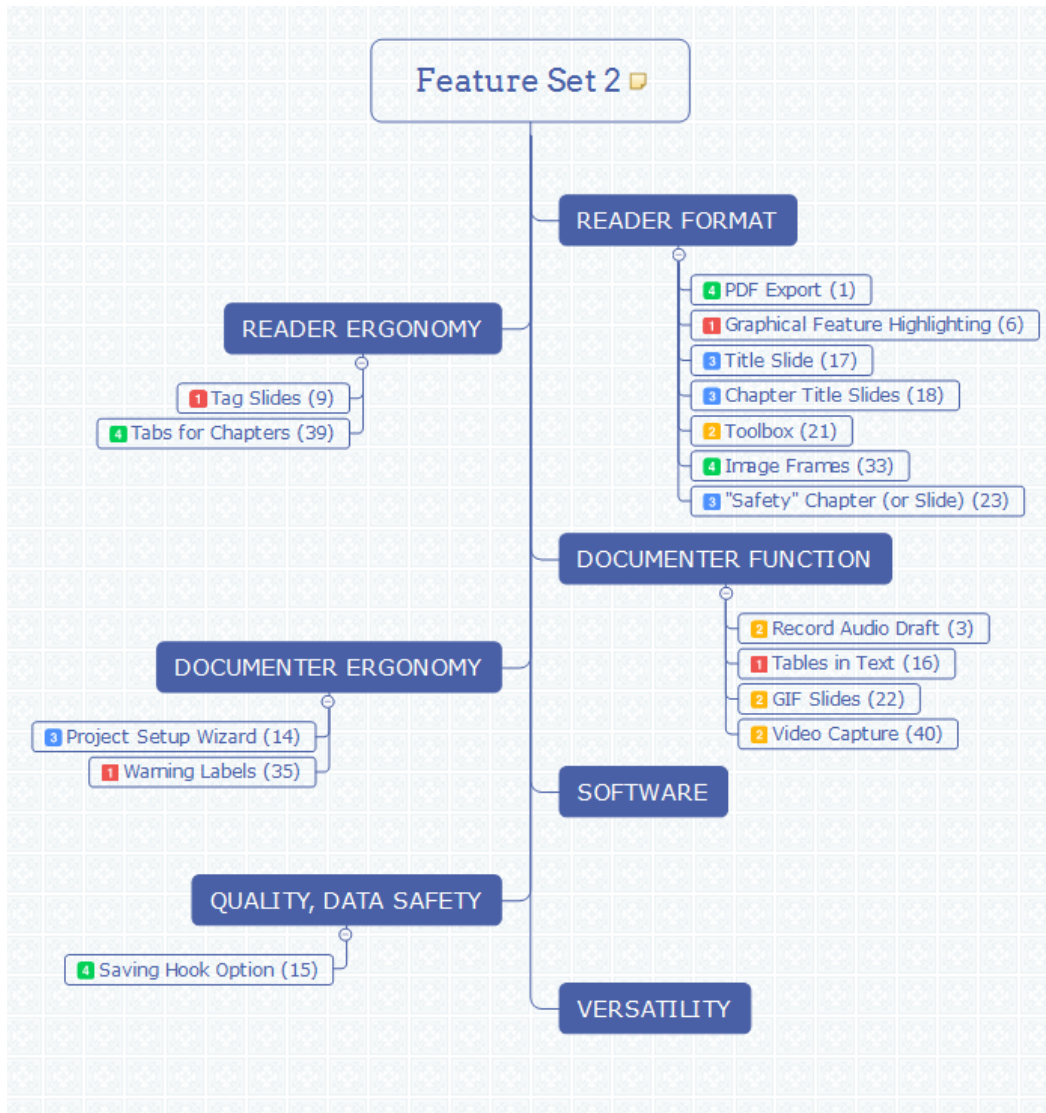


Figure 4.3: Selected features focus on information collection and structuring.

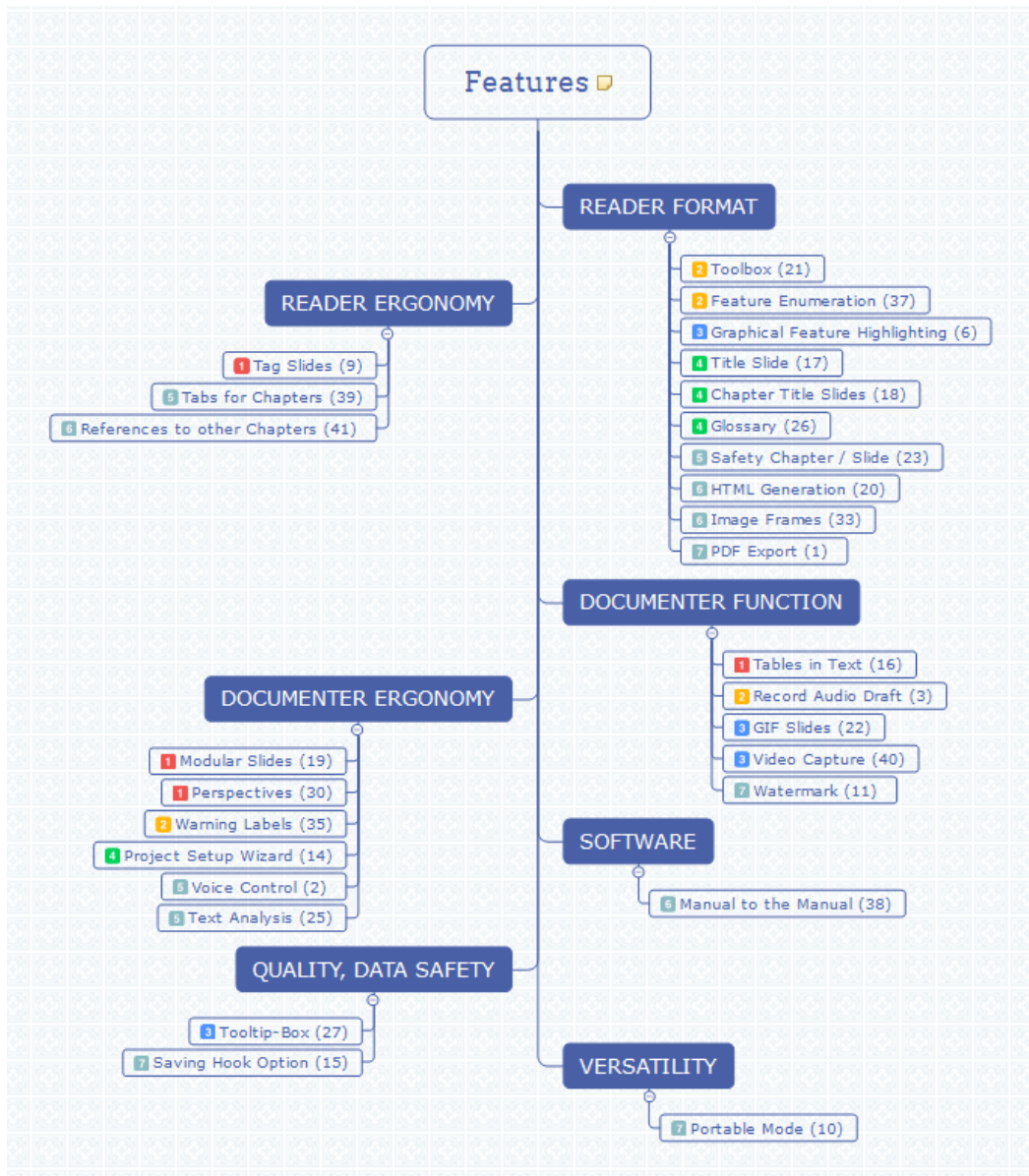


Figure 4.4: Ideal case for implementation without time constraints.

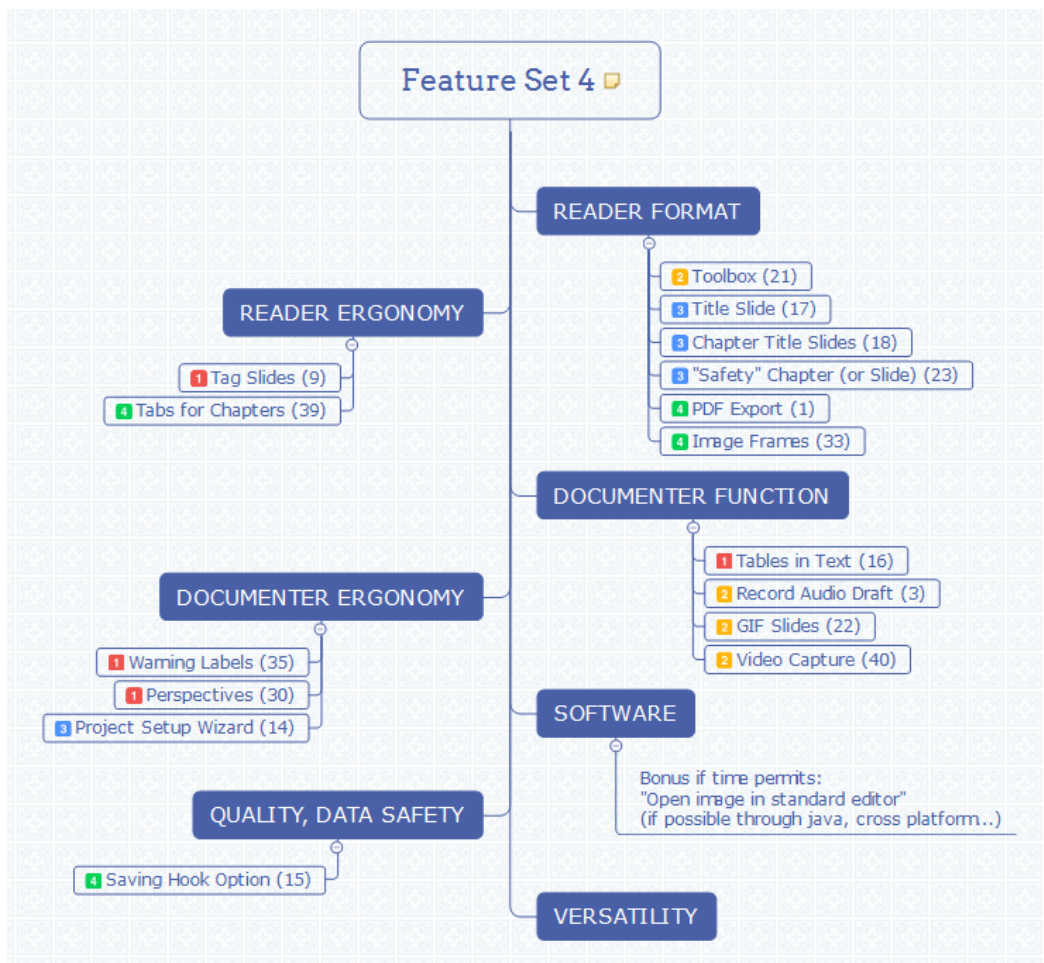


Figure 4.5: Selected features focus on features not available in other free tools, such as word-processors and graphics utilities

For the fourth feature set (4.5), the balance between documenter- and reader-related features was disregarded. Instead, the features deemed least likely to be available in external tools were selected. This led to the image annotations being dropped, since there are several free tools available to edit an image, insert numbers, arrows and other highlights. Features which are not commonly known to be available, such as a warning-label creator or a tool which enables convenient recording of stop-motion images via webcam, are favoured. These features are complemented by tagging functions as well as support for tables and multimedia elements (audio and video). The pdf-export function completes the set in terms of real-world compatibility.

4.3 Selection of a Feature-Set for Implementation

The selection of a suitable set was done by picking a focus. At the beginning, the driving idea was to integrate as many features for the convenience of the documenter and user as possible. During the process of grouping features together in sets (see: 4.2.4) it became clear that with the approach that was selected, the third option would offer a wholesome documentation experience. However, since its projected implementation time is 218h, which is almost 100h above the allocated time budget, it was not a realistic choice. The solution was found in the fourth set. The graphical feature highlighting alone had an estimated time cost of 24h, yet similar capability could be found in freely available snipping tools. From a software-engineering point of view, reinventing the wheel is rarely considered a good solution. The main reason is that if a tool exists which does the job and has been tested, making use of the tool has several advantages over creating a parallel solution:

- No time is spent on concept work
- No time is spent on implementation (besides perhaps some necessary surrounding code to integrate the external solution well)
- No time is spent on testing and debugging. This is perhaps the biggest factor, as debugging can take longer than all the preceding stages together.

Thus, the fourth solution was determined to fulfill the roles which could least likely be fulfilled by free third-party software:

- Organizing the collected information in a quick and efficient way
- Offering direct processing of the drafted information to create a presentable manual
- Facilitating information retrieval through keyword-search and shortcut functions
- Facilitating the distribution of the manual in a common format (pdf)

4.4 The Software-Engineering Process: some points of interest

4.4.1 Designing the GUI

The choice to start with the design of the Graphical User Interface (GUI) was made, because having the visual reference in the development environment during the application logic de-

sign reduces the risk of forgetting to implement a component. For laying out the discreet User Interface (UI) components each feature would consist of, such

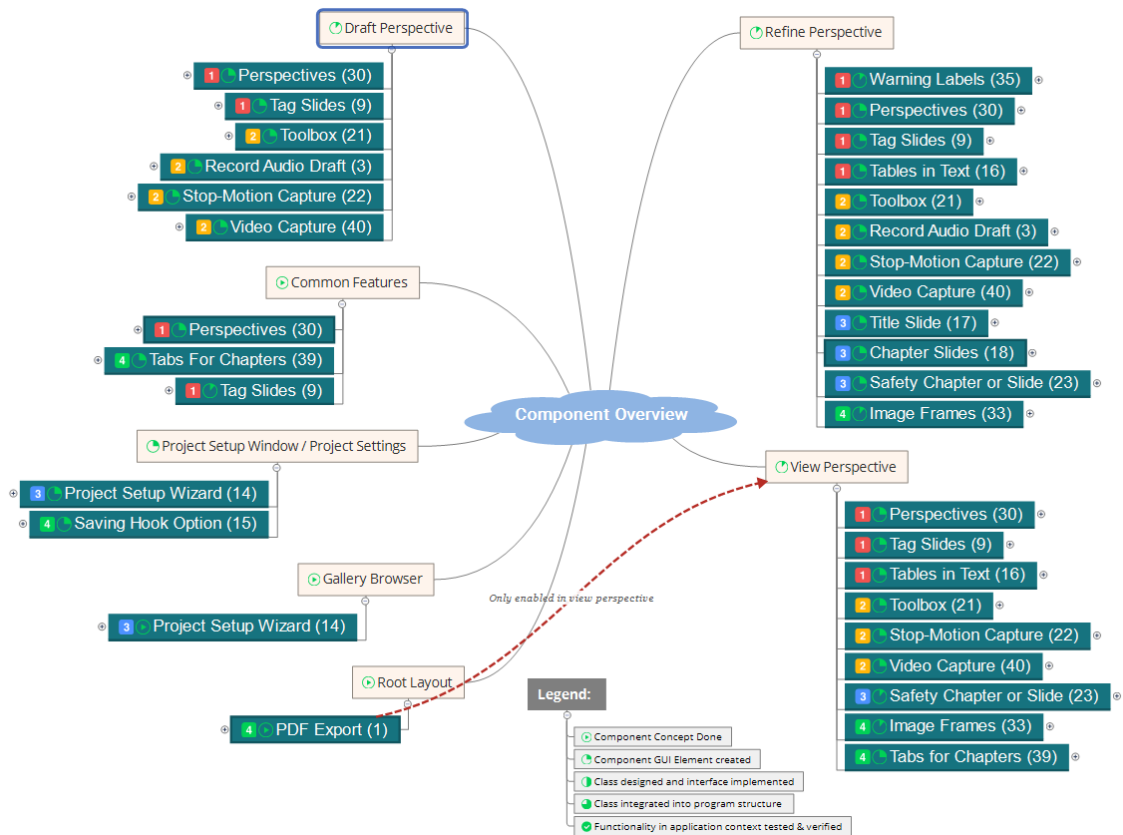


Figure 4.6: The features of Feature-Set 4, grouped in the GUI structure given by the perspectives

The most basic component of the chosen feature set is the perspectives feature, which effectively separates gallery browser, the central part of the application's GUI, into three distinct views. This is illustrated in Figure 4.6, which gives an overview of the first two levels of the GUI hierarchy. Additionally, there is the root layout to be considered, which defines the window frame of the application and the top menu bar which hosts e.g. the File-menu, Settings menus and the application help menu.

Figure 4.10 presents a small section of the lower hierarchy of the GUI. Note that all the subtopics presented in Figure 4.6 contain definitions of their subcomponents down to the level presented in Figure 4.10.

The advantage of using a mindmapping tool for the development of the GUI is, that progress visualisation is possible by adding a pie-style progress indicator to each element and periodically updating them as progress is made.

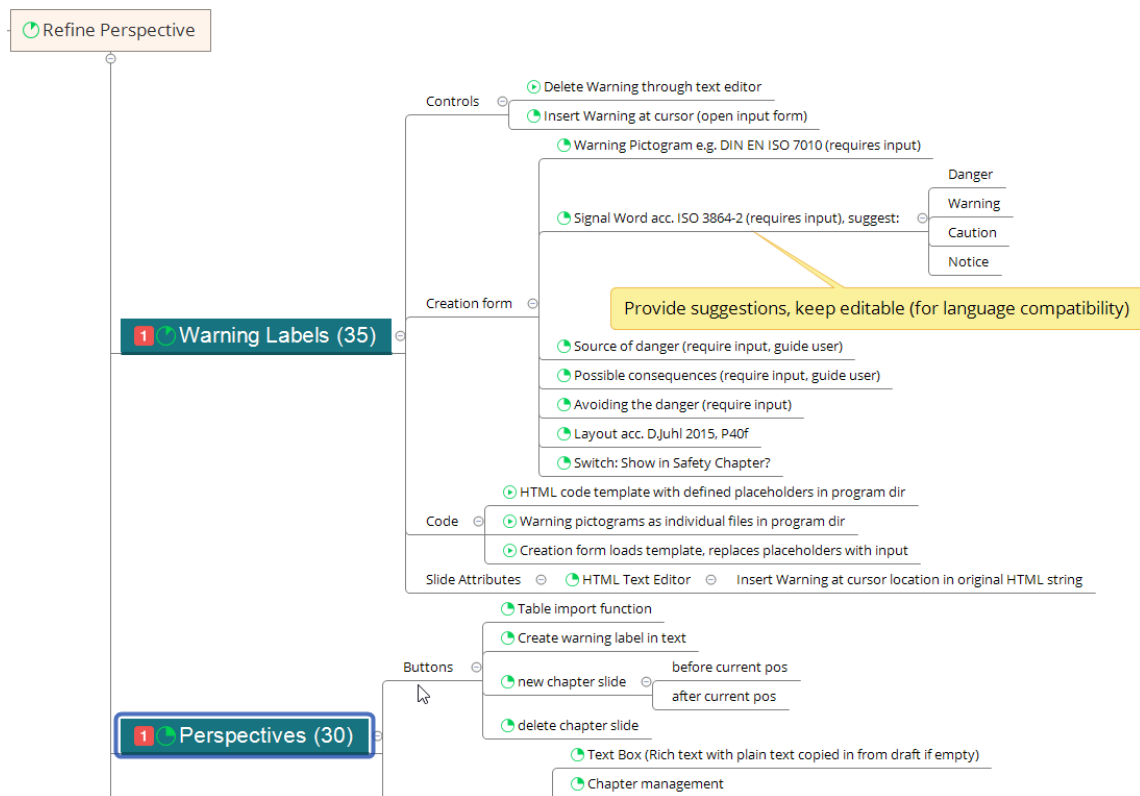


Figure 4.7: A small example subset of the GUI specification

To model the GUI, a mockup was created of each principal mode (Draft, Refine, View).

The layout of the gallery browser in draft mode can be found in B-1 - GUI Mockups for Draft Mode. The text on the individual elements is secondary in this case, the main purpose of such an exercise is to get an impression of whether a GUI is overly crowded, and if the features are arranged in an accessible way.

The layout of the gallery browser in refine mode can be found in B-2 - GUI Mockups for Refine Mode, and for view mode in B-3 - GUI Mockups for View Mode.

4.4.2 Designing the Gallery

Visualizing the application logic before implementation is a crucial step towards building a working application. Not every little detail requires modeling prior to implementation, since modeling all trivial things would slow down the development process considerably without providing measurable benefit. However, modeling procedures where timing and order are

critical, such as saving or loading a project with all its components, helps to reduce the time needed for debugging considerably.

The gallery is an entity which contains one or more gallery items (called slides) in an array list, and has a defined state determined by the currently active slide. By passing an index to the gallery, the gallery item from that index will become the new current item. The current item can be extracted from the gallery by the application, and its information (image, text, audio...) loaded into the gallery browser to be displayed and manipulated by the user.

Gallery slides can represent instruction-steps, but they can also represent e.g. a title slide, which has different requirements than a slide intended for instructions.

Generally, all slide types have the same basic functional requirements (all slide types have an annotation object attached, for example), but need their own extra features.

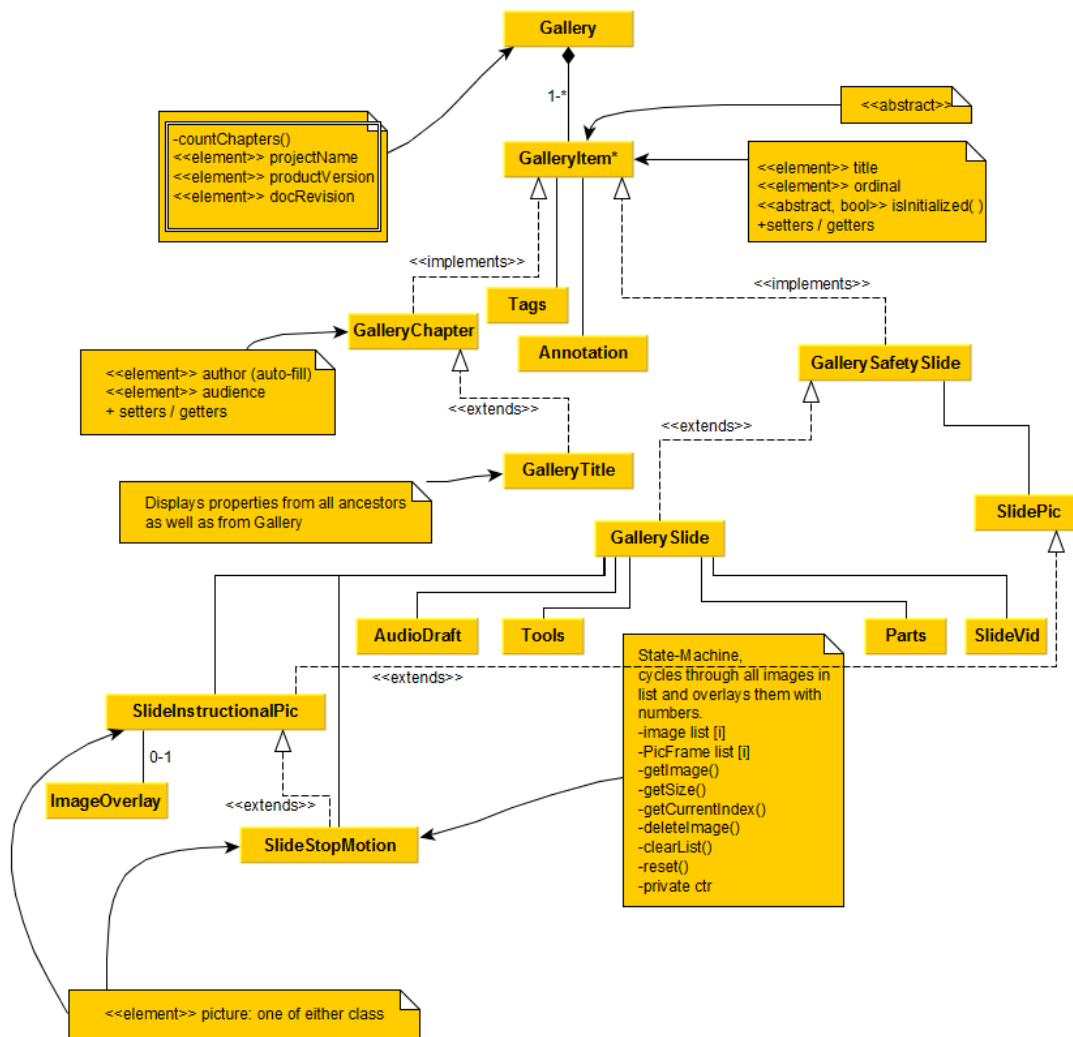


Figure 4.8: The inheritance structure of the gallery items.

Figure 4.8 presents the inheritance tree of all gallery slides, and shows the features implemented by each slide type. All the way on top is the gallery which owns one or more gallery items. All heirs of the gallery item class are accepted by the gallery, since inheritance is a strictly hierarchical concept, and a class derived from another will be accepted both in places where its specific class is required, and in places where the superclass (the ancestor) is required. The current item from the gallery is given to the gallery browser, which distributes the contained information (image, text, tags, etc.) among its GUI elements. When information is edited in the gallery browser, e.g. if the text in the annotation box is altered, the gallery browser copies these changes to the current slide immediately. When a navigation button is pressed or a jump via the slide list initiated, all the information in the browser is passed to the current slide once more before exchanging the data in the GUI with the data from the new current slide. Since Java objects are passed by reference, the current slide does not need to be copied back into the slide list. Rather, the item stays referenced by the slide list when passed to the gallery browser, and all changes made to the current slide in the gallery browser affect the same item as in the slide list.

This is an advantage in the case of the documentation software, but has to be taken into consideration at times as the notion of passing a variable by copying it tends to be taken for granted. Not being aware of the implications of a pass-by-reference system can create errors in the program which are very difficult to locate.

4.4.3 Capturing Images with the Camera

When a picture is being taken, the camera is first activated to provide a steady stream of images (like a video stream). The user then has time to position the subject of the documentation or the camera according to his needs, and then take a snapshot via clicking on a button, which stops the camera stream, retains the last image and passes it to the current slide.

The camera stream is generated by taking a snapshot every 33ms (at high resolutions the interval is longer due to the limitations of CPU and camera) and then passing it on to the gallery browsers image view. If this was done by the main application thread, starting the camera would occupy the thread and the GUI would stop responding.

The solution to this, is to start a separate thread which uses the camera to take the pictures periodically and load them into the image view.

The application thread is free to manage asynchronous events from the GUI, such as capturing the click of the snapshot-button while the camera thread is running.

The camera thread is then interrupted by the application thread, and after running its last cycle it can be disposed of.

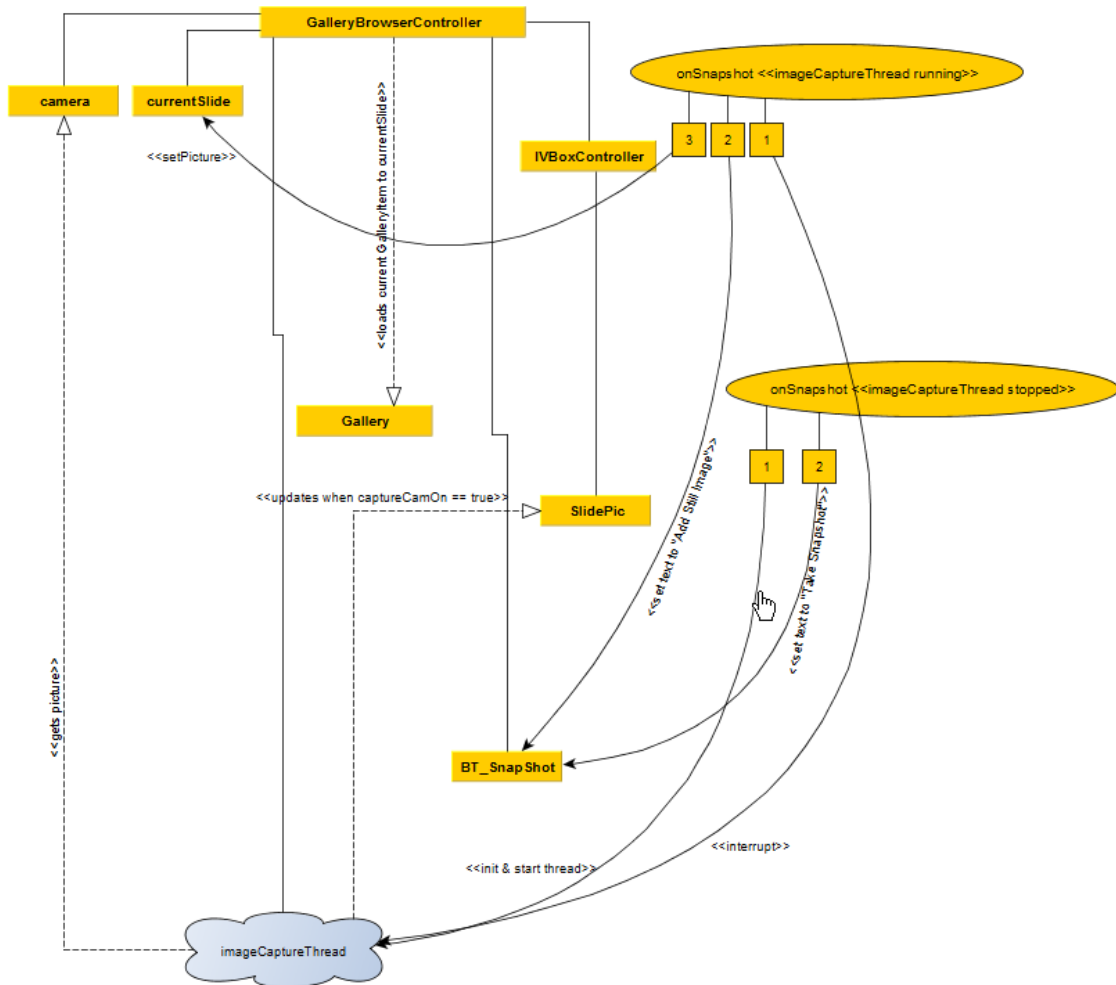


Figure 4.9: The interaction between main application thread (represented by the gallery browser controller) and camera thread.

4.4.4 Designing the Stop Motion Pictures

For the feature number 22 (labeled "Stop Motion Capture" in Fig. ??), the idea to employ the common .gif format was abandoned early, as the JavaFX specification explicitly supports the .gif format, but only as still image. The solution was to use regular single images for each frame, contain them in a class which contains a list and a cyclic counter (meaning the counter is reset when it's value reaches the size of the list), as well as a method to get the next image in the list, which returns the image at the current counter-position and increases the counter by one, so the next retrieval attempt will yield the next image.

This image container class (SlideStopMotion in Fig. 4.8 is an heir of the class SlidePic, which is the default container for images owned by a slide. While the base class only contains one image at a time, SlideStopMotion can contain any number of images. To facilitate replay, the image view element of the gallery browser checks the type of the picture everytime the gallery browser passes it a new one. If the new picture is a simple SlidePic, the image view controller will simply set the static picture in its viewport. If the new picture is a SlideStopMotion, the image view controller will start a new thread which gets the next image from the SlideStopMotion object in fixed intervals. The length of the intervals can be set within the application settings.

Through this approach, it is also possible to record a stop motion image and open it for editing later, e.g. replace individual frames or add more frames to the series.

4.4.5 Saving the Gallery

To save a large number of items which may each contain different constellations of data, such as the gallery which may contain slides with text only, with image and text, or with stop-motion images attached, it makes sense to confine the save- and load logic to the slides themselves. The base class of all items in the gallery, the class GalleryItem, contains the logic to save the annotation text as well as the tags. All classes that inherit from GalleryItem simply override GalleryItem's save method, implement the code to save their respective features to the file system, and then explicitly call the superclass' save-method.

This mechanism enables a distant heir of GalleryItem to focus only on saving its own unique attributes, call its ancestor's class' save method afterwards, and be sure that all other information in the slide is saved.

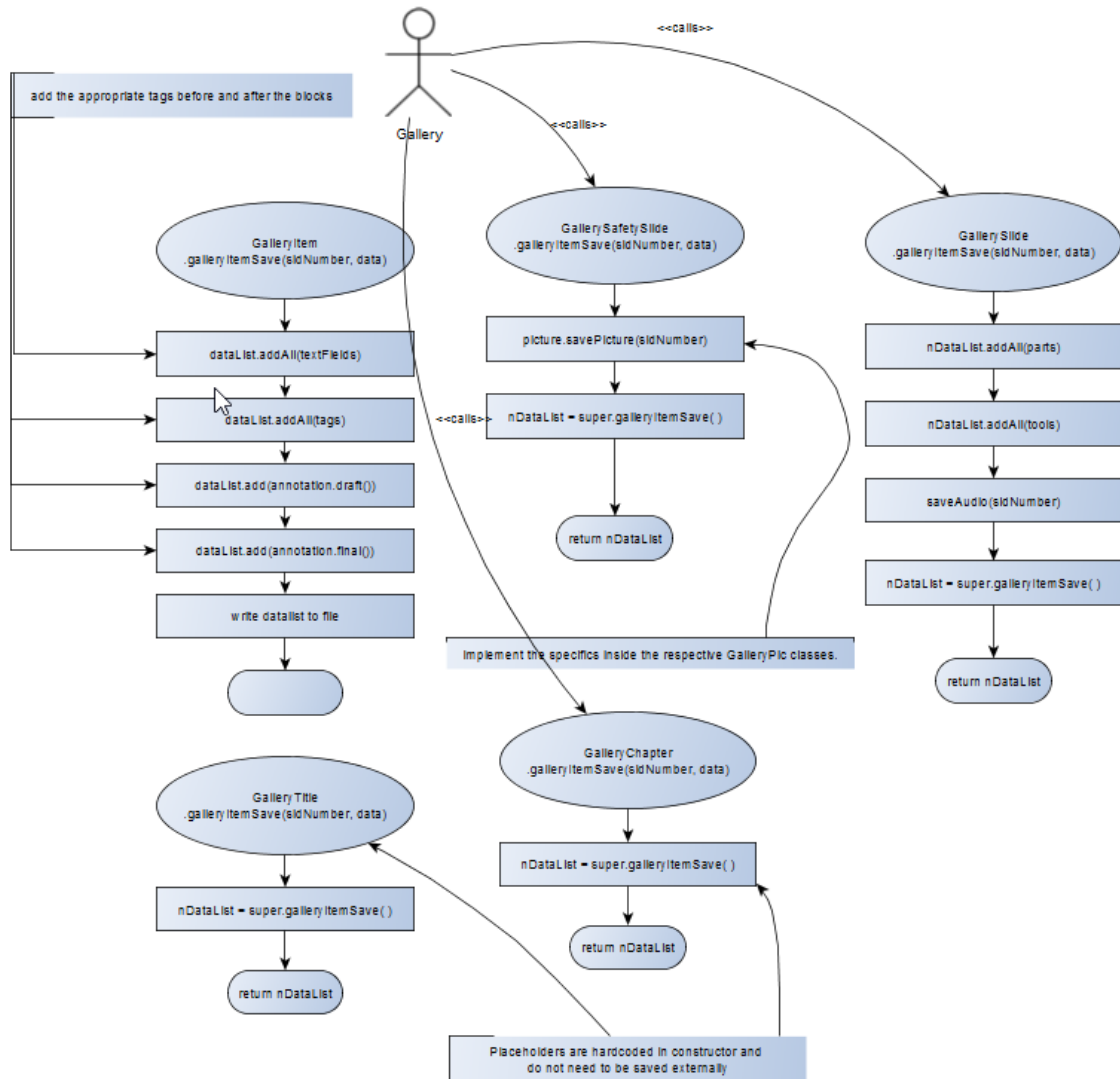


Figure 4.10: The hierarchy of gallery items during the saving procedure.

4.4.6 Making the Project- and Application Settings Available to all Classes

One challenge arising during the development of a complex software is the need to have certain operational constants and variables which are accessible to many different classes in many different contexts.

For example, by giving the logic for saving the gallery's data to the GalleryItem class, this class must have a way of determining where the project directory is located on the filesystem, to be able to specify the path where e.g. the image should be saved.

This is achieved by using the Singleton pattern. A singleton is a class, which can only be instantiated once within a given system. Instead of every object creating its own settings object, which would need to be updated manually, every object needing access to the settings gets a reference to the same settings object.

If one class manipulates the singleton, all others can instantly see the change.

E.g. the path of a project may be set through a directory chooser mechanism in the root-layout, but the GalleryItem will save its data at the specified location the next time the project is saved.

5 Software Development

5.1 Language and Tools

5.1.1 The Java Programming Language

One of the goals of the project was from the start, to keep the software as portable to different platforms as possible. The Java language offers the concept of Write Once, Run Anywhere, which means the source code does not have to be compiled for specific operating systems or hardware platforms. Instead, it is compiled once into a specific format (Java Bytecode) and then executed inside the Java Runtime Environment. Therefore, a java program can be run on any platform for which a JRE exists, which range from all common desktop OSs to mobile devices, embedded devices, and large non-desktop computing systems such as servers (Oracle (2016b)). This makes it stand out among other compiled languages like C and its derivatives, which need to be compiled for specific platforms. The basic tools required to make and run java applications (Compiler, JRE) are free for personal and commercial use, and are supported by free IDEs like Eclipse or Netbeans. As opposed to most programming languages, a lot of specific functionality (e.g. image manipulation, GUI development) is directly integrated into the java standard and supported by Oracle, which ensures portability retention across all supported systems. For functionality not provided with the official java release, there are numerous third party libraries (free as well as commercial ones) for many specific applications available online. Furthermore, the development of the java platform is ongoing and expanding, making the language a future-proof choice for applications which will be undergoing periodical improvement over a potentially long product life-cycle. Since the intention of this work is to create a first published version of software which will be useful in a subset of all technical documentation situations, but also to act as the basis for further development after the conclusion of this thesis, the Java language was selected because of the unique qualities described above.

5.1.2 GUI Framework

There are several GUI-building frameworks available for the java language. Oracle currently provides official support for two: Swing and the newer JavaFX.

Swing GUIs are coded in Java (for which the NetBeans IDE provides a WYSIWYG editor), which can be manually edited by the developer.

In JavaFX, the software developer can choose at any point to implement a GUI element completely or partially in java code, or create it using the SceneBuilder 2.0 tool supplied by Oracle, which provides a drag&drop interface for creating a GUI description in the FXML format. The JavaFX GUI is organized in the JavaFX Scene Graph (an excellent overview of the JavaFX architecture can be found at Oracle (2016c)), which is implemented as a tree structure consisting of nodes. The root node can be seen as the basic canvas on which the GUI is rendered, and its children are nodes which may respectively own various arrangements of nodes (e.g. controls such as buttons) which are displayed when appropriate. Each node in the JavaFX Scene Graph can have zero or more children and must have exactly one parent. The nodes in the JavaFX Scene Graph can be linked to the application logic through event-handler methods, and nodes requiring dynamic configuration at runtime (e.g. text boxes) can be created in code and linked to the respective GUI element. For example, most software buttons only react to a simple left-click of the mouse. In JavaFX, the button does not need to be created in java code by the developer - instead, the button is created in FXML and its event attribute (in this case the event of a mouse click) linked to a specific method (called an event handler method).

The FXML description of a Graphical User Interface is loaded at compile time, and only then turned into java code. This decoupling of the Graphical User Interface and the underlying application logic has the advantage of keeping the code leaner and easier to maintain, because only elements which get manipulated at runtime have to be coded. For those, only the changes have to be written in java code, while their initial state (size, colour, enabled / disabled) can still be declared in FXML.

Because both JavaFX and Swing are supported actively by Oracle, the newer JavaFX has been designed for compatibility with Swing. This is important in the case of a third party library offering a specialized functionality only being available for one of the toolkits, but needed in an application which has been coded in the other. An image processing library which uses the Swing Image class can still be used in the context of an application written in JavaFX. Whenever the image variable is passed from one toolkit to the other, it can simply be converted to the respective format using utility functions provided as part of the Java API. JavaFX is portable to all potentially relevant platforms for this project (Windows/Linux PC, Mac, ARM), and since it is newer, likely to be supported further into the future than

Swing. The modernisations regarding development concepts (separation of GUI and application logic) and its acceptance by industrial development groups as a stable solution lead to the conclusion that JavaFX will be used as the main toolkit during this project.

The reason why third party GUI toolkits (e.g. GTK+) were eliminated from serious consideration for this project is, that they are not as inherently portable as the officially Oracle-supported toolkits. Also, since both JavaFX and Swing are built into current java distributions, their integration does not have to be handled by the developer, decreasing the likelihood of time-costly resource-linking problems during the development phase, the resolution of which brings no benefit other than the ability to proceed as planned behind schedule.

5.1.3 Setting up Eclipse JDT with e(fx)clipse Plugin

Oracle is officially supporting two well-established free Integrated Development Environments: NetBeans and Eclipse. Eclipse has been chosen because of good operational experience from previous projects, although both IDEs are equally suitable for Java development.

This section will serve as the setup documentation for the Eclipse IDE and the plugins necessary to develop a JavaFX application.

The process of setting up the Eclipse IDE is well documented on the website of The Eclipse Foundation (2016a) Different pre-packed versions of Eclipse can be downloaded for e.g. different languages or tasks, although plugins for other languages can always be added and removed later. For developing a Java application, The Eclipse Foundation (2016b) is the recommended version to download.

After setting up the IDE, the JavaFX plugin named e(fx)clipse has to be installed. This can be downloaded and installed using the plugin wizard integrated into eclipse. Detailed instructions of the installation process of the plugin is made available at BestSolution.at EDV Systemhaus GmbH (2016).

5.1.4 SceneBuilder 2.0

SceneBuilder 2.0 is a WYSIWYG Graphical User Interface layout tool for use in conjunction with the JavaFX toolkit. By using SceneBuilder 2.0 the developer avoids having to write the whole GUI in Java code or FXML manually, thereby reducing the amount of code potentially requiring maintenance or debugging. Modifying an existing design also becomes more intuitive when the items can be placed and resized by mouse, as opposed to coding their arrangement manually and having to visualise the layout by studying the code. The application is distributed separately from JavaFX and e(fx)clipse, but it is possible to start it from

inside the Eclipse GUI by right-clicking on an FXML file and selecting the option [Open in SceneBuilder 2.0].

The instructions on how to install SceneBuilder 2.0 can be found at Oracle (2016a)

5.2 Software Libraries

5.2.1 Criteria

Any API considered for use in this project has to conform to the following requirements:

- It must be freely accessible, preferably open-source
- It must be published under a license which does not restrict commercial use
- It must work without an internet connection if it is not an optional feature of the application. Non-Optional are any features needed for picture taking, creating, saving and loading a documentation project.
- It must be under active development or supported as a final product
- It shall not affect the portability of the application to the intended platforms: Windows, Linux, Mac, ARM

5.2.2 Webcam Capture API

The Java programming language does not offer a native API for connecting to and using webcams, therefore a third party library had to be found for this purpose. The specific selection criteria for this API are:

- It must support selection of a specific USB-Webcam if several are connected
- It must support detecting listing available webcams
- It must support the capture of still images, or video from which still images can be extracted
- It must allow the user to change the camera's resolution

Table 5.1: Comparison of Webcam Capture APIs

API Name	Webcam Selection	Webcam Listing	Capture of Still Images	Allows setting Resolution	License
JavaCV V1.1	Yes	No	Yes	Yes	Apache License 2.0 Or GNU-GPL Version 2 Bytedeco (2015)
OpenCV V3.10	Yes	No	Yes	Yes	3-clause BSD License OpenCV (2016)
Webcam Capture API V0.3.11	Yes	Yes	Yes	Yes	The MIT License Firyn (2015)

The three APIs which were found to be conforming to the the criteria in 5.2.1 were compared regarding the criteria set in 5.2.2. As presented in Table 5.1 only one of the APIs provided the functionality to scan for available cameras and present them in a list. The API by Firyn (2016) does not require additional dependencies and supports all functionality required during this project. For integration into the documentation software, the Webcam Capture API V0.3.11 will be implemented behind a generic interface to allow easy switching to another API at a later point, if a more suitable one becomes available.

5.2.3 Apache Commons Library

Java does not provide functionality to delete whole directory trees including all contained files, which is an important feature for saving a new version of a project in an existing location. The necessary functions are available in compliance with the criteria set under 5.2.1 from Apache Software Foundation (2016).

5.3 Summary of the implementation Process

5.3.1 Successfully implemented Features

The perspective-oriented user interface was successfully implemented. Most features such as Stop-Motion capture, title and chapter slides, the tagging system and table import as well as the warning label generator were implemented without any significant problems.

The audio draft feature proved more difficult than initially expected, because the audio file could only be created at recording time, and from then on has to be managed by its owning slide in a way it does not get lost when e.g. the slide is changed to a different index. Furthermore, the member-method of the JavaFX Media class, which was supposed to get the duration of the recorded audio clip never returned a value until the point when the audio clip was overwritten. If a clip was recorded for 3s, and the recording stopped, the get-duration method was supposed to update a label in the audio player controller. Instead, the displayed value remained "0" until the recorder was started to record another audio draft. Then, the label was suddenly updated with the 3second duration from the just-discarded clip. Those issues resulted in an implementation time for this feature which was several times of its initially projected 6h.

During the implementation of the PDF export feature, a problem was discovered: The Apache Foundation's PDFBox needed plain text as input, but the text from the rich text editor in the refine-tab is HTML-based. A possible solution may have been to pass the html text through two separate conversion APIs, but success was deemed unlikely since the HTML code may suffer from impurities caused by the import of external tables and warnings.

Since a printable form was determined necessary, the HTML variant was chosen. This proved much easier to implement, and through strategic use of pagebreaks is able to produce a printable version of the manual (See C - The Software Manual for the manual created for the software, with the software).

5.3.2 Discarded Features

Some features could not be implemented due to a shortage of time. The time shortage was not only caused by difficulties during the implementation, but also during the deployment of the software package.

The Video Capture function was not implemented beyond the GUI elements (which have been removed from the now final version of the software). The main reason for this is its conceptual similarity with the audio draft. The audio draft is, in principle, a very simple feature. Yet when it came to implementing it, features like exporting the entire project to HTML took

less time than getting the audio draft into usable condition. Since the time originally allotted for the implementation phase had already been exceeded by 3 weeks, due to difficulties with the audio draft, thread-synchronisation and the deployment of the application, it was decided to drop the feature from the V1.0 of the application, and keep it for consideration for the next release.

The Safety Chapter feature became obsolete during the development. The reason was, it was found that the functionality could easily be covered by the existing features, namely by using the regular chapter slide in combination with regular gallery slides which are filled with the relevant warnings.

The automatically coloured frames for images of warning-equipped slides had to be dropped due to a shortage of time.

The saving hooks were determined to be a relatively unimportant feature and therefore dropped in favour of an extra function: A button to open the current slide's image in the system's default editor for e.g. adding highlights.

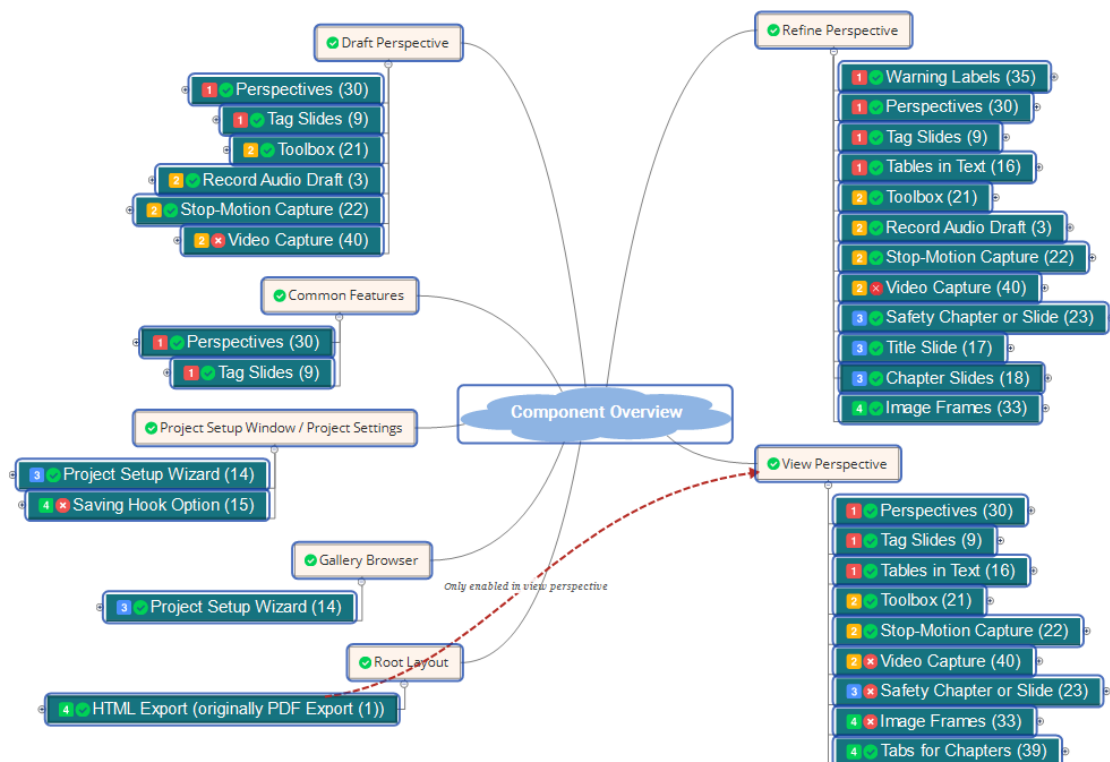


Figure 5.1: Overview of the implemented and the discarded features

The Figure 5.1 shows the result of the implementation phase: The Features with a green check have been implemented, the ones with a red X have been dropped.

6 Testing and Evaluation

6.1 Objective & Outline

The objective of the testing is to determine stability of the software's features as well as the user-friendliness of the interface. Originally the software was meant to be tested against three different testcases. Due to a severely extended implementation and commissioning phase, there was no time to test the software on external subjects. However, the software has been successfully used to create a user-manual for itself, which was subsequently embedded in the application and can be loaded via the help-function.

During the course of the manual's creation, all the software's features were tested. The self-documentation shall therefore serve as the basis of the evaluation.

6.2 Summary of the Test

The application performed well. The interface proved usable in its current configuration, and the draft as well as refine-mode features all proved to have their use. Through managing the information structure in the background, the documenter can focus on collecting the information and, after getting used to the UI, quickly file it in the right location.

Whether the produced manual is of high quality remains in the documenters hands, since functions which actively guide the user are not extensive.

Guidelines on good writing are available in the form of tooltips, which appear when the mouse cursor hovers over a control element. Many controls in the gallery browser have been fitted with tooltips, but it remains up to the user to read them.

For a documenter who knows the rules for making good manuals, the software provides the necessary means and streamlines the process considerably.

One feature stood out in particular during the test: The function to open a slide's image in the operating system's default editor was an idea which was added after the principal feature selection. The reason this extra feature was added despite being short on time was, that this

feature actually only required one GUI-element and approximately 10 lines of straightforward code, and did not require lengthy debugging. Opening an image externally, editing it, and re-inserting it into the documentation project via the clipboard quickly became a routine task, and proved to be quite ergonomical for the reason that the complete cycle of opening, editing and reinserting the image does not require handling any file-chooser menus or concerning oneself with the images location in the local filesystem at all.

The Stop-Motion image function proved useful breaking longer sequences of actions into several frames. From the text, the reference to a highlight in a frame of the stop motion image can then be made by stating the frame number and the highlight in question. The decision to implement this selfmade solution after determining that using the .GIF format was not natively available in JavaFX also had the added benefit of keeping the image series easily editable.

The capability of the JavaFX HTML-Editor which is used in the refine tab, proved very adequate at formatting the instructions with e.g. different levels of headings, bullet points or enumerations. One small caveat is, that the numbers used for enumerated lists cannot be printed in bold while keeping the rest plain. This would have increased the clarity of the enumerated lists.

Comparing to the manual used as an example for the state of the art (KUKA Roboter GmbH (2015)), the manuals produced by this software can be considered on a similar level when it comes to clarity and structure, providing the documenting person can provide those. Two distinct advantages of the manuals produced by the developed software are:

- Warnings are always structured to the point where the text is, at the very least, separated into the underlying topics and cannot be one long, badly readable paragraph. Even bullet lists and enumerations can be inserted into the warnings and notices post-generation.
- Tables cannot be inserted without having an identifier as well as a caption text.

The total length of the Application's manual is 47 pages, and it makes use of separation of topics by using chapter slides. Due to the different colour-scheme used for chapter slides in the index list, the feature of generating extra quick-access links for each chapter becomes mostly obsolete.

The Title slide is useful to state the purpose of the manual, and to present a logo or a picture of the product.

The tagging system works well, and can be used to find information quickly after learning its basic principles.

Exporting a manual to printable HTML works, condition is that the relevant section of the manual was read and its advice adhered to.

The audio draft function works, but due to a lack of situations during the test, where the documenter needed to keep his hands free, it was only used for the sake of testing it.

Furthermore, the automatic resizing of the GUI elements depending on a slide's content makes the experience more lean, since the part of the manual which contains the information is always filling the screen.

Overall, the result of the test was positive.

Refer to C - The Software Manual for the print version of the application's manual.

7 Conclusion

7.1 Personal Development

During the course of this thesis, a great many things were learned.

For one: Making a software from a prototype into something akin of a product, which can be given to others without having to worry about them having a bad user experience, takes a lot of fine-tuning and more time than initially anticipated.

Second: The amount of time needed for adjusting the frameworks of a project is easily underestimated. After debugging, the deployment of the software so it could be run from outside of the Eclipse IDE was planned to take place within an hour. It took about two days of trying until the resources were found both when starting the application during the development in the IDE, and after it was deployed in .jar and .exe format.

The planned tests for mechanical assembly, commissioning and maintenance could not be conducted and included in this thesis due to previously mentioned shortage of time as a result of difficulties during the implementation and deployment phase. While this may affect the quality and thoroughness of this thesis from the reader's point of view, I stand by the decision to allocate the necessary time to turn the software into a usable product to the detriment of the testing and evaluation phase, which was changed from three planned cases of documentation to one single case that was not previously intended to be used as an example.

The reason why the priorities were picked this way is, that a software which does not work will not be useful to anyone. After committing the better part of a year to learning the basics and some more advanced functions of the Java programming language, designing a basic prototype as a feasibility study on which the software developed during the thesis could be based, and committing fully to the methodical development of a software product after the thesis was registered officially, the notion of publishing a well-documented but unfinished or unusable end-result was not considered an acceptable conclusion of my bachelor's study.

My overall personal verdict, however, is a positive one. The application works, and it is not just a reasonably stable prototype but a working system which can be used, even if some rough edges may still be present. During the creation of the software's manual, which was

created by using the software itself, several bugs were discovered which were subsequently fixed and their function verified. The usability of the applications primary functions were confirmed during the creation of its own user manual.

7.2 Plans for the Future

The software, which has been named **ManuMaker** owing to the fact that it is meant for, colloquially speaking, *making manuals*, will be published under the GNU GPL license.

The intention is to build a user base by introducing the software on internet platforms frequented by the target user groups, and open a bugtracking service for the project to give the users the chance to log issues they discover. Using feedback from the users, the current version will be refined further and problems fixed, as time is available. Participation in the continued development of ManuMaker by the users will be encouraged though publication of the project on a source hosting platform.

Furthermore, the following features are planned for being added to the current version:

- The file- and directory-paths which are used internally are currently based on Windows standard. The first order of business will be to make the paths system-independent or at least deploy versions of the software which work under Linux and Mac operating systems
- A new mode will be added to the GUI, which will provide optimized controls for touch-screen devices, to enable the full-featured use of the application on e.g. tablet computers. This will enable the use of a camera-equipped tablet as a documentation tool for larger subjects, e.g. industrial machinery components which cannot be extracted and placed on a desktop
- The option to automatically list the required parts and tools on dedicated slides at the beginning of a documentation
- Making the export function more versatile and adding options such as page-orientation, and placing two or more slides on one page (for documentation with short instructions)
- The GUI-behaviour and possibly the arrangement of some components will be optimized based on user-feedback and my own experience gathered from using the software
- The handling of the audio draft will be streamlined

- Compiling the gallery in a second HTML-based form which makes use of links to navigate from one slide to the next will be considered. This is meant to be generated everytime the main project is saved, to always have an up-to-date portable version of the documentation on hand, which can be shared and viewed in any web-browser
- If there are users who require the application output to be localized, a language-setting option will be considered if contributors can be found for the translations

Meanwhile, the next version (V2.0) will be designed and developed, which will focus on making the user-experience more smooth and adding some of the features which could not be added to the current version due to time constraints, such as perhaps voice-control for basic functions (navigation, camera operation).

The speech-to-text feature is also a major point of interest, but for the moment I intend to keep the software offline-only, and currently available and free to use STT engines need the internet to function properly.

Finally, my primary motivation for developing this software as a bachelor's thesis project is, to conclude my studies with a useful contribution to at least a subset of our society. My hope is, that the software developed during this thesis will improve life for the people who are developing and publishing freely accessible tools and products for the benefit of others, by making the process of documenting the functions of a product less bothersome and maybe even enjoyable.

Bibliography

- [Apache Software Foundation 2016] APACHE SOFTWARE FOUNDATION: *FileUtils (Commons IO 2.4 API)*. 2016. – URL <https://commons.apache.org/proper/commons-io/javadocs/api-release/index.html?org/apache/commons/io/package-summary.html>. – Accessed on: 2016-06-03
- [BestSolution.at EDV Systemhaus GmbH 2016] BESTSOLUTION.AT EDV SYSTEMHAUS GMBH: *e(fx)clipse - JavaFX Tooling and Runtime for Eclipse and OSGi*. 2016. – URL <https://www.eclipse.org/efxclipse/install.html>. – Accessed on: 2016-06-30
- [Bytedeco 2015] BYTEDECO: *javacv LICENSE.txt at master bytedeco/javacv GitHub*. june 2015. – URL <https://github.com/bytedeco/javacv/blob/master/LICENSE.txt>. – Accessed on: 2016-06-03
- [Firyn 2015] FIRYN, Bartosz: *Update LICENSE.txt sarxos/webcam-capture@9827348 GitHub*. april 2015. – URL <https://github.com/sarxos/webcam-capture/commit/982734884c1c197fe2443eb366f538a739197334>. – Accessed on: 2016-06-03
- [Firyn 2016] FIRYN, Bartosz: *Webcam Capture in Java*. 2016. – URL <http://webcam-capture.sarxos.pl/>. – Accessed on: 2016-06-02
- [Hahn 1996] HAHN, Hans P.: *Technische Dokumentation leichtgemacht*. Muenchen : Hanser, 1996. – OCLC: 75763969. – ISBN 978-3-446-18178-6
- [Juhl 2015] JUHL, Dietrich: *Technische Dokumentation*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2015. – URL <http://link.springer.com/10.1007/978-3-662-46865-4>. – Accessed on: 2016-06-05. – ISBN 978-3-662-46864-7 978-3-662-46865-4
- [KUKA Roboter GmbH 2015] KUKA ROBOTER GMBH: *KR AGILUS sixx Montageanleitung*. march 2015
- [OpenCV 2016] OPENCV: *OpenCV license | OpenCV*. 2016. – URL <http://opencv.org/license.html>. – Accessed on: 2016-06-03

- [Oracle 2016a] ORACLE: *1 Installing JavaFX Scene Builder (Release 2)*. 2016. – URL https://docs.oracle.com/javase/8/scene-builder-2/installation-guide/jfxsb-installation_2_0.htm. – Accessed on: 2016-06-30
- [Oracle 2016b] ORACLE: *Oracle JDK 8 and JRE 8 Certified System Configurations*. 2016. – URL <http://www.oracle.com/technetwork/java/javase/certconfig-2095354.html#os>. – Accessed on: 2016-06-09
- [Oracle 2016c] ORACLE: *Understanding the JavaFX Architecture (Release 8)*. 2016. – URL <https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-architecture.htm>. – Accessed on: 2016-06-30
- [Piehl 2002] PIEHL, Jona: *Gebrauchsanleitungen optimal gestalten: ueber sinnvolle und verstaendliche Gestaltung*. Berlin : Springer, 2002 (X.media.press). – OCLC: 248490043. – ISBN 978-3-540-42619-6
- [The Eclipse Foundation 2016a] THE ECLIPSE FOUNDATION: *5 Steps to Install Eclipse*. 2016. – URL https://eclipse.org/downloads/index.php?show_instructions=TRUE. – Accessed on: 2016-06-30
- [The Eclipse Foundation 2016b] THE ECLIPSE FOUNDATION: *Eclipse IDE for Java Developers*. 2016. – URL <http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/mars2>. – Accessed on: 2016-06-30

Appendix

A

A-1 - Considered Features

feature comparison

Index	Feature Title	Description	Feature Dependency (List index numbers)	Potential conflicts / mutually exclusive with (Index)	Projected time Cost (in Hours) 99 = unknown, potentially very high	User Benefit (for documenter or reader, use highest) 1 = Low, 5 = High	Category
1	PDF export	Exporting documentation projects as .PDF	-	29; 34;	8	4	Readerformat
2	Voice Control	Creating voice command interpreter for some GUI buttons. Candidates: Snapshot / Abort, Next Slide, Previous Slide,	-	3; 4;	4	3	Documenter-Ergonomy
3	Record Audio draft	Possibility to record comments as Audio during assembly process and later transcribing (Manually, auto) to text. Add audio recording in record/draft mode. Allow several samples and naming them	-	2;	6	5	Documenter-Function

feature comparison

4	Voice to Text input	Create a voice command system to add text to the annotation box, possibly with special commands for enumeration / bullet points	-	2;	48	4	Documenter-Ergonomy
5	2-Mode Documentation	Save Assembly and Disassembly in one Project. Switch between modes at click of a button, reuse slides when possible, use special notation to insert reversible components. Requires Chapters to be self-contained, individually reversible	9;	12; 18; 31; 32; 36; 41;	16	3	Software

feature comparison

6	Graphical Feature Highlighting For still images	Enable documenter to add.: - Circles - Arrows (single and double-ended) - Boxes to images, preferably editable after save / load. Numbering shall be optional or forced on.	-	13; 22; 36;	24	4	Readerformat
7	Special purpose slides	Create different slide types, e.g.: table, text only, picture only, video...	-	-	16	1	Software
8	Save Project As One File	Save projects as .zip and load into temp directory to make them less prone to accidental manipulation by the user, and easier to e.g. send by email	-	-	8	2	Software
9	Tag slides	Add a tag-search bar under slide list, highlight matches, give suggestions for searchwords using autocomplete if possible. Show results in new tab behind main slide list	-	-	24	5	Reader-Ergonomy

feature comparison

10	Portable mode	include option for portable mode, where app settings are stored in local directory to enable startup from USB stick	-	-	2	1	Versatility
11	Watermark	Include option to add a company logo / watermark in every picture. Offer choice: Which corner Ideally only merged with image after explicit command	6;	-	4 / 10	2	Documenter-Function
12	Product variants	Add support for product variants sharing many common features by tagging variant-specific slides and offer variant selection via drop down menu in GalleryBrowser	9;	5; 36;	16	3	Readerformat

feature comparison

13	Linked images	Instead of snapshot, allow insertion of image via open-choose menu from e.g. Project CAD repo. If image gets updated the image in doc will get updated as well. Save Project local image in case source cannot be found, indicate link health.	-	6; 37;	8	1	Documenter-Ergonomy
14	Project Setup Wizard	After "New Project", a window opens where project specifics can be set (e.g. Project type, target group, etc. if applicable)	-	-	5	4	Documenter-Ergonomy
15	Saving Hook option	Add checkboxes to project settings: Do not allow saving unless slides have Titles, Text, etc.	-	-	2	4	Quality, Data Safety
16	Tables in Text (1)	Option to integrate Table (HTML, CVS) into text through import from file (unsynced) Force input of an identifier and a caption	-	-	8	5	Documenter-Function

feature comparison

17	Title Slide	Create a Title slide for each documentation, which presents the title, target group, etc. And must be filled in when a new project is created.	7;	-	4	4	Readerformat
18	Chapter title slides	implement slide organisation by chapters, which can have their own title and target group specification and so on. Collapsible list or chapter-slides marked in different colours	7 or 17; 23	5; 36;	2	4	Readerformat
19	Modular Slides	Offer function to fill in advance organizer, footnote, tools required, etc. In recording mode. Empty features for a slide are not shown in viewing mode.	21*;	28;	4	5	Documenter-Ergonomy
20	HTML Generation	Compile Project to HTML on save. Retain basic structure, use links for navigation, enable viewing by browser.	-	29; 34;	12	5	Readerformat

feature comparison

21	Toolbox	Add toolbox field on every slide, If possible, automatically print tools mentioned in bold, Compile to tool-requirements at beginning of the Manual	19 or 28; 34;	36;	8	5	Readerformat
22	GIF Slides	Support the integration of animated .GIFs and their creation using stop-motion capture.	-	6; 33;	5	3	Documenter-Function
23	Safety Chapter / Slide	Create a dedicated slide at the beginning of the manual with all safety related warnings before instructions start	17; 18; 35;	36;	4	5	Readerformat
24	BOM Import	Allow importing of BOM for a project, enumerate components throughout all slides in pictures and text according to BOM. (As in, only allow enumeration of numbers present on the BOM)	-	-	16	2	Documenter-Function

feature comparison

25	Text Analysis	Create a (parametric) automated text analysis tool, which checks the text acc. (2) and highlights the non-conformous sentences: - 13 Words - Max one comma - ":" before enumeration or bullet-points	-	-	12	4	Documenter-Ergonomy
26	Glossary	Create a glossary analogous to Toolbox function	-	-	4/ 8	4	Readerformat

feature comparison

27	Tooltip-Box	<p>Add tooltip box in UI which shows instructions on how to use the last item clicked / hovered over, or what to look out for during the documentation process, e.g.:</p> <ul style="list-style-type: none"> - Clear structure and concise information - Headlines repeat instructions - use command form and active verbs - using the glossary - Consistent terminology - numbering starts at 12o'clock - How to refer to numbered components - Numbered items in Image must be referenced in text 	-	-	8	4	Quality, Data Safety
28	Selective Features	<p>offer checkboxes beside every feature in record mode, to disable e.g. image, text, toolbox, etc. Allow user to create image-only or text-only pages.</p>	21;	19;	8	5	Documenter-Ergonomy

feature comparison

29	Layout retention	Save relative sizes of panels (text, image, toolbox...) for each slide inside the object to give documenter more control over the layout. Accept changes in view mode, reset on reload.	30;	1; 20; 34;	4	2	Readerformat
30	Perspectives	Offer perspectives: -Draft -Refinement -Viewing With appropriate features in the UI	3; 6; 35;	-	24	5	Documenter-Ergonomy
31	Navigation History	Offer quick links to last n locations left by clicking on a reference. Retain slide number of chapter and navigate to point of origin in last, n-d to last chapter Only active in viewing-mode	-	5; 36;	6	5	Reader-Ergonomy
32	User Annotations	Offer user-added text annotations outside of editing mode, display on demand	-	5; 36;	16	2	Readerformat

feature comparison

33	Image Frames	Add coloured frames to slides containing safety warnings: Different colours for Caution, Warning, Danger Automatically set colour acc. to highest warning level present in slide	6; 35;	22;	4	2	Readerformat
34	Tabbed Slides	Add tabs for Image, BOM, Tools when needed. Image exists as default.	21; 24; 26; 28; 40;	1; 20; 29;	4	3	Documenter-Function
35	SAFE Warning Labels (3)	Add Template for inserting well-structured Safety Warnings to the documentation text. Do not allow creation without adequate information	16-		6	5	Documenter-Ergonomy

feature comparison

36	Reverse Gallery	Provide the option to mark a documentation as "reversible", allowing the reader to choose between assembly and disassembly using the same slides. May work well on mostly-image projects such as lego documentation	-	5; 6; 12; 18; 21; 23; 31; 32; 40; 41;	8	2	Software
37	Feature Enumeration (4)	Provide means to add position numbers with optional lines to feature Automatically arrange from 12 o'clock	6;	13;	8	5	Readerformat
38	Manual to the Manual (5)	Solve through GUI-tooltips and a search function for slide tags	9; 27;	-	4	4	Software
39	Tabs for Chapters (6)	Add a quick-access tab to the side of the slide list for each chapter Maybe implement chapters als self-contained objects	-	-	4	4	Reader-Ergonomy

feature comparison

40	Video Capture (7)	Allow recording of a video clip in the place of an image. To retain printer compatibility, force the selection of one frame or the capture of one still image as alternative.	7; 34;	36;	6	5	Documenter-Function
41	References to other Chapters	Provide clickable reference links in annotation text which enable jumping to another slide Provide reference stack which can be backtracked Or a list of last n slides	18;	5; 36;	12	4	Reader-Ergonomy

Enumerated references for the table above (numbers are printed in bold, in 2nd or 3rd column from the left):

1. also found in: (Juhl, 2015, p. 253)
2. (Hahn, 1996, p. 40)
3. also found in: (Juhl, 2015, p. 37 ff)
4. also found in: Juhl (2015), Hahn (1996), Piehl (2002)
5. (Juhl, 2015, p. 199)
6. (Juhl, 2015, p. 209)
7. also found in: (Hahn, 1996, p. 55)

B

B-1 - GUI Mockups for Draft Mode

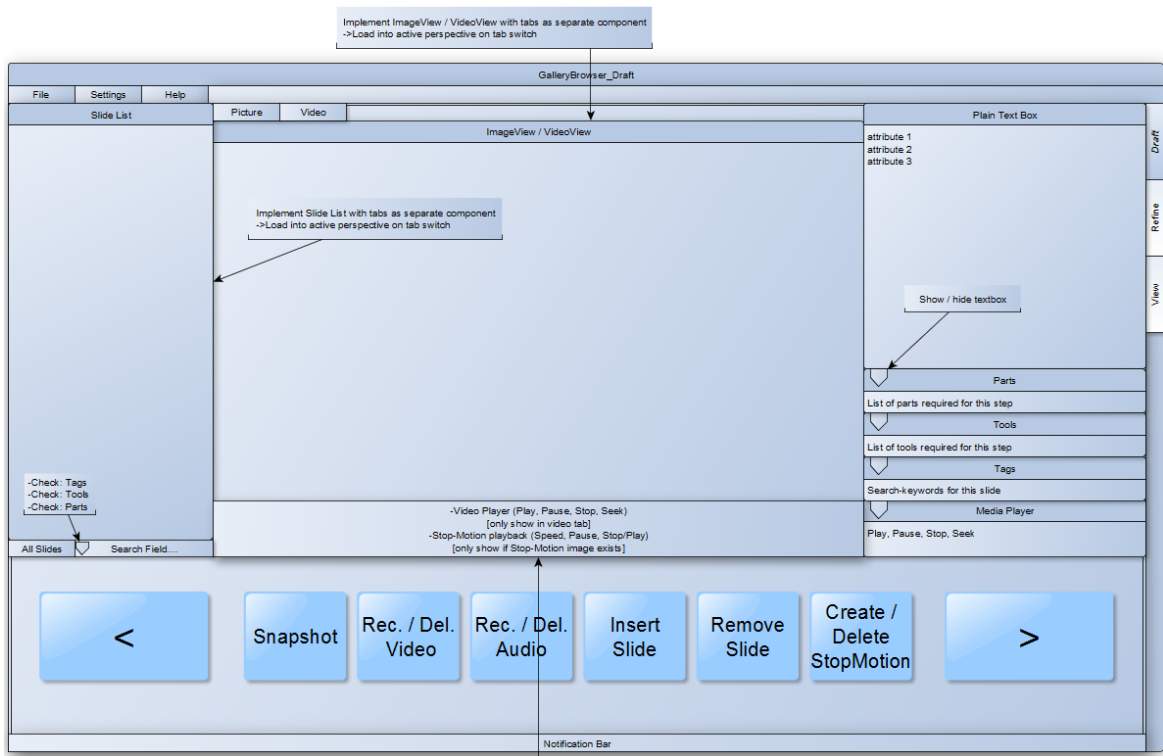


Figure 7.1: The initial design of the draft mode's layout

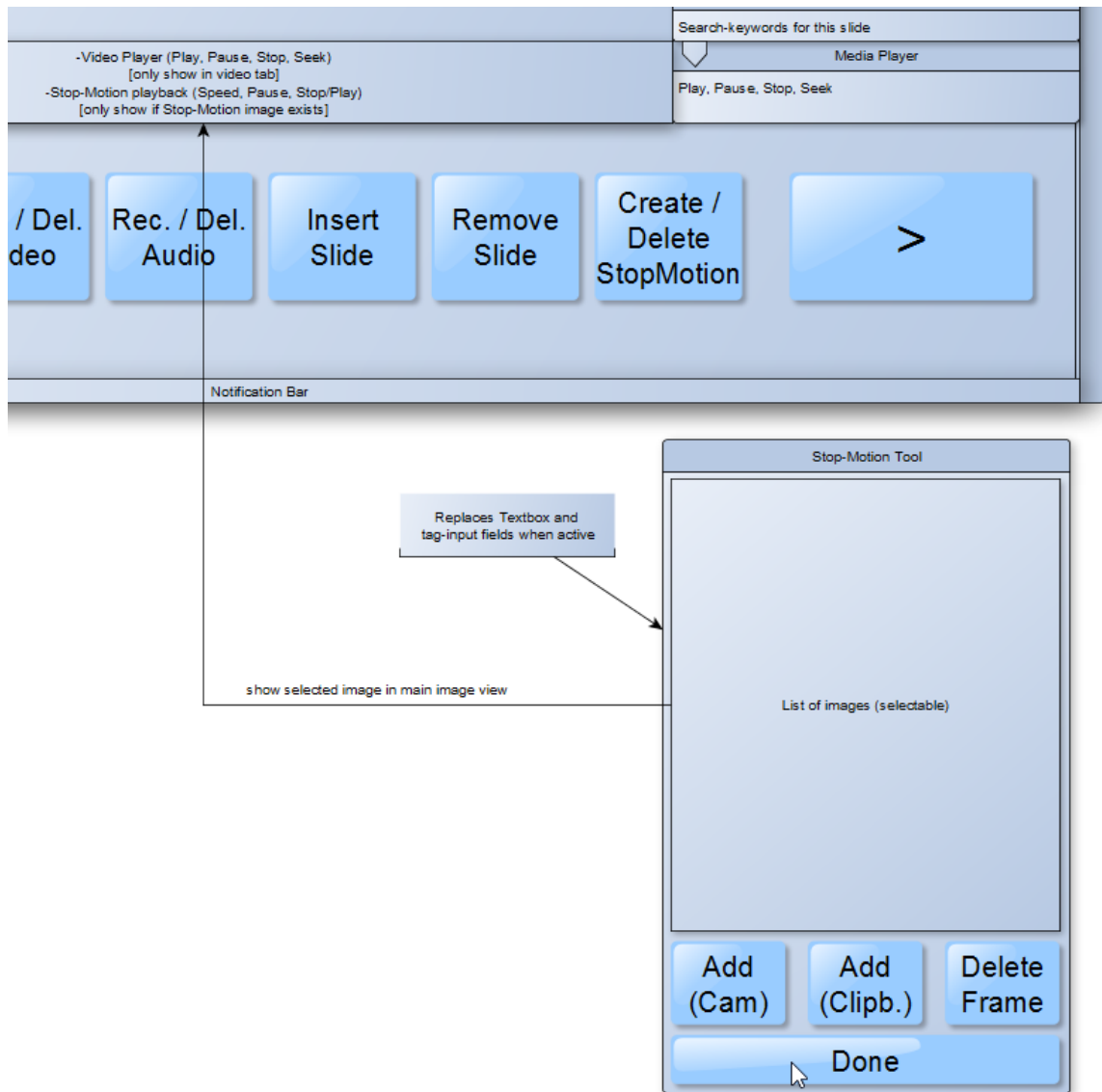


Figure 7.2: The sub controller operating within the draft mode

B-2 - GUI Mockups for Refine Mode

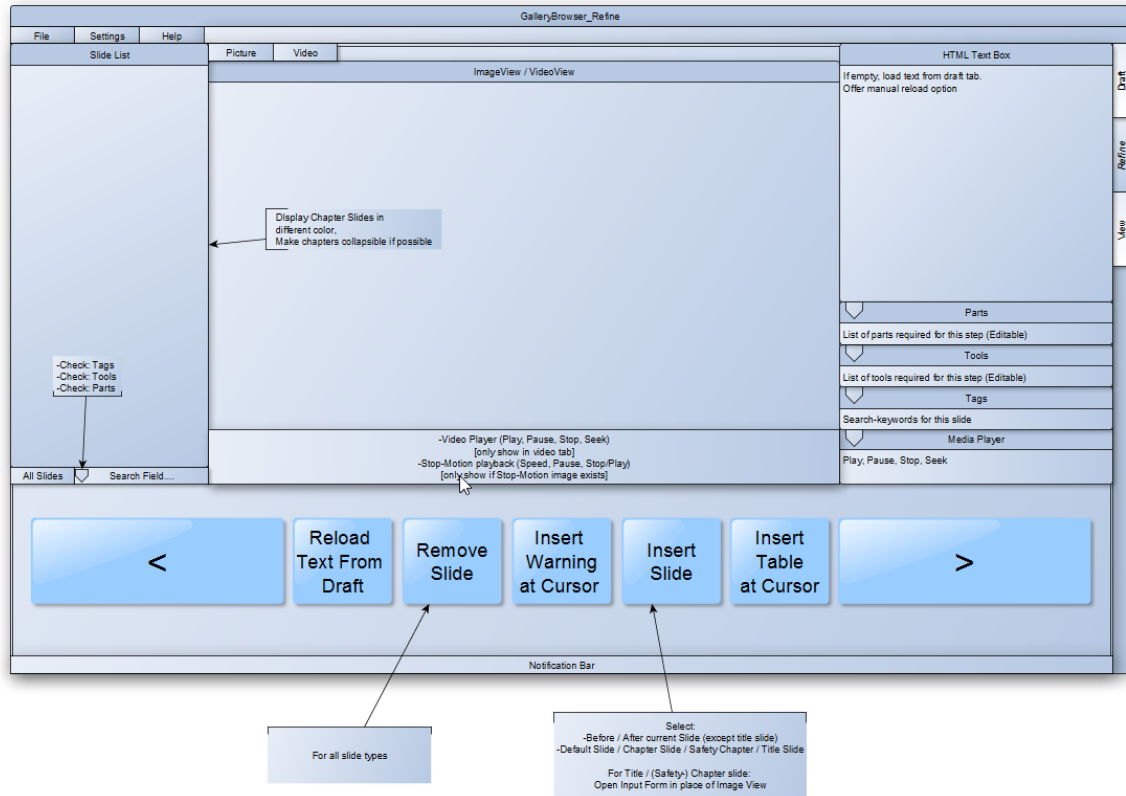


Figure 7.3: The initial design of the refine mode's layout

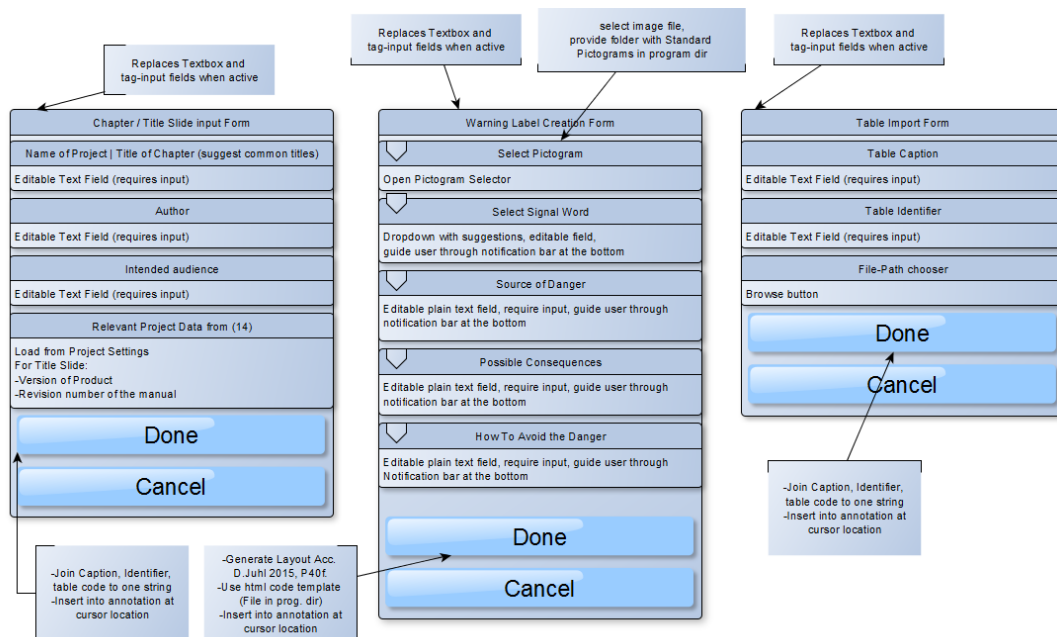


Figure 7.4: The sub controllers operating within the refine mode

B-3 - GUI Mockups for View Mode

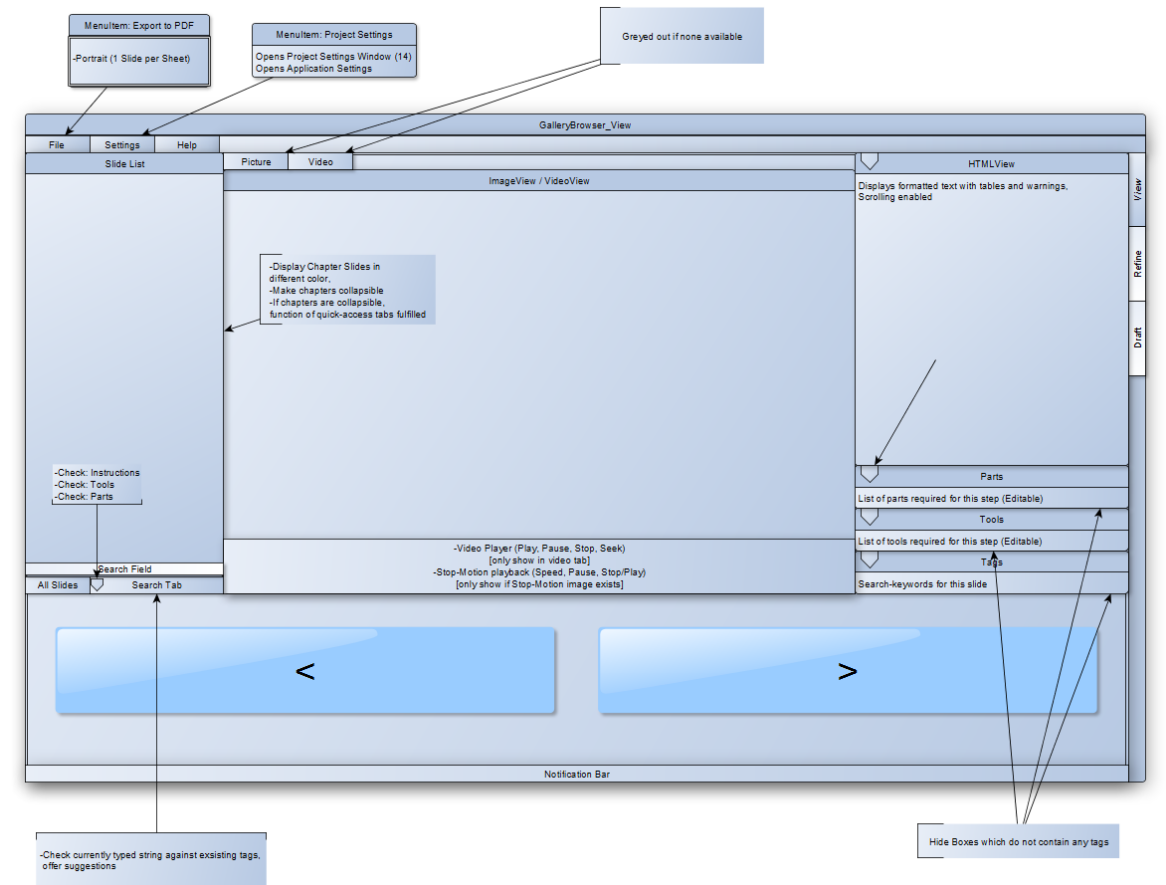


Figure 7.5: The initial design of the view mode's layout

C - The Software Manual

User Manual

for

ManuMaker

Version:

1.0

Intended Readers:

Users of the ManuMaker application

Development Context:

This application was developed as part of a bachelor's thesis at the University of Applied Sciences in Hamburg (HAW Hamburg)

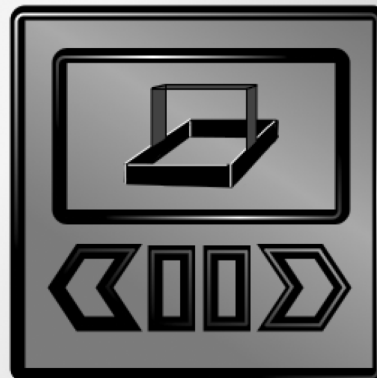
Author(s):

Daniel Lachmann

Manual Revision:

1.0.1

ManuMaker V1.0



HAW HAMBURG

Publisher:

HAW Hamburg

Berliner Tor 21
20099 Hamburg
Germany

Contact:

Email:
manumaker.dev@gmail.com

Chapter 1:

Introduction

**Intended
Readers:**

All users of the ManuMaker application

Author(s):

Daniel Lachmann

Publisher:

HAW Hamburg

Berliner Tor 21
20099 Hamburg
Germany

Contact:

Email:
manumaker.dev@gmail.com

Terminology

- **ManuMaker:** Name of the Application this manual describes. This manual was created for ManuMaker, with ManuMaker. The screen capture and highlighting of image-features was achieved using the very capable tool "Greenshot".
- **Gallery:** The currently opened project in ManuMaker is also occasionally referred to as "The Gallery". This is owed to the fact that the reader navigates from one item to the next in a mostly linear fashion, just like someone would do in e.g. an art gallery. Jumping to the part which interests you the most is also allowed, via use of the slide list / tag search.
- **Gallery Slide:** The items in the gallery are also referred to as "slides". This is due to the similarities to slides in a digital presentation program. Every slide has a slide title which is displayed in the title-bar above the image and text editor, as well as in the slide list.
- **Slide List:** The list at the left side of the window. It has two tabs, one which permanently displays all slides in the current gallery (labeled "Index") and one which displays all slides matching your search keywords, if there are any.
- **Main Button Bar:** The button bar found at the bottom of the window in all tabs. Depending on the currently selected mode, it gives access to different tools to create documentation, and provides the buttons for linear navigation through the gallery.
- **Annotation Panel:** The section at the right side of the window, which contains the annotation text, tag utilities and the audio player.
- **Image View:** The element in the center of the window. The current slide's images are displayed here, if the current slide contains an image. In draft mode, the image view is presented as a grey rectangle even if no picture is present, in the other two modes the image view's width is shrunk to zero if the current slide contains no image, to make more space for the text.
- **Annotation:** refers to the text of a gallery slide which usually contains instructions. There are two different annotations attached to each slide:
draft - and *final* (or *refined*) Text. The refined text is the one visible to the reader of the manual, and can be styled in a WYSIWYG manner in the refine tab.
- **Gallery Browser:** The part of the UI which displays a given project. If you've opened this manual with ManuMaker, you're looking at the gallery browser right now.
- **Project:** A manual created or viewed in ManuMaker. Each instance of the software can only load one project at a time for viewing or editing.
- **Application Settings:** The window where the settings which are constant across all projects are selected. Mainly the camera-related options at this point.
- **Project Settings:** Every project has a set of settings attached to it, which are unique to the project (e.g. project name, type of documentation, author's name). All of these can be set at the creation of a new project, but only a few fields are mandatory. All settings for the current project can be changed at any time via the *Settings*->*Project Settings* menu. The project is automatically updated with the new information (e.g. information on the title slide).
- **Draft Tab:** The draft tab is one of three tabs available in the gallery browser. It contains all the tools necessary for quickly and conveniently *capturing* information. Use it if you want to add an image via camera or clipboard, or if you wish to record a step as audio, to capture all the essential details of a situation while you're using both hands for e.g. assembly work. To open the draft tab for a loaded project, select **Draft** in the vertical tab-bar in the upper right corner of the window.
- **Refine Tab:** The refine tab is one of three tabs available in the gallery browser. It offers the features to make recorded information look presentable. It modifies and adds to the information collected during the draft phase, and offers the functionality to e.g. include a table into the annotation text, add well-structured warnings or notices with pictograms to the annotation text, and format the annotation with headings, enumerations, bullet point lists and so on. To open the draft tab for a loaded project, select **Refine** in the vertical tab-bar in the upper right corner of the window.
- **View Tab:** The view tab offer the documentation user's perspective on the manual. It provides all the necessary features to retrieve the information contained in a manual, such as navigation through the slides (via forward / back button or via clicking on a slide in the slide list on the left) and searching tags of all three categories. It does not provide any features for editing. To open the draft tab for a loaded project, select **View** in the vertical tab-bar in the upper right corner of the window.

Intended Use

Use-Cases for ManuMaker V1.0 considered during its design

Manual types:

- Linear instruction manuals for specific tasks, like:
 - mechanical assembly
 - electrical assembly
 - soldering guides
 - maintenance procedure documentation
 - setup / commissioning of hard- and software systems
- Non-linear reference manuals for low- to medium complexity devices*.
 - user manuals
 - troubleshooting guides

*Low-complexity being a scientific electronic calculator, medium being a build-it-yourself 3D-Printer kit, complex being a subject which consists of multiple complex sub-assemblies, such as an automobile.

Documentation Subjects:

- Hardware systems (using the webcam). The size-limit of the subject is mainly set by the documenters workspace configuration and the length of the webcam cable. ManuMaker has been created mainly with devices in mind which can be placed on a desktop or workbench equipped with a PC and e.g. a desk lamp on which the camera can be mounted for steady and precise positioning.
- Software (using a screen capture tool)
- Systems which consist of hardware as well as software, such as a robotic arm and its control-software

Target User Groups:

- **Technical Writers** who are not set up with a content management system by their company and have been using a digital camera and office software for creating manuals.
- Note that if the the documentation format does not conform with a company's requirements, ManuMaker can still serve as an organisational tool for drafting the documentation.
In case of hardware-documentation done with a digital camera and an office-grade word processor the following steps are eliminated when using ManuMaker:
 - Operating the camera by hand
 - Organizing the notes for each step
 - Transferring the images from the camera's sd card to the PC
 - Sorting through the images to see which are usable
 - Renaming the images to give a hint for which step of the process they are documenting
 - Organizing the images and text information in the correct order in case of a step-by-step manual
- **Engineers** who need to document their work as well as the tools they create to facilitate their work (especially test stands and prototypes, which may not receive a complicated documentation due to shortage of time).
- **Entrepreneurs** who need to provide documentation for the products they manufacture or import and distribute .
- **Students**, especially in practically-oriented subjects of the S.T.E.A.M.-variety who need to document their contributions to intra- or extra-curricular projects where the basic functions of their work needs to be documented for posterity.
- **Makers**, who are these days developing incredible open-access technology at their desktops at home. Last but not least, ManuMaker is intended to reduce the time needed to create the often crucial and just as often non-existent documentation which makes these inventions accessible to the crowd.

Limitations

Limitations to be aware of:

Technical:

- **A webcam** connected to a computer can only be used by one application at the time. If the webcam selected in ManuMaker's **Application Settings** is already in use elsewhere when an instance of ManuMaker is starting up, it will not be usable in that new instance. The Application will notify you if a webcam is not available. If multiple webcams are connected to the system, go to Application Settings to select an available one, if needed. For just viewing a documentation or for documenting on-screen subjects, a webcam is not required.
- **The elements of a ManuMaker project** are saved in common formats such as .txt, .html and .png for simplicity's sake, and because this makes integration of a documentation project easier in the context of a versioning system like e.g. Subversion. Some drawbacks of this storage format are:
 - Component files can be corrupted accidentally by accessing and modifying them through the file explorer
 - Components can be opened externally and therefore not be writable when ManuMaker is attempting to save the current project. This may result in loss of data.
 - When copying a documentation project (e.g. when restoring a corrupted documentation from backup), great care has to be taken that it is a clean copy.

If a directory "Dir" contains a project with name "Project A" and another project named "Project A" is copied into the same directory, your OS may ask you to merge the project folders in some way. That may leave residual data of the initial project in "Dir" in the project structure, and cause ManuMaker to load wrong or incomplete data when loading from the merged folder.
- **Use with constantly updating versioning- or cloud services** can lead to loss of data because the updater may block a file while ManuMaker tries to modify it.

Workaround: Disable automatic updates of the file structure while handling documentation, or use a manual-commit based service.

Known cases where data-loss occurred:

 - Documentation Project saved in Dropbox (only service on hand during testing)
 - This list will be expanded as information is collected
- **The Export to printable HTML function:** It works when the conditions are right, although there are no customization options available in the current version of ManuMaker (The default setting is: Portrait orientation, 1 Slide per Page). Please refer to the chapter "*Exporting the Project to HTML Format*" in this manual for some guidelines on how to create a printable ManuMaker project.

Conceptual:

- **Multi-Topic-Manuals:** Due to the shallow structure of the documentation projects created in ManuMaker (only project->chapters->instructions, no sub-chapters), it is recommended to limit a documentation project to one specific task, e.g. assembly of an apparatus.

If a product needs a manual consisting of several instruction-sets such as assembly, commissioning, troubleshooting, and maintenance, it is recommended to create one ManuMaker-project for each purpose, to avoid making the documentation difficult to navigate through.
- **Complex Troubleshooting:** To troubleshoot a complicated product, the corresponding manual likely requires a lot of cross-references ("If situation A occurs, read on on page X, read page Y for situation B..."). Reasons are:
 - ManuMaker V1.0 does not support embedded links between slides
 - Slide titles may be reformulated ad-hoc during the refinement process, as better formulations come to mind, which requires an impractical amount of management-overhead of references-by-slide-title.
 - The slide- and chapter numbers change across the whole project when a slide is inserted near the beginning of the manual, making manual referencing by chapter- or slide number impractical.
 - A possible workaround would be to use the tag-search feature for cross referencing ("If situation A occurs, type <code A> into the search box and keep reading..."). Then it is just a matter of distributing unique tags to the referenced slides. But since these links need to be managed by the documenter, this workaround is not an ideal solution.
 - For troubleshooting situations, flowcharts can be very helpful. As of ManuMaker V1.0, importing flowcharts in e.g. an XML form is not supported.

While inserting images of flowcharts may work for small charts, the lack of a zoom and scroll functionality in ManuMaker's **image view** makes viewing large images impractical.

Summary of the Available Features

Purpose of the Draft Tab:

Use the **Draft Tab** to quickly capture any operational information (visual, text, audio) in a way that causes as little distraction from the task at hand as possible.

ManuMaker takes over most of the information structuring.

In the **Draft Tab**, you can:

- Add still images or image series (**Stop Motion Pictures**)
- Capture images using a webcam or inserting them via the clipboard, e.g. using a snipping tool
- Open images in the system's default editor for graphical annotations
- Write down a draft of the instructions without regard for formatting or styling
- Record an audio clip (one can be recorded per slide, default microphone is used) which can be replayed in draft and refine tabs
- Remove the audio or image from from slides
- Delete slides from and add slides to the gallery
- To make the contents searchable, add keywords to the instructional slides. There are three categories of keywords:
 - Relevant to the instructions ("Tags")
 - List of tools used in a slide's context. The tools listed in all slides can be compiled to a list.
 - List of parts used, e.g. the parts added to the assembly in the context of a slide. The parts listed in all slides can be compiled to a list.

Purpose of the Refine Tab:

Use the features of the **Refine Tab** to make the captured information presentable and include additional data.

- Load the text from the draft tab into the rich text editor in the refine tab
- Listen to the recorded audio draft and transfer the information to the text annotation
- Add warnings and notices in the instruction text
- Add tables in the instruction text as HTML (alternatively, add tables as images to the image view)
- Add formatting slides: title- or chapter slide
- Adjust the order of slides in the gallery
- Add additional chapters, e.g. "safety" or "troubleshooting"

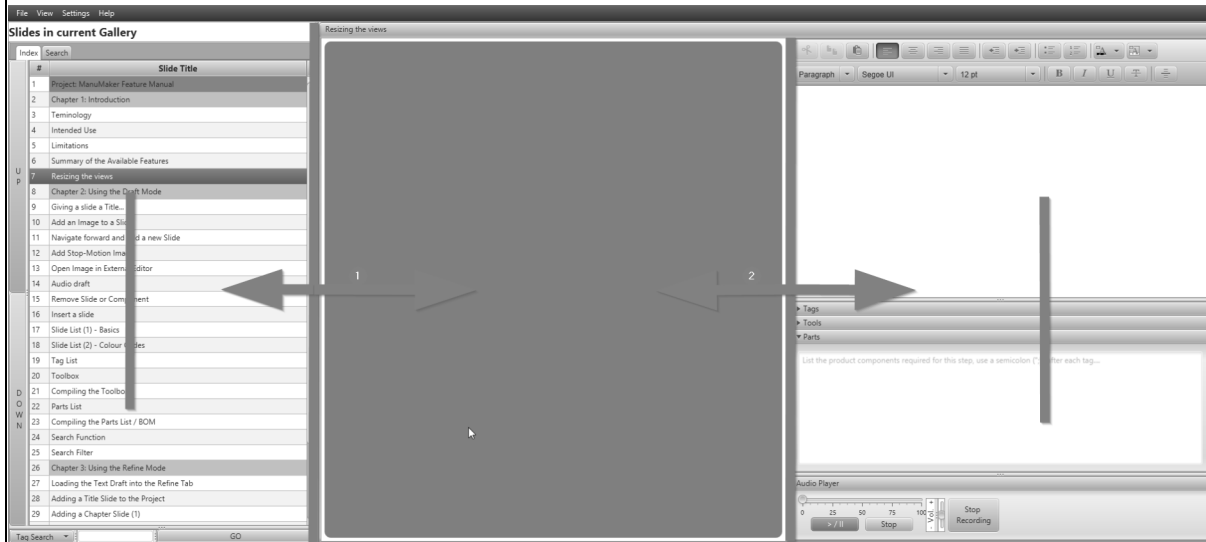
Purpose of the View Tab:

Use the **View Tab** when *using* the documentation.

In the **View Tab**, you can:

- Navigate through the gallery using the arrow buttons
- Read the annotation text as seen in the refine tab
- See the tags (non-editable)
- Search slide tags (any combination of the three categories can be searched at one time)
- Be sure you will not modify the documentation, as the view tab does not offer any editing functionality.

Resizing the views



The Gallery Browser consists of four major sections:

- The **Main Button Bar** at the bottom of the window, which is not resizable to a significant degree.
- The **Slide List** on the left side of the window
- The **Image View** in the center
- The **Annotation Panel** with text, tags and audio features.

The two vertical separators (marked with long red lines in the picture) can both be moved horizontally.

The separator **(1)** between slide list and image view can be moved as far as the separator **(2)** between image view and annotation panel, and vice versa.

Both **(1)** and **(2)** are limited in how far they can move towards the frame of the window, as both slide list and annotation panel have a minimum width, as denoted by the shorter vertical lines in the picture.

Chapter 2:

Settings

**Intended
Readers:**

All users of the ManuMaker application

Author(s):

Daniel Lachmann

Publisher:

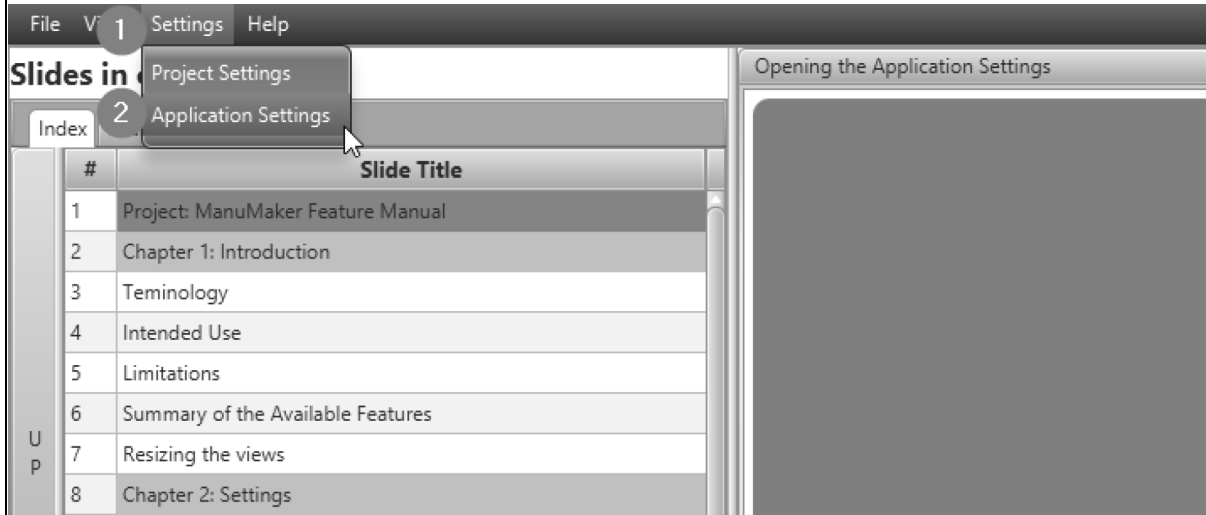
HAW Hamburg

Berliner Tor 21
20099 Hamburg
Germany

Contact:

Email:
manumaker.dev@gmail.com

Opening the Application Settings

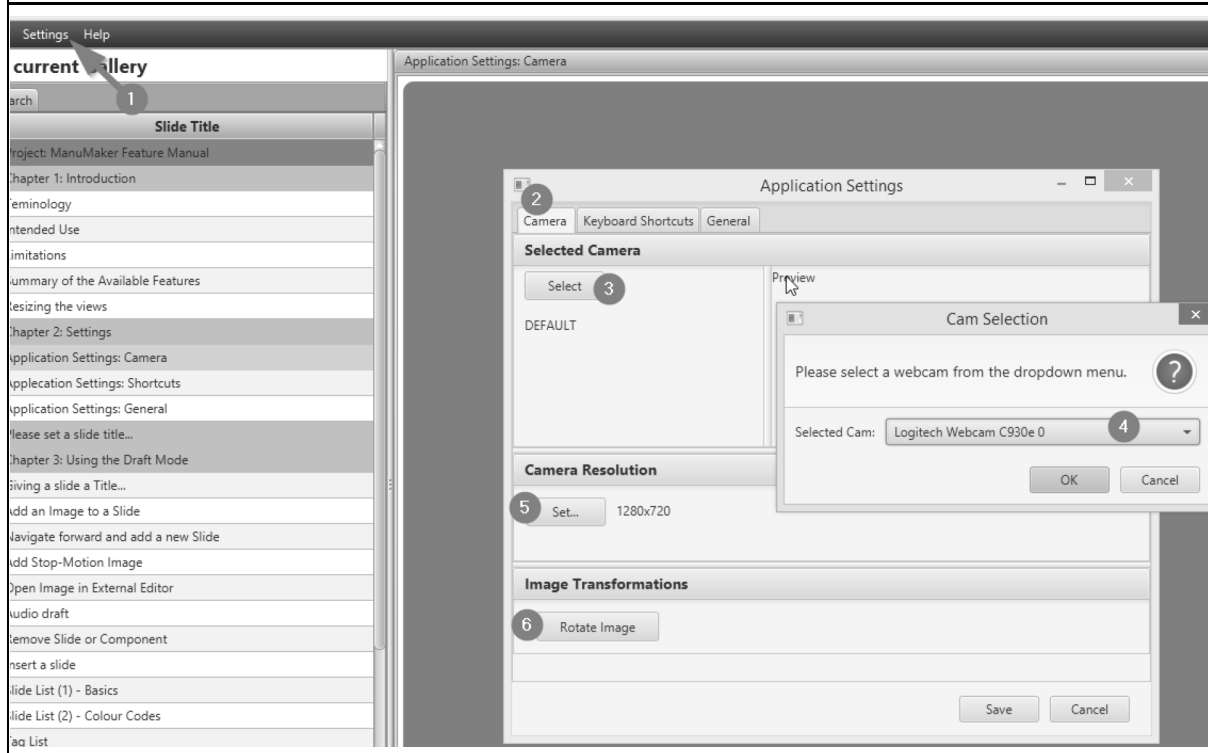


The Application Settings Window:

To open the **Application Settings**, click on

Settings (1) -> *Application Settings (2)* in the top menu bar

Application Settings: Camera



The **Camera Tab (2)** of the Application Settings contains:

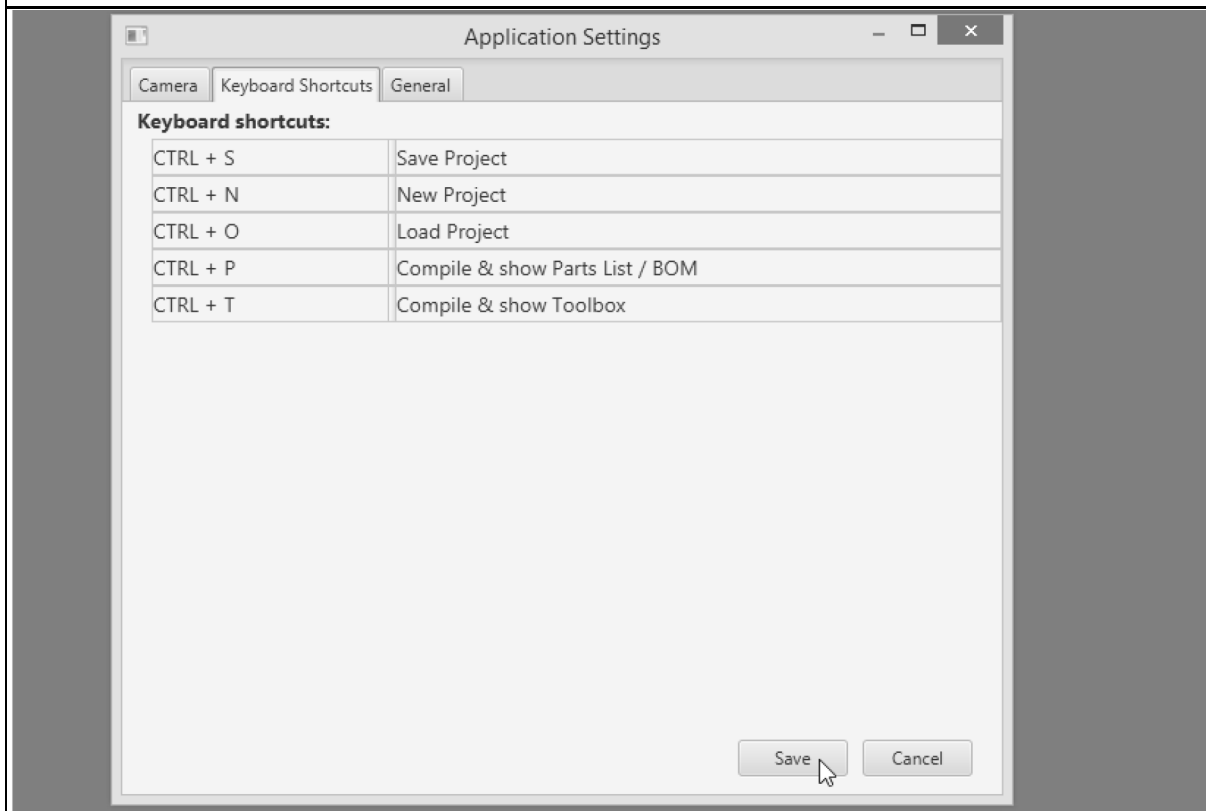
- The **Select (3)** button to set the webcam used for documentation, which opens a dialog with a dropdown list of all available webcams (4). The name of the camera is displayed in the label underneath the button (3).
- A button to set the camera's resolution (5) and a text label which displays the currently selected resolution (here: 1280x720)
- The **Rotate Image (6)** button to rotate the image of the webcam by multiples of 90°, to enable adapting to different camera mounts.



NOTICE

- The resolutions presented in the camera resolution settings are not necessarily supported by your webcam. The webcam API used in ManuMaker does not yet support querying the webcam for supported resolutions at HD level.
- This appears to be a safe-failing feature, as all tested cameras defaulted to standard VGA resolution when an unsupported resolution was selected.
- If you select a high resolution and the camera doesn't seem to have adopted it, the likely cause is that the selected resolution is not supported.

Application Settings: Shortcuts



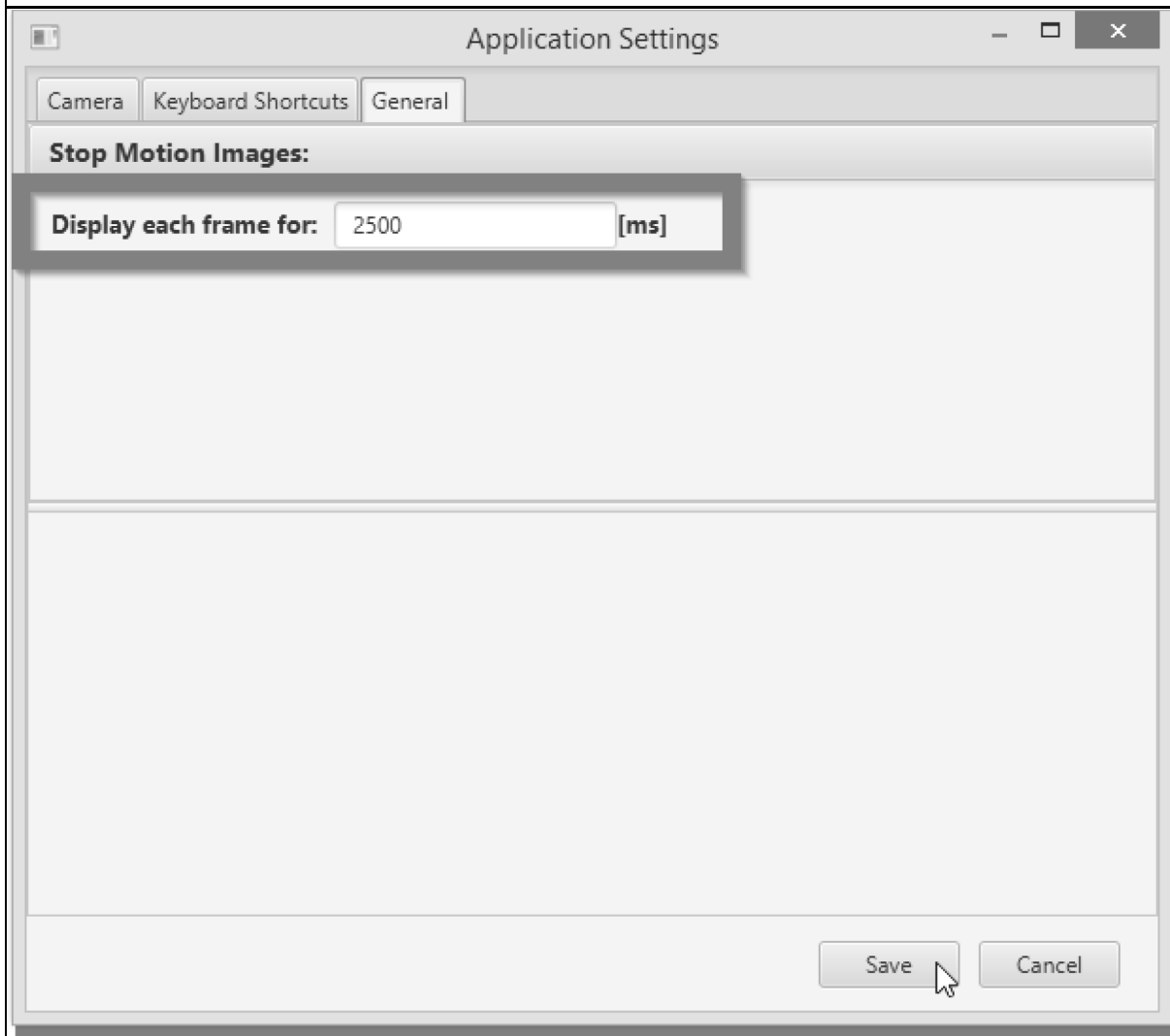
ManuMaker supports keyboard shortcuts for:

- Saving the project
- Creating a new project
- Loading an existing project
- Showing the list of all required parts (BOM)
- Showing a list of all required tools

In Version 1.0, the key-combinations are fixed (see image) and cannot be changed. The most important for most people will likely be the commonly used combination *CTRL + S* to save the project.

NOTE that saving can take a while for larger documentation projects, and that the GUI is non-responsive while saving takes place.

Application Settings: General



The General Tab:

As of ManuMaker V1.0, the application setting's *General* tab only contains the delay-setting for stop-motion pictures, which defines the duration every frame in the image series is displayed.

The value can be set between 250ms and 25000ms (0.25s to 25s, respectively).

The given value must be a positive integer.

Project Settings: Mandatory Fields

Project Settings

-Fields printed in bold are required.
-Use HTML formatting tags to define e.g. linebreaks (
) or bold type (...).

Project Title

ManuMaker Feature Manual

Product Name

ManuMaker

▶ Product Version(s) covered by this Manual

Documentation Type

User Manual

▶ Documentation Revision Number

▶ Target Audience

Author

Daniel Lachmann

▶ Company Name

▶ Company Address

Company Logo

D:\AppDocumentation\Resources\Logo.png

Load Delete

ManuMaker V1.0

HAW HAMBURG

▶ Contact Information

▶ Additional information for the title page

Project Path

D:\AppDocumentation\ Select Project Path...

The **Project Settings Window** appears whenever a new project is started, or after clicking *Settings->Project Settings* in the top menu bar.

There are two types of text-input found within the project settings window:

1. Text Fields: Single line text input, does not support linebreak by pressing ENTER
2. Text Areas: Multi line text input, make newline with ENTER

All text put in this window can be appended by html codestyling (old-fashioned but effective) to get e.g. bold-printed words or linebreaks inserted to the plain-text inputs. See **(1)**

Mandatory fields in the project settings:

- **Project Title (2)**: The title will only be visible in the slide list as the title of the title slide.
- **Product Name (3)**: The product being documented.
- **Documentation Type (4)**: E.g. "Assembly Manual", "User Guide" or "Maintenance Manual".
- **Author (5)**: The name of the author.
- **Project Path (6)**: Note that the path selected in the chooser dialog will be appended by a directory named "\<project title>\".

Project Settings: Optional Fields

▼ Product Version(s) covered by this Manual 1

1.0

Documentation Type

User Manual

▼ Documentation Revision Number 2

1.0.0

▼ Target Audience 3

Users of the ManuMaker application

Author

Daniel Lachmann

▼ Company Name 4

HAW Hamburg

▼ Company Address 5

Berliner Tor 21
 20099 Hamburg
 Germany

Company Logo

D:\AppDocumentation\Resources\Logo.png

Load 6

Delete 7

ManuMaker V1.0 8

HAW HAMBURG

▼ Contact Information 9

Email: manumaker.dev@gmail.com

▼ Additional information for the title page 10

Label for the additional information

Please fill in the additional information to display on the Title page of the documentation...

Optional fields in the Project Settings:

1. Product version or versions which are covered by the manual
2. The revision number of the documentation (The format can be chosen freely)
3. Definition of whom this manual was written for (E.g. maintenance technicians, engineers, end-users)
4. Name of the company which builds the product and publishes the manual
5. Address of the publishing company
6. A button for selecting a logo (Any image file in jpg, bmp, png or gif format, display size is constrained by width-parameter).
7. A button to delete the image.
8. A preview of the currently selected logo. Note that when a new logo is selected to replace an existing one, the gallery needs to be saved, closed and reloaded for the change to effect on the title slide.
9. Contact information to be printed on the title slide of the documentation
10. A heading and a content-field for displaying additional information on the project's title slide.

Chapter 3:

Using the Draft Tab

**Intended
Readers:**

All users of the ManuMaker application

Author(s):

Daniel Lachmann

Publisher:

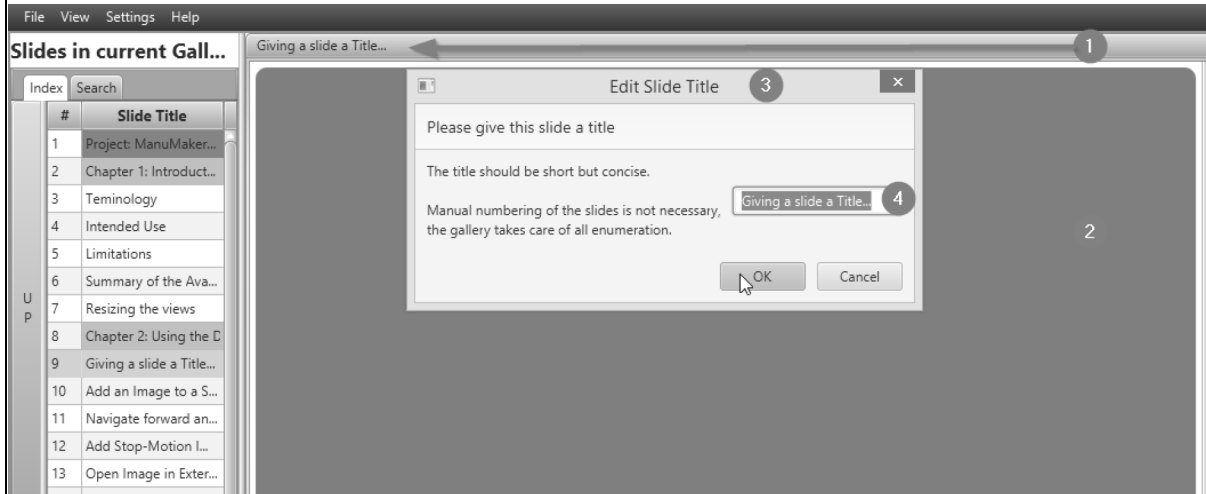
HAW Hamburg

Berliner Tor 21
20099 Hamburg
Germany

Contact:

Email:
manumaker.dev@gmail.com

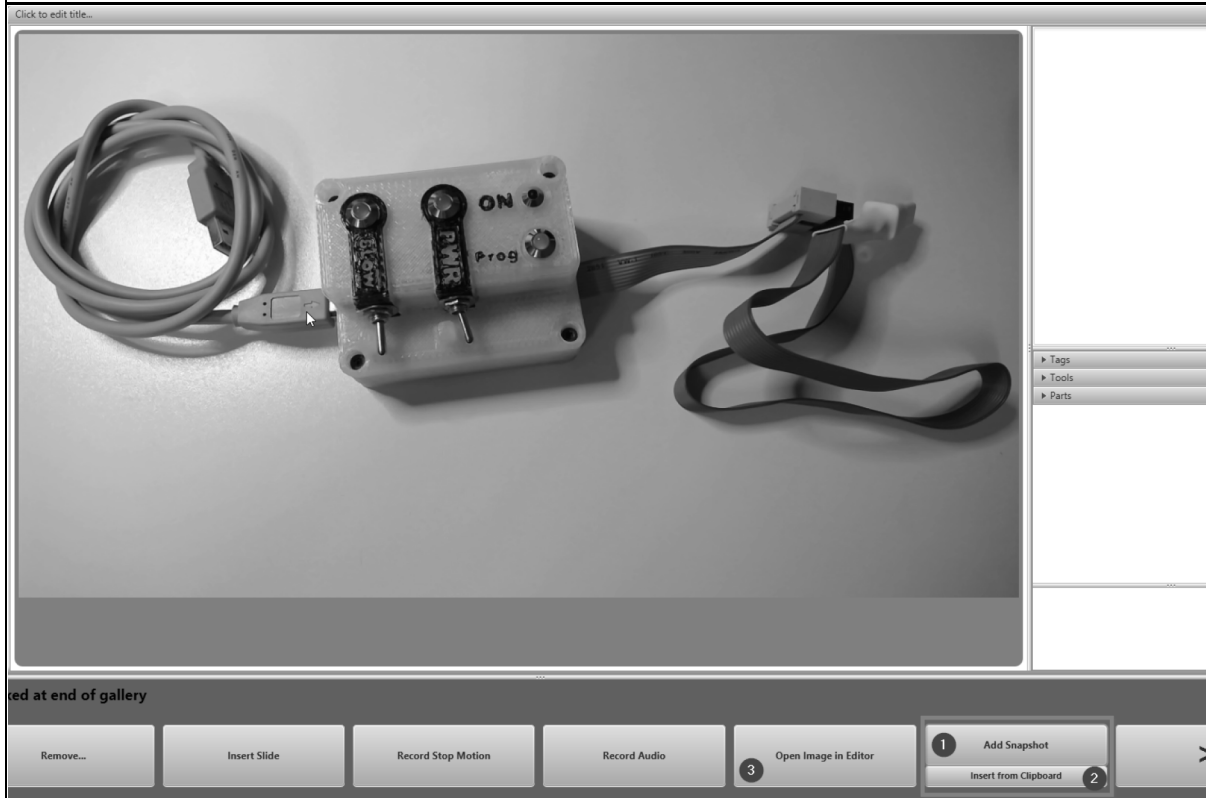
Setting the Slide Title



To add or modify a Slide's Title:

1. Click on the **Title Bar** above (1), or on the **Image View** (2)
2. The **Edit Slide Title** dialog (3) opens
3. Type a short, concise title for this step into the **Text Field** (4)
4. Confirm with **OK**.

Adding an Image to a Slide



There are two ways to add an Image to the Current Slide:

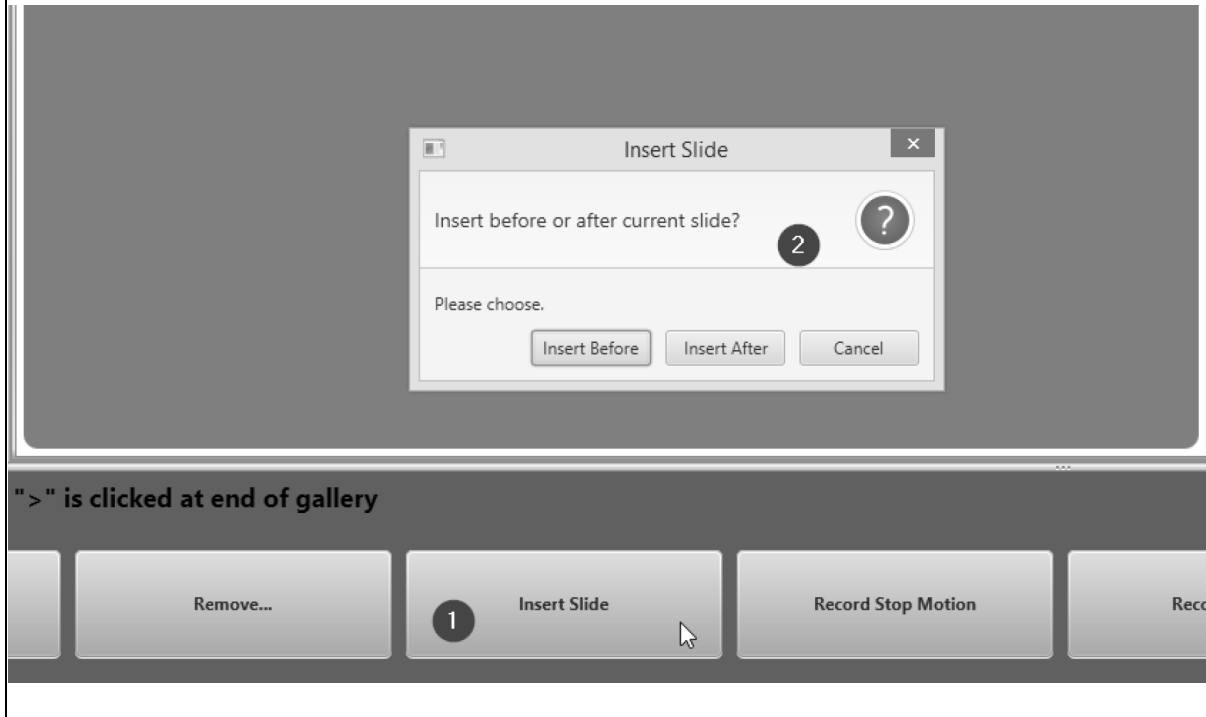
For hardware documentation

- Click **Add Snapshot (1)**, to start the camera stream.
- You can now position the subject and your camera until you are satisfied with how it looks.
- Clicking the **Snapshot (1)** again will capture the last picture from the camera and stop the camera stream.

For documenting things presented on a computer screen

- Copy the screen-portion to the clipboard (using a snipping tool is recommended)
- Insert it using the **Insert from Clipboard (2)**.

Inserting a Slide



To insert a Slide:

If information needs to be added between two slides, navigate to one of the adjacent ones and click the button **Insert Slide (1)**.

A dialog **(2)** gives you the choice to insert the new blank slide before or after the current one.

Slide List (1) - Basics

The screenshot displays the 'Slides in current Gallery' interface. On the left, a table lists slides with their titles and indices. The 'Slide List' panel on the right is currently empty. Numbered callouts (1-4) highlight key features: (1) the main slide list area, (2) the selected slide, (3) the 'UP' button, and (4) the 'DOWN' button.

#	Slide Title
1	Add an Image to a Slide
2	Navigate forward and add a new Slide
3	Create a Stop Motion Image
4	Open Image in External Editor
5	Audio draft
6	Remove Slide or Component
7	Insert a slide
8	Slide List
9	Tags & Tag search

Using the Slide List:

The **Slide List** (1) presents a complete overview of all slides in the Gallery. The currently selected slide is highlighted (2) and can be moved up (3) or down (4) in the list by clicking on the respective buttons.

By clicking on an item in the list, the **Gallery Browser** jumps to the selected position in the gallery.

The **Slide List** also displays the title of each slide, so choosing well-formulated titles makes the navigation easier.

Slide List (2) - Colour Codes

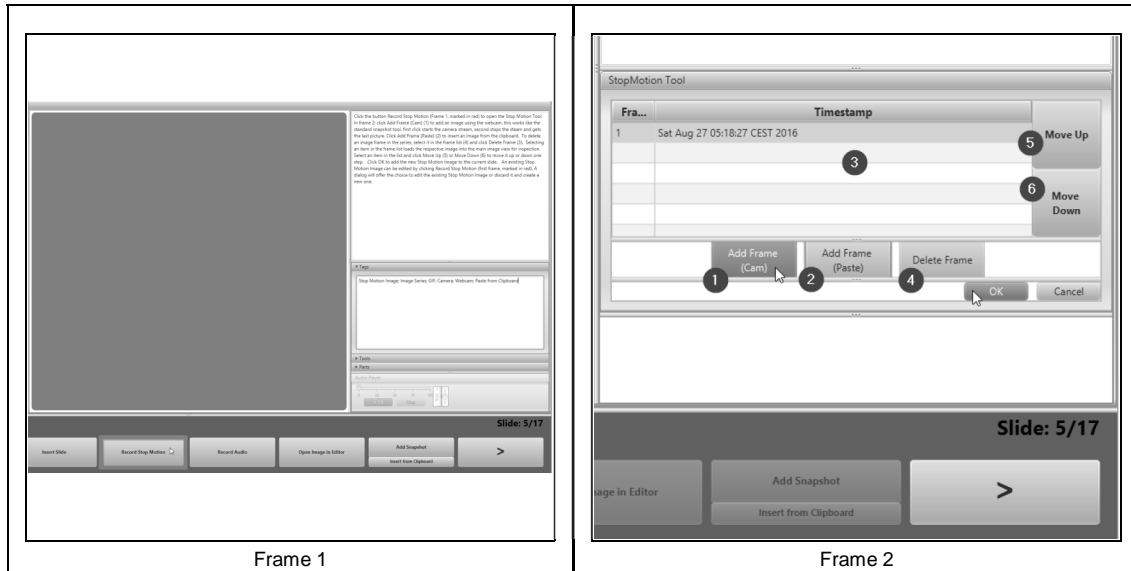
Slides in current Gallery

Index		Search	1	2	3	4
#	Slide Title					
1	Project: ManuMaker Feature Manual		●			
2	Chapter 1: Introduction			●		
3	Capabilities				●	
4	Intended Use					●
5	Chapter 2: Using the Draft Mode					
6	Giving a slide a Title...					
7	Add an Image to a Slide					●
8	Navigate forward and add a new Slide					
9	Add Stop-Motion Image					
10	Open Image in External Editor					

The slides in the list can be distinguished by colour:

- **(1)** (blue grey) is the title slide of the project.
 - It must always be the first slide in the gallery.
 - The title presented in the list is prefixed with "Project: ".
- **(2)** (light blue) denotes a chapter slide.
The title presented in the list is prefixed with "Chapter : "
- **(3)** denotes a standard slide in the gallery, which is not currently displayed.
- **(4)** denotes the slide currently displayed in the **gallery browser**.
Note: The selected slide is the only one whose colouring extends to the slide's ordinal number.

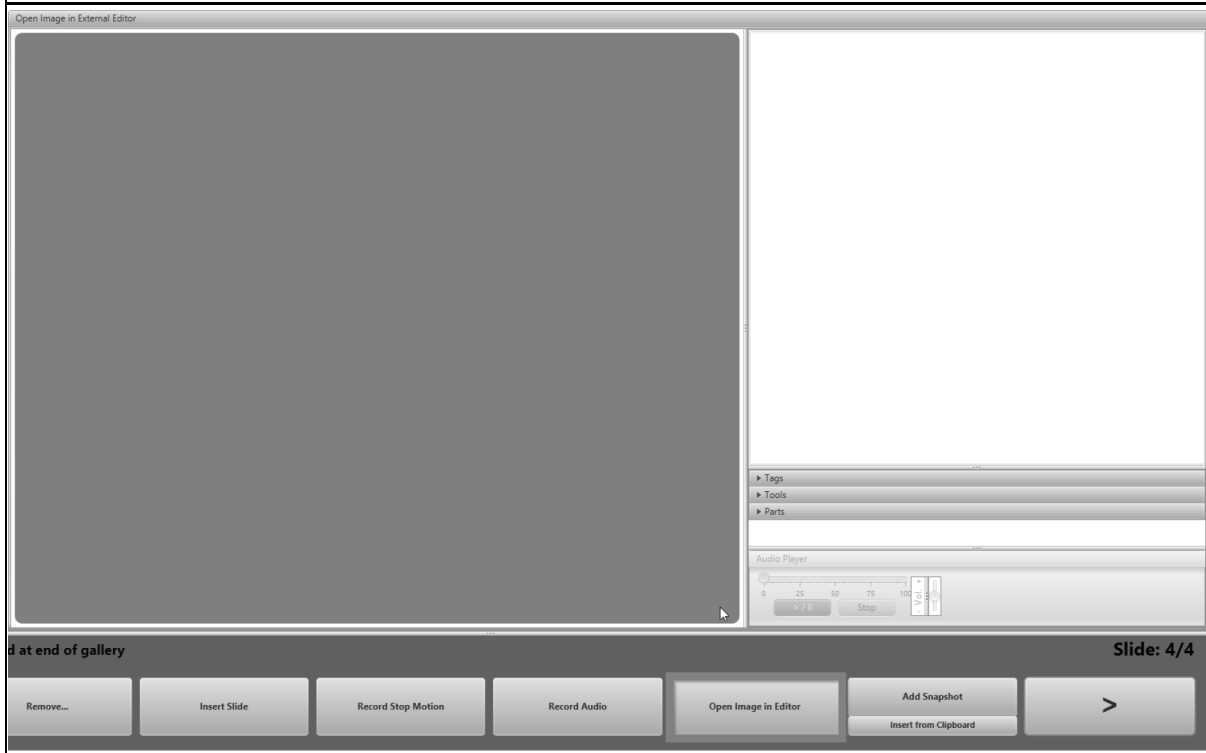
Adding a Stop Motion Picture



To add a Stop Motion Picture:

- Click the button **Record Stop Motion** (Frame 1, marked in red) to open the **StopMotion Tool**.
- In Frame 2: click **Add Frame (Cam)** (1) to add an image using the webcam. This works like the standard snapshot tool, first click starts the camera stream, second stops the steam and gets the last picture.
- Click **Add Frame (Paste)** (2) to insert an image from the clipboard.
- To delete an image frame in the series, select it in the frame list (3) and click **Delete Frame** (4).
- Selecting an item in the frame list loads the respective image into the **Image View** for inspection.
- Select an item in the list and click **Move Up** (5) or **Move Down** (6) to move it up or down one step.
- Click **OK** to add the new stop motion picture to the current slide. An existing stop motion picture can be edited by clicking **Record Stop Motion** (first frame, marked in red). A dialog will offer the choice to edit the existing stop motion picture or discard it and create a new one.

Opening Images in an External Editor



Editing a Captured Image:



NOTICE

- The image is loaded from the filesystem.
- Make sure the project is saved before using this function, otherwise an outdated image may be loaded.

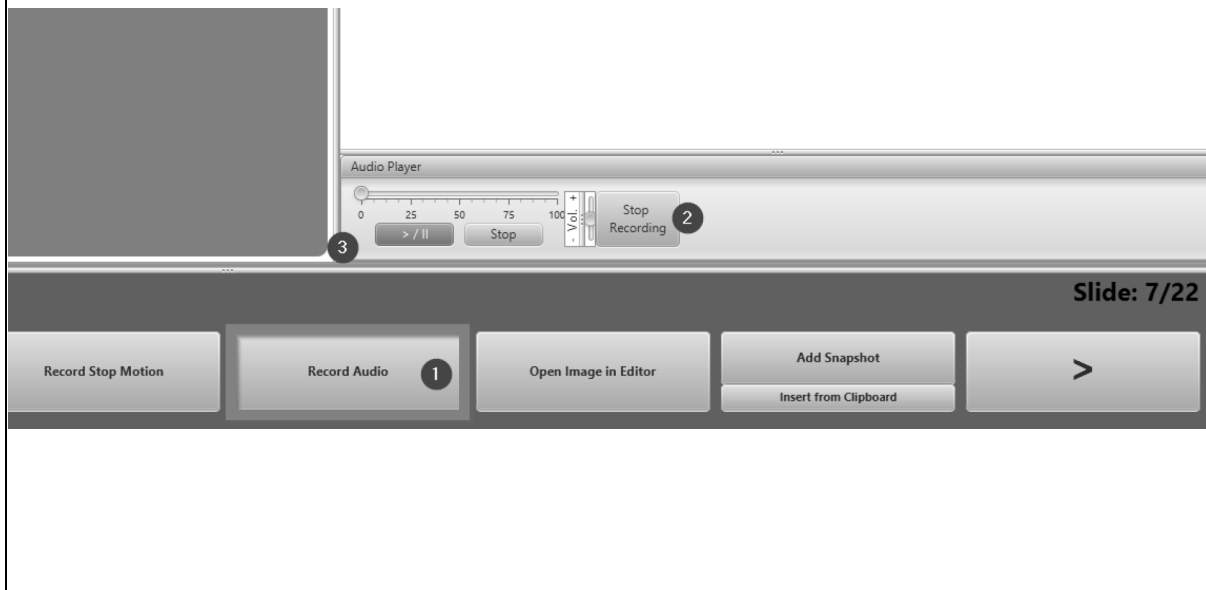
To edit a captured image (be it from webcam or from the clipboard), click **Open Image in Editor** (See red frame).

This will open the image in your system's default editor.

To re-insert an image back into the gallery:

- Copy the image to clipboard in the editor
- Insert it into the current gallery by clicking **Insert From Clipboard**.

Audio Draft



Adding an Audio Draft:

To log critical information during a process that requires use of your hands, you can record an audio draft of a steps documentation.

To start recording, click the button **Record Audio Draft** (1) and speak into your default microphone.

When you're done, click the **Stop Recording** button (2), and the audio clip will be attached to the current slide.

To replay the audio draft, you can use the **Audio Player** (3) available in the **Draft** and **Refine** tabs. The **Audio Draft** is a tool exclusively intended for the documenter, and is hence not available in the **View** tab.

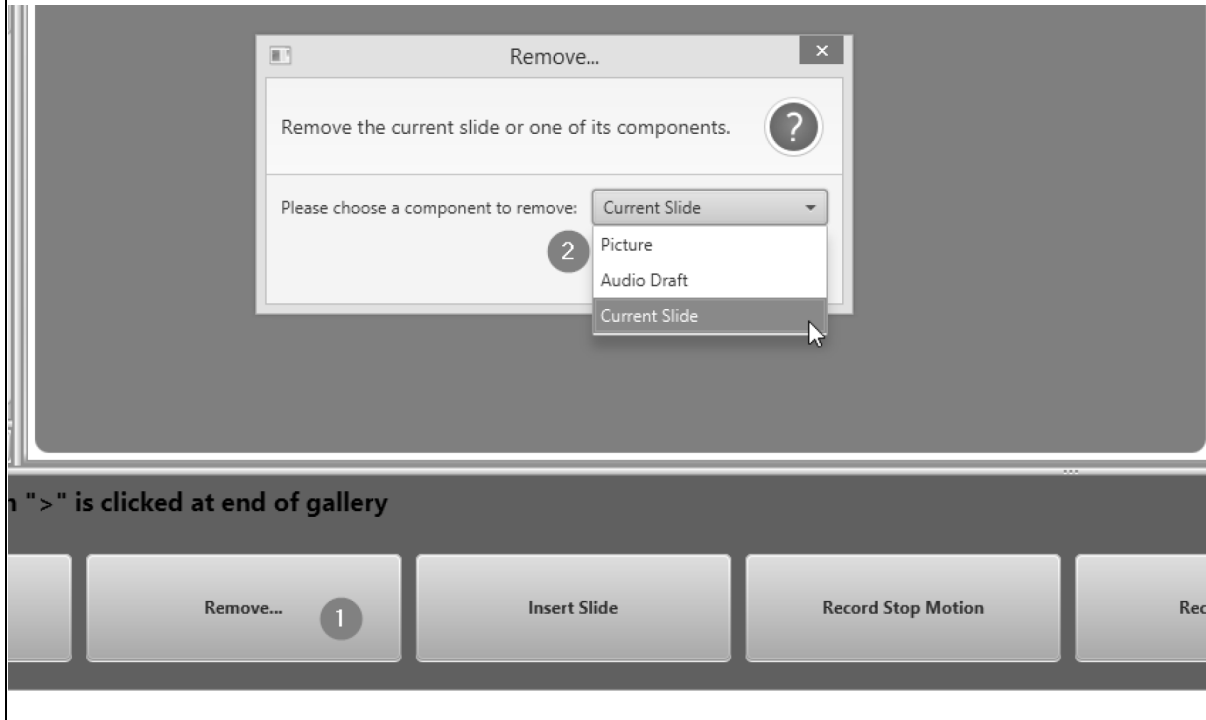
All relevant information from the **Audio Draft** should be transcribed to text during the refinement-phase of the documentation (as of **ManuMaker Version 1.0**, this has to be done manually).



NOTICE

You can store ONE audio clip per slide.
If you wish to overwrite the existing one, just click on (1) again and confirm that you wish to discard the existing audio clip

Removing Slides or Components



To remove the current slide or any of its external components such as the **Picture** or the **Audio Draft**, click on the button **Remove... (1)**.

A dialog **(2)** will show where you can chose which component to delete:

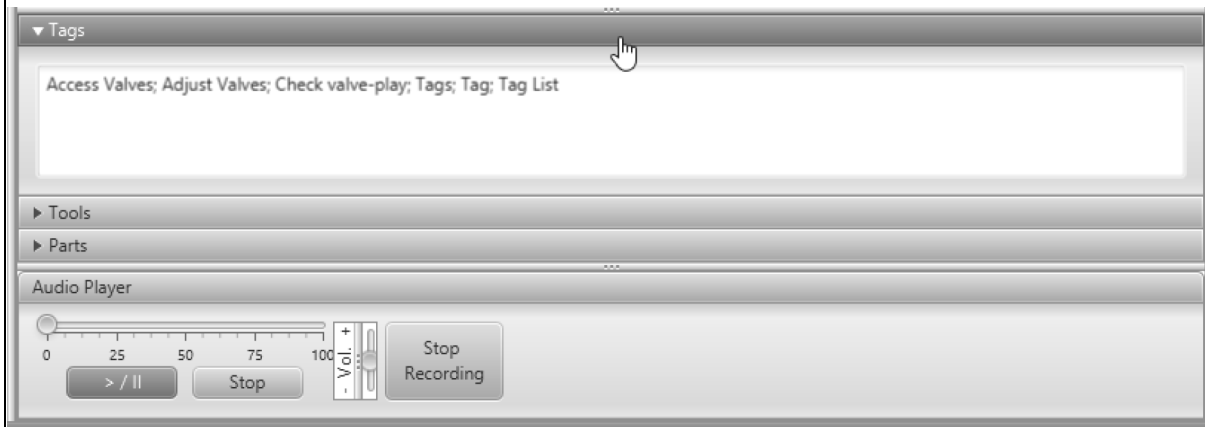
- The complete current slide with all its text, images, audio...
- The current image, in case the slide shall be text-only
- The audio draft



NOTICE

Components and slides are permanently removed. There is no mechanism to get them back, once deleted.

Tag List



Adding Tags to Slides:

All slides have a **Tags** field. The documenter provides tags for each slide, separated by a semicolon (;).

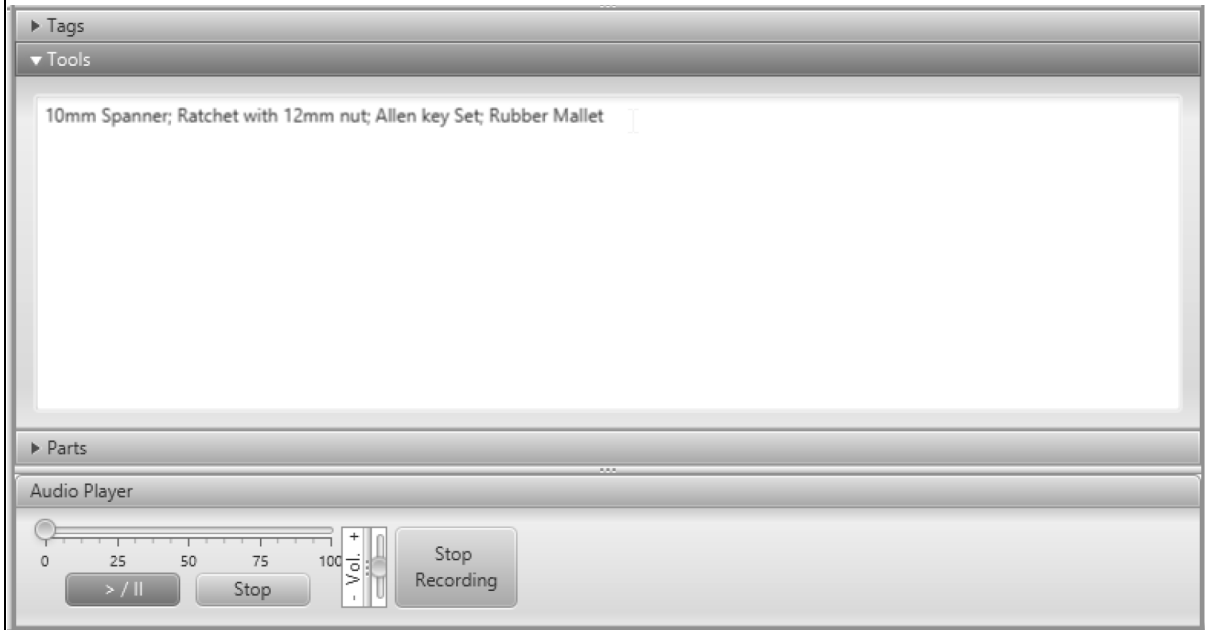
The tags can consist of single or multiple words, and should contain the keywords related to the current slides instructions. E.g. *gearbox maintenance; replace shaft seals;*



NOTICE

Include synonyms of the keywords used in the instruction text.
This raises the chance of success for keyword search.

Toolbox



Using the Toolbox:

All Instructional slides have a **Toolbox** field.

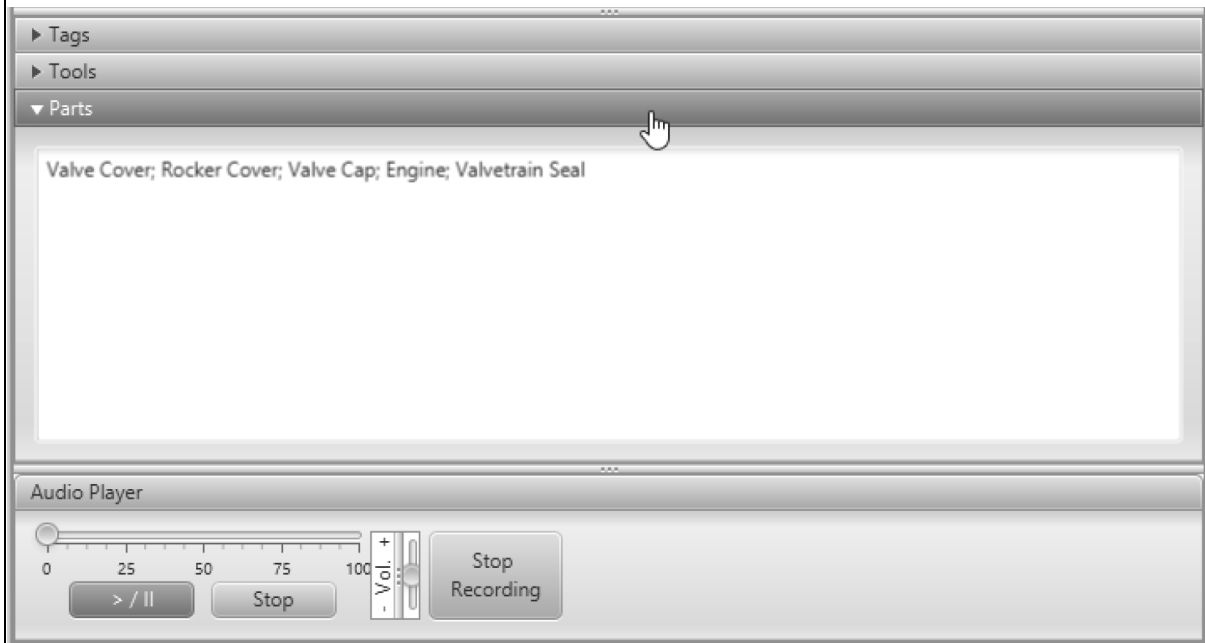
Here, the documenter specifies the tools needed for this specific step.



NOTICE

- Do not add synonyms for tools.
- The tools are compiled into a tool list, different words for the same tool make it less readable.
- Use the most common term for any given tool, and be consistent.

Parts List



Using the Parts List:

All instructional slides have a **Parts** field.

In an assembly manual, the documenter specifies the parts which are added to the assembly in the current step.

In a repair or maintenance manual, the parts being maintained as well as the parts being handled in the process are mentioned.

If the parts have identification numbers, add the numbers to the **Parts List** to enable search by part-number.

Chapter 4:

Using the Refine Tab

**Intended
Readers:**

All users of the ManuMaker application

Author(s):

Daniel Lachmann

Publisher:

HAW Hamburg

Berliner Tor 21
20099 Hamburg
Germany

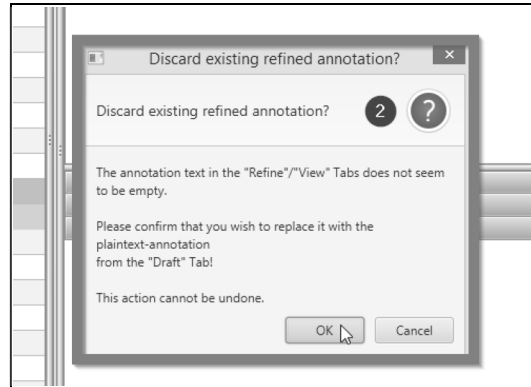
Contact:

Email:
manumaker.dev@gmail.com

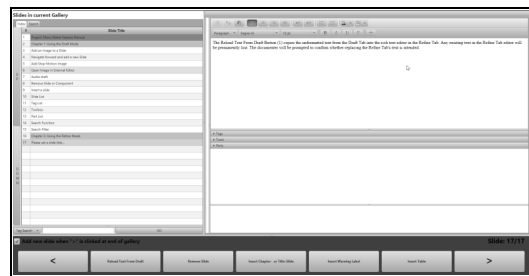
Loading the Draft Text into the Refine Tab



Frame 1



Frame 2



Frame 3

To load the Draft Text into the Rich-Text Editor in the Refine Tab:

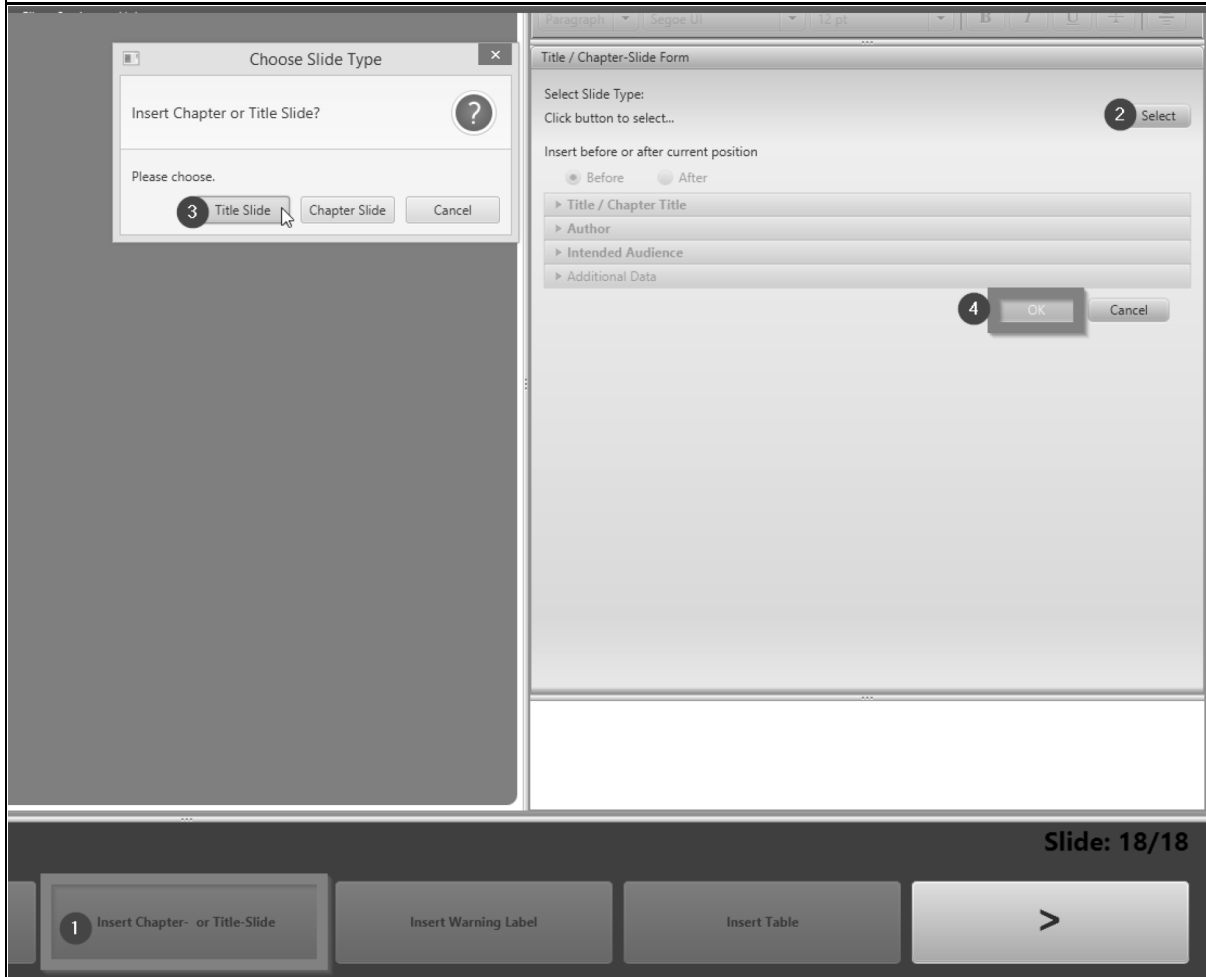
- [Frame 1] Click on the **Reload Text From Draft** button (1)
- [Frame 2] You will be prompted by a dialog (2) to confirm whether replacing the Refine tab's text is intended.
- [Frame 3] The unformatted text from the Draft tab is copied into the rich text editor in the Refine tab.



NOTICE

Any existing text in the Refine tab editor will be permanently lost.

Adding a Title Slide



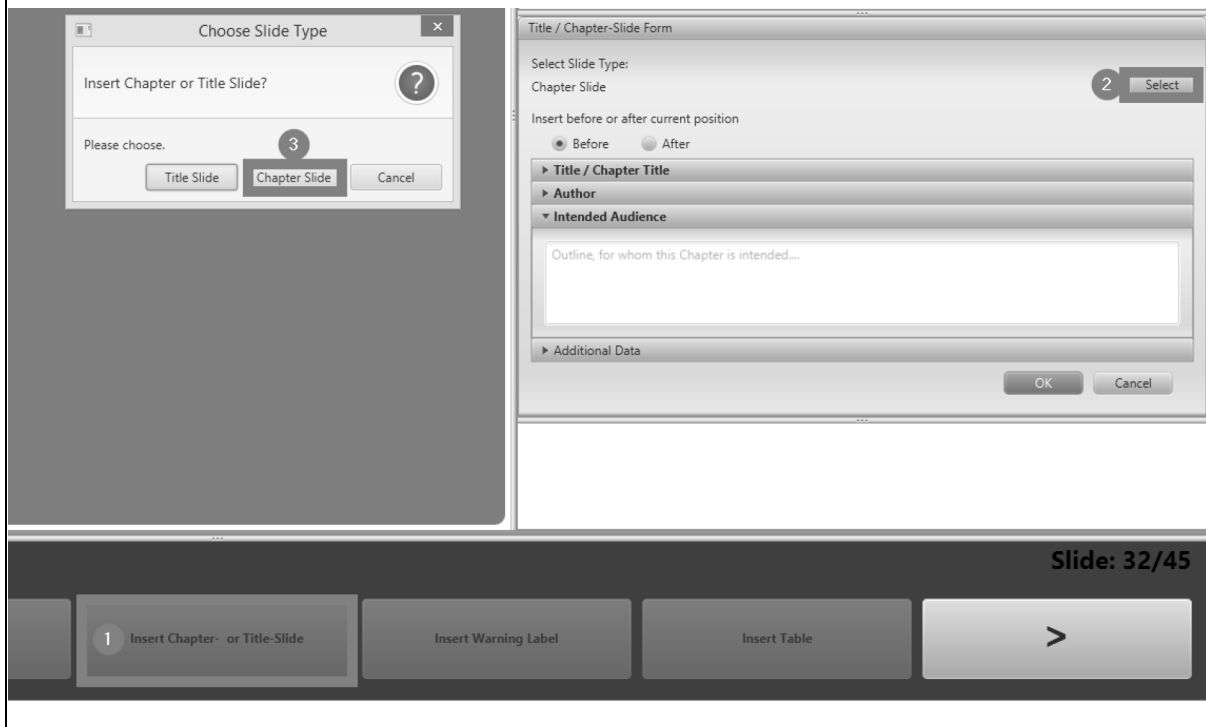
To add a Title Slide to the Documentation:

1. Click on **Insert Chapter- or Title Slide (1)**.
2. Click **Select...** (2)
3. Click **Title Slide (3)**
4. Confirm with **OK(4)**.

The title slide information (e.g. type of documentation, company logo to be displayed) can be set via **Settings->Project Settings** in the top **Menu Bar**.

After modifying the **Project Settings**, the **Title Slide** is automatically updated.

Adding a Chapter Slide (1)

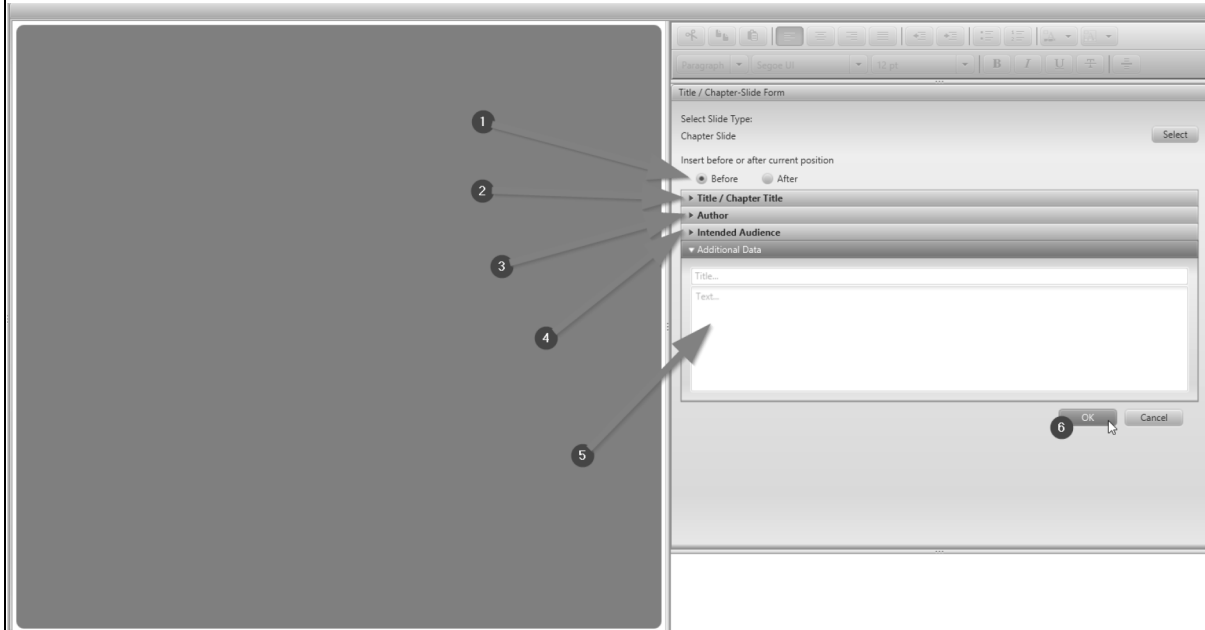


To add a Chapter Slide:

1. Click **Insert Chapter- or Title Slide (1)** in the **Refine Tab**.
2. Click on **Select(2)** in the **Title / Chapter Slide Form**
3. Choose **Chapter Slide(3)** in the popup dialog.

See next slide on how to proceed.

Adding a Chapter Slide (2)



Fill out the form:

1. Select whether the chapter should start before or after the current slide **(1)**
2. Type a title in the respective **Text Field(2)**.



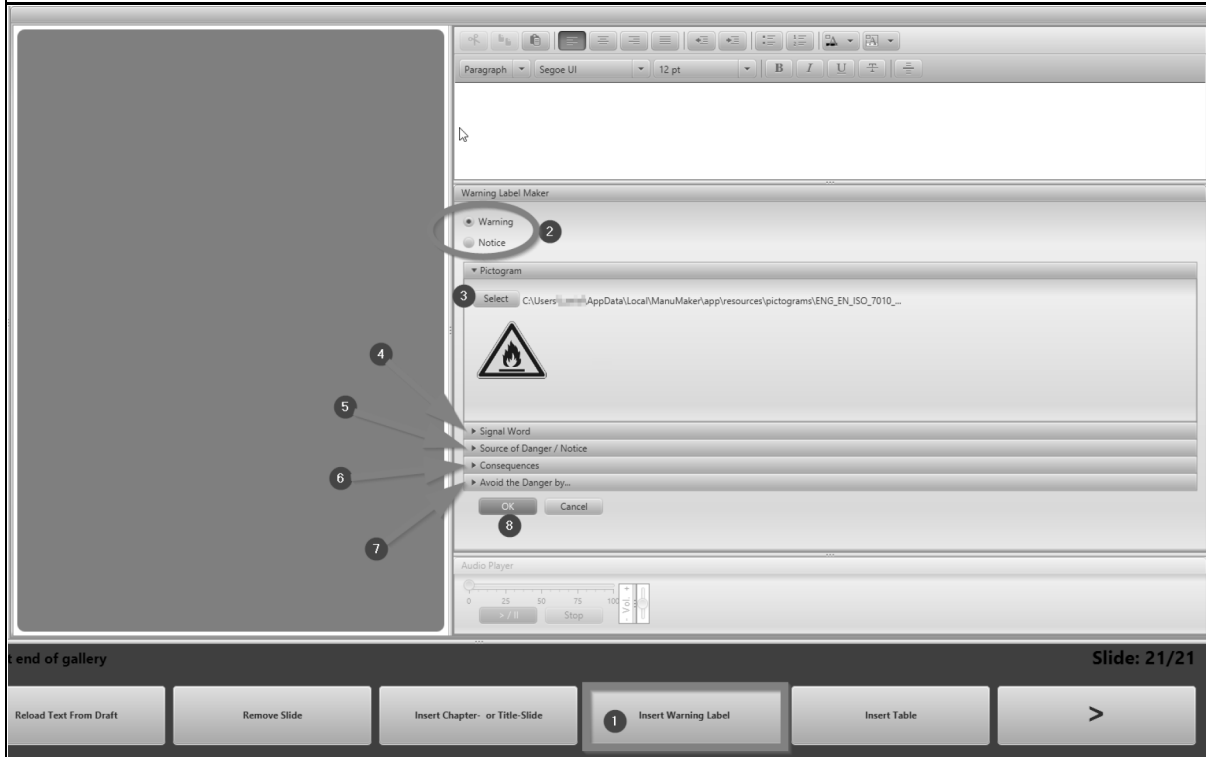
NOTICE

DO NOT add a "Chapter:" or "Chap."-prefix to the Title Field.
All chapter slides are automatically titled with a "Chapter :
"-prefix, where applicable.

DO NOT manually number the Chapters (e.g. in the title field). The chapter number is automatically assigned and updated whenever a new chapter is added to the gallery.

3. Fill out the mandatory fields: **Title (2), Author (3), Intended Audience (4)**
4. There is an extra Text Field **(5)** which can be used for presenting additional information on the chapter slide.
This extra information can be headed by a custom title which will be printed in bold.
5. Confirm your input by clicking **OK** and the new slide will be added to the gallery at the specified location.

Inserting a Notice or Warning



To insert a Warning Label or a Notice:

- Click the **Insert Warning Label** button (1)
- Select the desired label type (2).
- Load a warning pictogram from the filesystem (3).
A selection of *EN ISO 7010* pictograms are packaged with the ManuMaker application.
You can find them in the installation directory of the application at `..\ManuMaker\app\resources\pictograms\..`
- Fill out all the the accessible text fields in the form. **Notices** only require a **Signal Word** (4) and a **Notice Text** (5), **Warnings** require all fields (4) to (7) to be filled out.
- To insert the label into the text, *right-click* in the desired location in the **Rich Text Editor** on the **Refine** tab, and select the *Paste* option in the **context-menu**.
- The label is then inserted at the specified position in the text. The pictogram may only be visible after reloading the slide.

Inserting a Table into the Annotation Text

Table Import Form

Table Caption 2

Comparing different APIs (anonymized) regarding their compatibility with ManuMaker's proposed features

Table Identifier 3

1

File Import

Please select file path...

4 Choose File...

5 OK Cancel

Slide: 35/47

Insert Chapter- or Title-Slide Insert Warning Label 1 Insert Table >

To Insert a Table:

- Click on the **Insert Table** button **(1)** in the **Refine** tab.
- A form opens where you need to specify a **caption (2)** and an **identifier (3)** (e.g. 1.1).
The identifier will be automatically be completed with a prefix "Tab. " when the table is generated.
- Next, select a table file to import by clicking **Choose File...(4)**.
- The table must be in the .html format, which can be exported from e.g. LibreOffice by clicking File->SaveAs and selecting .html as filetype.
- Click **OK (5)** to confirm your selection.
- The table is now available in the system's clipboard and can be added to the **Refine Text** by right-clicking in the appropriate location and selecting *Paste*.

Example Table

An Imported Table:

Tab. 1

API Name	Webcam Selection	Webcam Listing	Capture of Still Images	Allows setting Resolution	License
Webcam Capture API A	Yes	No	Yes	Yes	Apache License 2.0 Or GNU-GPL Version 2
Webcam Capture API B	Yes	No	Yes	Yes	3-clause BSD License
Webcam Capture API C	Yes	Yes	Yes	Yes	The MIT License

Comparing different APIs (anonymized) regarding their compatibility with ManuMaker's proposed features



NOTICE

Do not import very large tables (especially regarding their height), if the manual is intended to be printable. For tall tables in printing-oriented projects, consider adding the table to a slide without a picture. The **Export Tool** will then load a template without an image for that slide, so there is more space, and the table is less likely to "leak" onto the next page.

Chapter 5:

Using the View Tab

**Intended
Readers:**

All users of the ManuMaker application

Author(s):

Daniel Lachmann

Publisher:

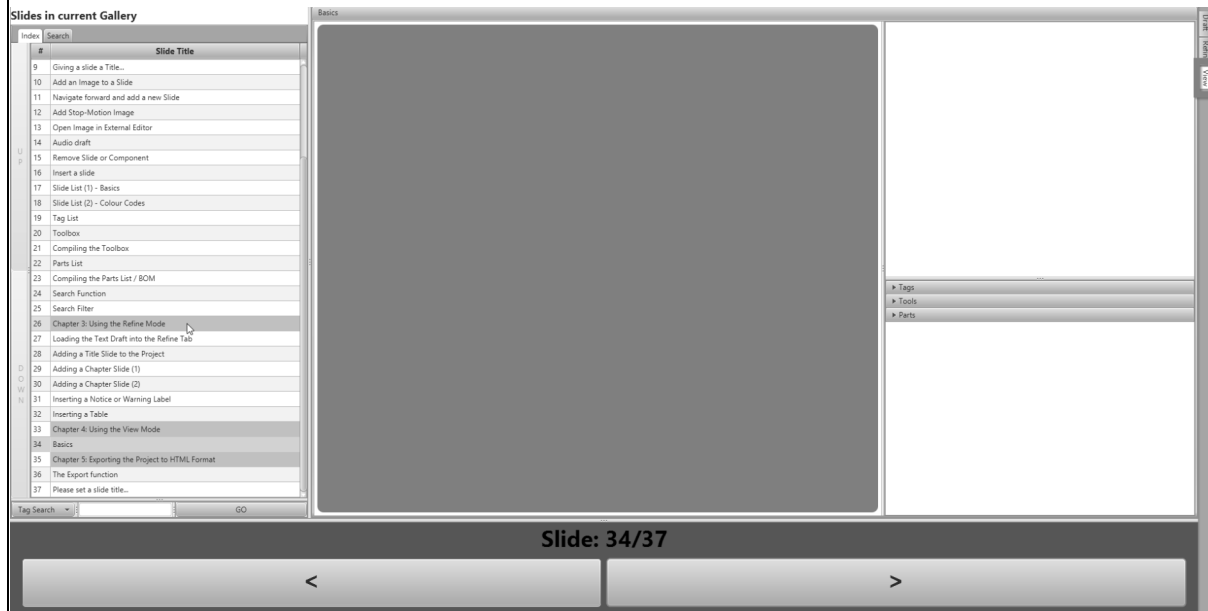
HAW Hamburg

Berliner Tor 21
20099 Hamburg
Germany

Contact:

Email:
manumaker.dev@gmail.com

Summary



Features of the View Tab:

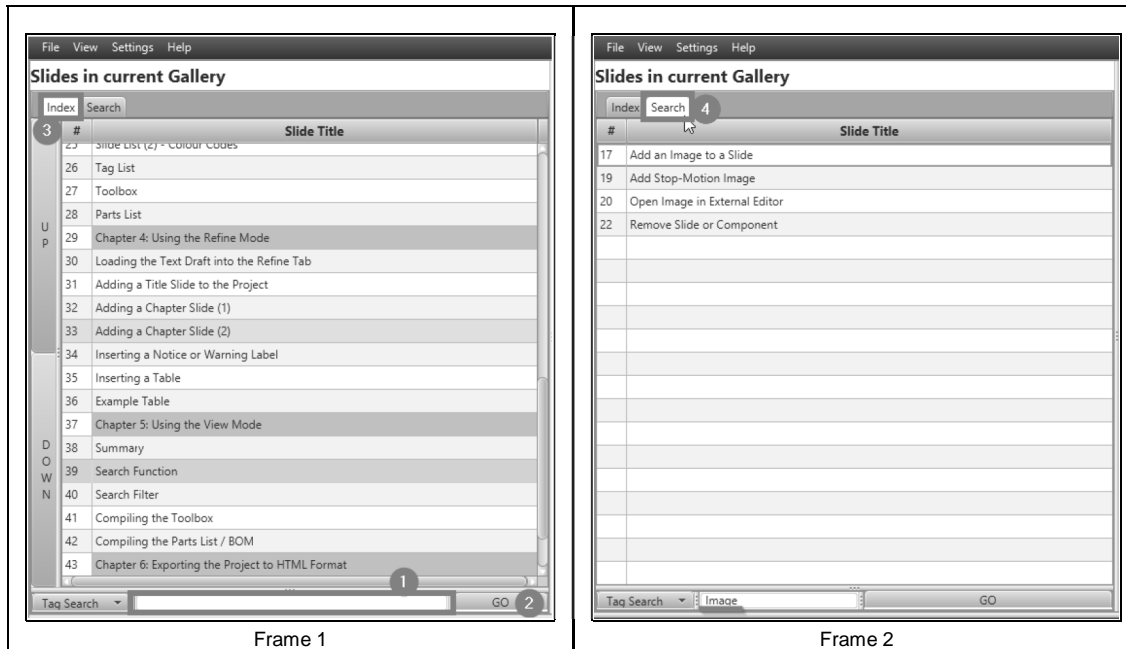
The **View** tab is intended for the readers of the documentation. It does not offer the user any editing features.

The functions which are available in the View tab are all related to information retrieval, such as:

- Linear navigation through the documentation via the **Forward-/Back-button** in the **main button bar**
- Direct navigation to a specific slide via a click on the respective item in the **slide list**
- Search function to find only slides which deal with a specific subject

If Stop-Motion pictures are cycling through too fast, go to *Settings->Application Settings* and then open the *General-Tab*. Here you can adjust the duration for which each frame is displayed in milliseconds (Accepted range is 250ms - 25000ms). Click on *Save* to apply the new settings and close the application settings dialog.

Search Function



Searching Tags, Tools and Parts:

The tags exist to make the documentation searchable by keywords.

To search for all slides related to a keyword:

- [Frame 1] Type the keyword into the **Search Field (1)** below the slide list
- [Frame 1] Click on **GO (2)**.
- Note that the search is case-insensitive.

If initially the **Index** tab (3) [Frame 1] of the slide list is selected, the **Search** tab (4) [Frame 2] is automatically opened and all slides matching the keyword or keywords are displayed.

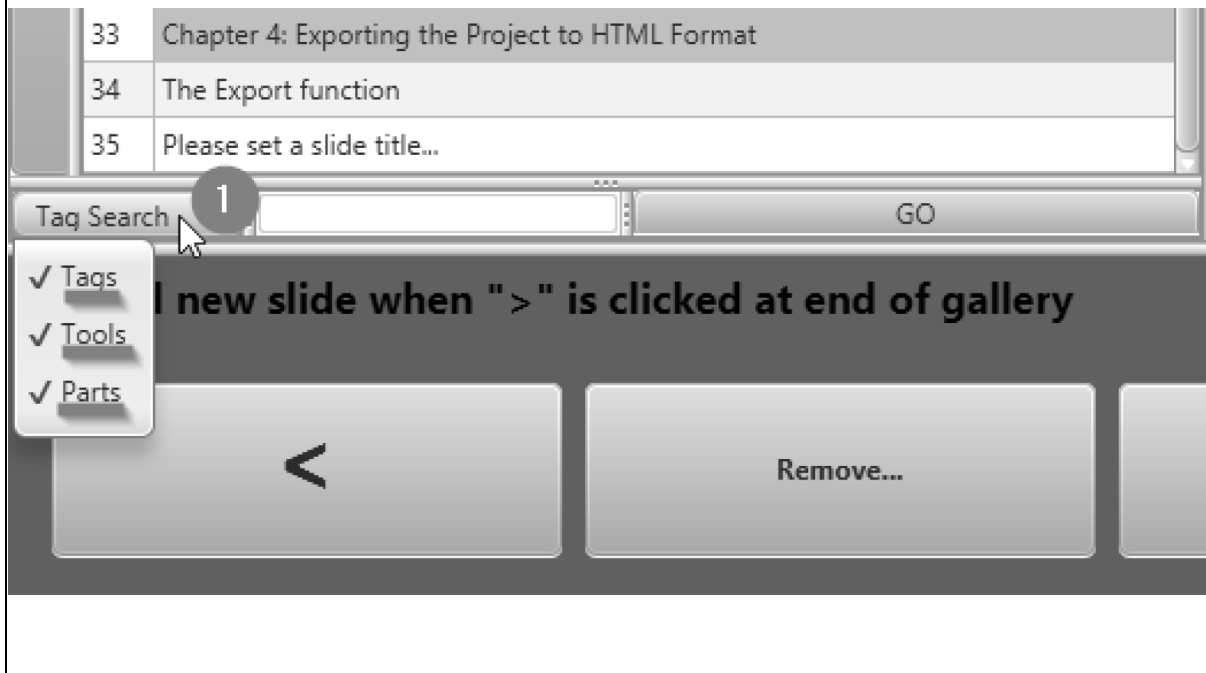
To search for multiple keywords, separate them with a semicolon, to navigate to a particular slide, left-click on it in the list of the **Search** tab.



HINT

- For a broad search, use short keywords, as the search returns words which **match** or **contain** the searchword.
- Searching for "valve; rocker;" finds all instances of " [...]valve[...]" and all instances of "[...]rocker[...]", while the search word "valve rocker assembly;" only finds exactly that term or terms which contain it.

Search Filters



The screenshot displays a software interface for searching through a list of slides. At the top, a table lists slide numbers and titles:

33	Chapter 4: Exporting the Project to HTML Format
34	The Export function
35	Please set a slide title...

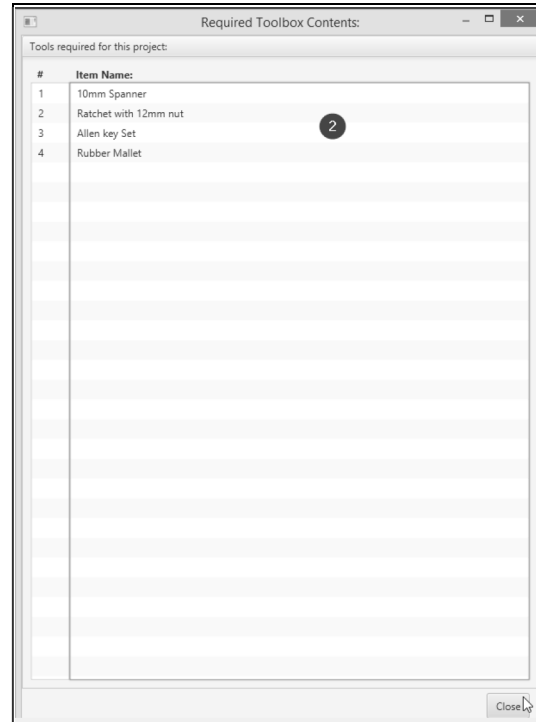
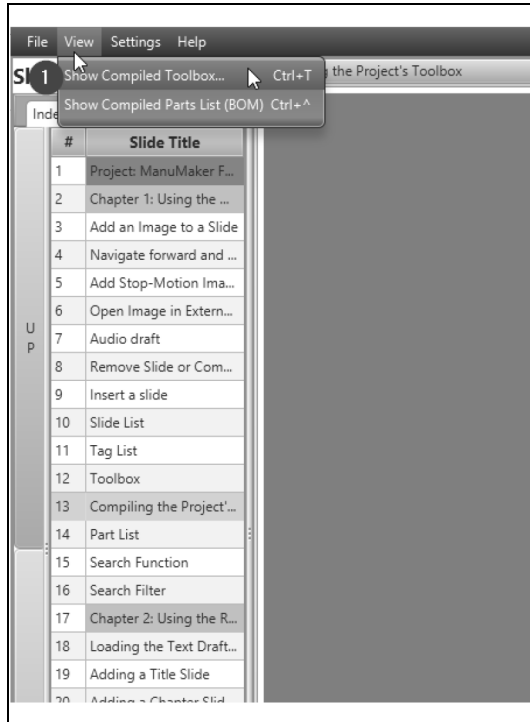
Below the table is a search bar with the text "Tag Search" on the left, a text input field, and a "GO" button on the right. A red circle with the number "1" is placed over the "Tag Search" label, and a mouse cursor points to it. A dropdown menu is open below the search bar, showing three categories with checkmarks: "Tags", "Tools", and "Parts". Below the search bar, a dark grey banner contains the text "A new slide when '>' is clicked at end of gallery". At the bottom, there are three buttons: a left-pointing chevron, a "Remove..." button, and a partially visible right-pointing chevron.

Search by category:

To search only for a specific type of keywords, e.g. parts or tools, click on the **Tag Search** dropdown menu (1) at the bottom left of the slide list, and select the keyword-categories you wish to search for.

By default, all categories are enabled.

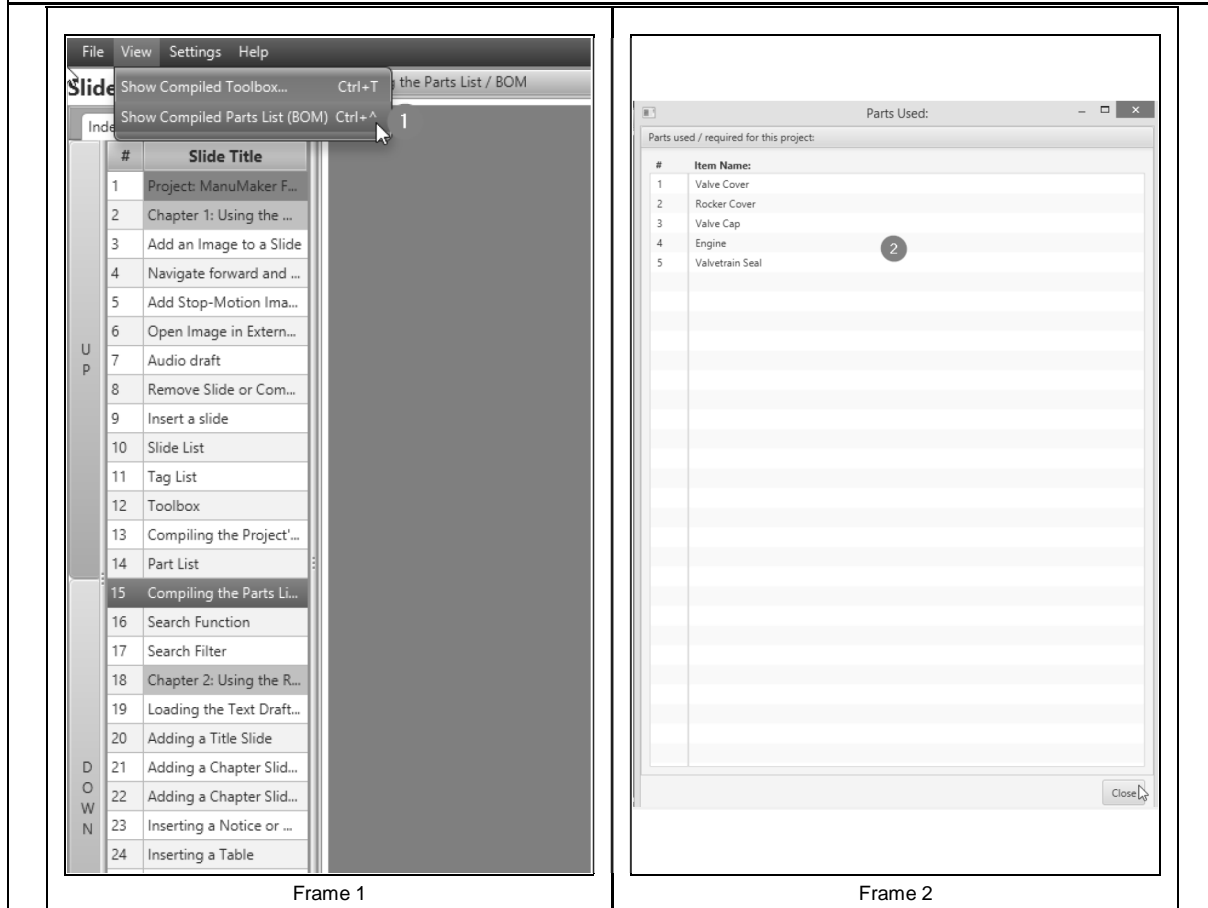
Compiling the Toolbox



List of all Tools used in the Project:

- To show the complete list of tools needed for the work described in this manual, click on **View** in the top menu-bar and and select **Show Compiled Toolbox... (1)**.
- The shortcut *CTRL + T* can also be used.
- The **Toolbox Window (2)** will open and the list of tools collected from all slides in the gallery will be displayed.

Compiling the Parts List / BOM



List of all parts used in the project:

1. [Frame 1] Click on **View** in the Menu Bar and select **Show Compiled Parts List (BOM) (1)** The shortcut *CTRL + P* can also be used
2. [Frame 2] The **BOM Window (2)** will open and display all the required parts from all slides.

Chapter 6:

Exporting the Project to Printable HTML Format

Intended Readers:

All users of the ManuMaker application

Please Note:

ManuMaker has been made with the intention to create a documentation format which takes advantage the digital medium.

Yet, for the moment, the need to print documentation on paper is still there, so adding an export feature to make the manuals printable was mandatory, from the perspective of usability.

However, not all documentation created with ManuMaker is automatically printable.

This chapter will provide some guidelines on how to create printable documentation in ManuMaker.

Author(s):

Daniel Lachmann

Publisher:

HAW Hamburg

Berliner Tor 21
20099 Hamburg
Germany

Contact:

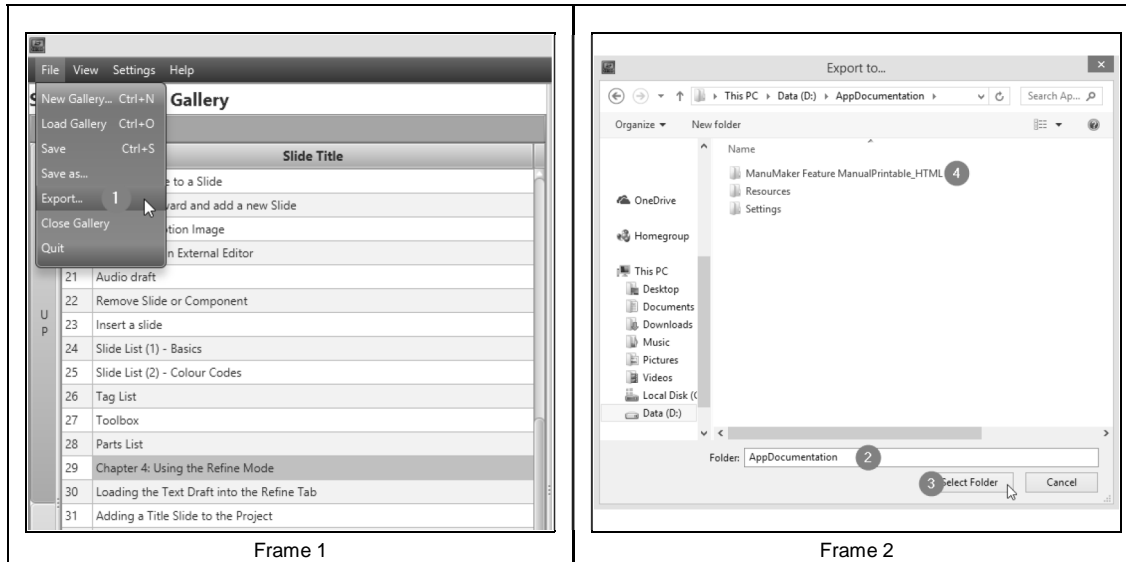
Email:
manumaker.dev@gmail.com

Fundamentals of the Export Function

Some Information on ManuMaker's Export to printable HTML - Function:

- Exporting the native ManuMaker gallery format to a printable form is recognized as a required bridge to the past. Printing is still a requirement in many cases, and it will still be for a while.
- One idea behind this project was to reduce paper waste. Therefore, if you have the opportunity to access a documentation on a portable device (e.g. Tablet PC), please consider that option. ManuMaker's development will go on, and there will be an effort to make the user-interface layout switchable between touchscreen and mouse-mode.
- The most commonly recognized format for portable documents is undoubtedly PDF. The export function does not use it for the following reasons:
 - The rich text editor in JavaFX (as found in ManuMaker's Refine tab) is HTML based and using existing APIs I would have needed two conversion steps to extract plain text from it.
 - The result would then have been fed into e.g. The Apache Software Foundation's PDFBox.
 - Since the HTML in ManuMaker's editor is at risk of not being very clean (start / end tags may be missing or misplaced through editing in the Refine tab's WYSIWYG Editor and also through import of title slides, chapter slides, tables and warning labels), it was deemed likely that these conversion API's would not easily be convinced to produce usable results.
- The chosen solution: Export to HTML. This was considered a good idea, because:
 - HTML is a very easy to use format, since it is completely open and the files can be opened and edited in plaintext-editors
 - As mentioned previously, ManuMaker's rich-text capabilities are HTML based (due to the JavaFX framework used)
 - Everybody with a computer has access to a web-browser, and an internet connection is not required to view the HTML-based documentation on the filesystem
 - Browsers offer a "print page" function, which charmingly chops the displayed web-page into segments that fit on A4 pages with no regard for aesthetics or continuity:
 - A solution for the chopping: Page breaks are available in HTML
 - The problem with with that solution: Page breaks do not limit the size of a page's content vertically, they only keep the *following* information off the current page
 - This works fine, if a page's content is *shorter* than the A4 Page
Otherwise it will still leak across pages, pushing the page-break tag to the next, partially filled page.
 - This way the next slide will still start on it's own page - so at least the error does not propagate
- Despite the noted difficulties: Page numbering is inserted into the page-template by the **Export Tool**. If slide 9 spans 1.5 pages, slide 10 will still be labeled with the correct ordinal (10), despite technically being the eleventh sheet of paper in the manual. There will just be an un-numbered "rougue" page in-between the numbered ones.
- With some discipline in the documentation process, pages can be kept short enough not to leak out of the A4-bounds. To check if a project is printable without page-leaks:
 - Export the project to HTML
 - Open the exported HTML documentation in a web-browser of your choice
 - Preview the page-boundaries by using the print-preview function
 - If a page is too long, consider splitting it's information across two slides, or shorten it by reformatting the text
 - Re-export and verify again

Exporting a Documentation Project



To export a Project:

- [Frame 2] Click on *File*->*Export* (1)
- [Frame 2] Select the parent directory (2)
In this example, the directory of the ManuMaker project is chosen
- [Frame 2] Click *Select Folder* (3) to export the project.
- The exported documentation will be saved in its own folder in the selected parent directory.
- [Frame 2] The folder name consists of the project title as set in the project settings, appended by "Printable_HTML" (4) .
- This folder is portable, since it contains all information needed for the HTML-based manual (images, logo, etc. are copied into the Printable_HTML folder)

Guidelines for Printable Documentation (1): Export Templates

Transferring the Gallery's Information to HTML:

To understand the reasons behind the following instructions, the underlying structure of the exported slides in HTML form must be understood.

- During the export process, all the data of a slide is transferred into a HTML template.
- Every slide type (title, chapter, instructional with picture(s), text-only) has its own template file, which is located in the installation directory under *app/resources/printable_html/...*
- The templates (all HTML templates ManuMaker uses for display, print and warnings) are **intended** to be replaced, if a different design is needed to e.g. conform to a company's style guide.
To replace them, please refer to the source code of the respective base files in the directory named above and make sure any replacements contain the same root elements
- Templates can easily be created by using table calculation software for defining the colour, position and size of background and text, use the export / save function which generates the most plain html
- Before you alter any of the html template files, make sure you create a backup first.
- Each template defines the visual appearance:
 - Minimum width of the page (defined by giving the template a white-coloured frame which the viewer doesn't see. otherwise the empty spaces would be shrunk together as HTML does not preserve whitespaces and "regular" newlines)
 - Fontsize and location of different text elements:
 - This is achieved through the use of placeholder words
 - When the export-version of a slide is generated, the html template is loaded (as a Java String, using UTF-8 encoding) and all the placeholders are replaced by the relevant slide data via *String.replace* operations
 - If a placeholder appears twice, both instances are replaced with the same data
 - If a placeholder cannot be found, no exception will be thrown and the corresponding data will simply not appear in the exported manual
 - Frames around elements and colour-scheme for the page
- The instructional page-templates all consist of the following elements:
 - Header, which contains the slide number (All slides including chapter and title slide are counted, counting starts at **1**)
 - Title of the slide printed in large, bold letters
 - Information data (image, text, or both), which usually makes up the largest portion of the slide
 - The placeholder for the information data is followed by a pagebreak (Not part of the slide-specific template, every slide is loaded into the page-template *TMP_Page.html*)
- There are two key aspects to be considered when making a slide printable:
 - Keep contents to a minimum width (which is not a concern unless the template-files are modified)
 - Keep the amount of information small enough to fit on its page, without pushing the page-break over the edge to the next page.

Guidelines for Printable Documentation (2): Page Size

The length of a page's content is determined by up to three attributes:

- Height of the image (on slides which contain an image)
- Number of images, for slides which contain **Stop Motion Pictures**, as all of the frames are printed on the respective page
- Length of the refined text

For regular **Still Images**:

- The images are fit to the width of the template's frame, retaining a few cm of whitespace to either side to facilitate binding and printing
- Aspect ratio will be retained
- Avoid tall, narrow pictures (screen-captures) in projects intended for printing, since they will be stretched to about 15cm width and a corresponding height
- Solutions for cases where a tall, narrow image format cannot be avoided and there is no space for the text:
 - Name two slides with the same title, and give them index numbers (As done for the "Adding a Chapter Slide"-slides in the Chapter "Using the Refine Tab")
 - Insert *only* the picture in the first slide, and check that it doesn't leak across pages in print mode
 - If it still leaks: add whitespace on both sides of the image, to make its aspect ratio more manageable
 - See the pages about the Project settings in the chapter *Settings* for reference, where this trick has been applied
 - Insert *only* the text into the second slide
- Above solution can also be applied for stop-motion pictures with many frames, which may fill a whole page by themselves

For **Stop Motion Pictures**, the same basic rules apply. The differences:

- Each image is printed in its own frame, with a subtitle denoting the index of every image
- The images are fit to the width of their respective sub-frames (ca. 6cm wide)
- Starting at the first frames, two frames are loaded in a dual template which presents them side-by-side
- If the number of frames is uneven, the last frame will be presented in a single frame centered below the others (same size as individual dual frames)
- Again: tall, narrow images take up a lot of vertical space, but are okay to use if sufficient space is available (See Slide about the *Search Functions* in chapter *Using the View Mode*)
- Images with an aspect ratio around 1:1 are a good way to get as many slides on a page while preserving readability
 - Please note that capturing an entire screen, annotating it with arrows and then shrinking it to 6cm width is likely to lose its informational value
 - It is therefore recommended, to only use small sections of the screen or very close-up images with the camera for printable documentation
- Short, wide images are also discouraged for Stop-Motion
- Rule of thumb: Only use stop-motion images for printing if the answer to the question "If this picture is reduced to 6cm width, can the reader still discern its use and read e.g. position numbers and arrows?" is "Yes." for all contained frames

For the **Annotation Text**, the number of lines is the deciding factor:

- Manual newlines will be preserved, and need to be used to create whitespace between enumerations and sentences
- Text is wrapped automatically when a line is longer than the page-width, so the number of lines visible in ManuMaker's annotation text box may not be as many as in the exported documentation
- It is recommended to separate an overly long annotation-text into two or more subtopics of equal or similar hierarchical level, and distribute those across the same number of slides
- Inserting some whitespace is more helpful to the reader than keeping the pagecount low
- **NOTE:** If page leaks occur for no visible reason (If the print-version contains blank pages):
 - Check the preceding slide for rogue newlines after the last paragraph / warning label
 - Delete all blanks up to the last letter / boundary of the last warning label

Glossary

API Application Programming Interface. 52, 54, 55

application logic Coded functionality of a program which is not visible to the user, but which is responsible for processing information (e.g. input from a GUI) and passing it on (e.g. to a GUI) . . . 53

BOM Bill Of Materials. 18

CMS Content Management System. 11, 14

FXML XML-variant used by JavaFX to describe GUIs. 52–54

GUI Graphical User Interface. 8, 10, 41–43, 46, 51–54

IDE Integrated Development Environment. 53

IDE Integrated Development Environment. An IDE combines all tools necessary to write a software program (Editor, Compiler, ...). 51

JavaFX Java toolkit for GUI applications. 52, 53

JavaFX Scene Graph The underlying structure of the JavaFX GUI elements. 52

JRE Java Runtime Environment. 51

Oracle Owner and publisher of the standard Java platform. 51–53

OS Operating System of a Computer. 51

SceneBuilder 2.0 WYSIWYG GUI design tool provided for JavaFX. 8, 52–54

UI User Interface. 42

WORA Write Once, Run Anywhere. 51

WYSIWYG "What You See Is What You Get": The notion of designing a UI by visually arranging interface components on a mockup application window, which then gets translated to code in the underlying language. 52, 53

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, September 2, 2016

Ort, Datum

Unterschrift