Fachhochschule Aachen

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

How can Q.wiki better convey its structure?

Seminarthesis

Marine Raimbault
Matriculation Number: 3643813

Faculty of
Medical Engineering and Technomathematics
Study Program: Applied Mathematics and Computer Science (dual) B.Sc.

1. Examiner: Prof. Dr. rer. nat. Philipp Rohde
2. Examiner: Pascal Brunner, M.Sc.

Aachen, June 2025

Contents

1 Introduction

In today's complex world, human work rarely occurs in isolation. For instance, building something as complex as a car involves the work of many individuals with different skills and roles. Collaboration has become crucial to today's productivity. The goal of every company is to produce fulfilled clients. For that, there is a need to deliver quality (value) and to constantly improve the work that is repeatedly done (processes). Therefore, for collaborating, companies must have a way to store, retrieve, and improve (i.e., manage) their knowledge, which is key to ensuring that individuals can work together to produce great work sustainably.

Q.wiki, an interactive management system developed by Modell Aachen GmbH, aims to address the need for managing shared knowledge by providing a wiki system [1] as a digital environment where users can share, retrieve, and improve information about their company's processes in one central location.

We want the process of working with Q.wiki to be an enriching and productive experience. However, we believe there is an issue with how the software conveys Q.wiki's system structure, especially during onboarding. The current structure and navigation of Q.wiki are challenging for first-time users, which is a problem since a poor understanding of the system prevents new users from quickly utilizing Q.wiki to its optimal potential.

Exploring the root of this problem, we realized that this is not only a challenge for first-time users but a more global issue affecting all its users: Q.wiki does not yet optimally convey its structure in a human-centered way. Therefore, we will explore in this thesis: *how can Q.wiki better **convey** its **structure***? This problem has two components: the verb *to convey* and the noun *structure*. The Oxford dictionary defines the verb **to convey** as transporting something to a place [2]. In our case, it refers to the fact that Q.wiki already has a structure, but we want to present it to the user more effectively. Secondly, **structure** is defined as *the arrangement of and relations between the parts or elements of something complex* [3]. This highlights two structural components: elements —the building blocks —and the relations between different elements, which one can think of as the cement between the bricks of a house—the subject under study, in our case, Q.wiki.

What does conveying structure mean in the context of Q.wiki? To answer this question, we will first analyze the building blocks of Q.wiki (processes) and see what is involved in conveying information in software (cognitive load as an issue, human-centered design as a solution) by drawing upon previous work (Section 2: Theoretical Background), before specifying our problem (learning more about our current and desired state through user observation and expert interviews) through an empirical investigation (Section 3: Empirical Investigation).

[1] More about wiki systems: https://en.wikipedia.org/wiki/Wiki_software
[2] https://www.oed.com/search/dictionary/?scope=Entries&q=convey
[3] https://www.oed.com/

2 Theoretical Background

This section will explore the main components of the problem: first, Q.wiki as a process-oriented Quality Management System; then, how to convey information in software, which leads to the concept of cognitive load; and finally, design strategies to solve our problem.

2.1 Quality Management Systems (QMS)

We have already stated in the introduction (Section 1) that the goal of every company is **sustained success**. Most organizations agree that Quality Management Systems (QMS) help them continuously improve their productivity and, therefore, achieve this primary goal [1]. Consequently, we need to define what a QMS is. Firstly, **Quality** is the degree to which a product or service satisfies its client,[1], and a practical synonym for it is value-generation [2]. Secondly, while a formal definition for **Management Systems** exists,[2] Modell Aachen's co-founder, Dr. Carsten Behrens, finds it unpractical and defines a Management System as the sum of all game rules (written or unwritten, explicit or implicit rules) of a company, regardless of whether it is in a digital system or not.[3] QMS are therefore both the software tools and the rules that enable companies to continuously improve their products and methods to fulfill and exceed customers' expectations.[4]

This section will first establish the need for a specific subset of QMS, known as process-oriented management systems, by understanding how processes are crucial to companies' productive work, and then introduce Q.wiki as a software solution for such a process-oriented system, before finally examining how Q.wiki implements it by analyzing a typical user workflow.

2.1.1 Process-oriented Management Systems

We will now understand the need for a system for managing **processes** with people's knowledge, whereby processes are a succession of actions that is repetitively done by a group of employees inside a company in contrast to projects, which are usually done only once.[5] The history of quality management teaches us that businesses wanting to sustain success require optimizing workflows (processes) while engaging employees (people) to improve them continuously.

To understand why processes are important to Quality Management, we need to learn about the recent historical evolution of human quality control practices. For most of its history, quality control involved removing defective goods to ensure the end product was not of bad quality, but a significant shift occurred in the 20th century when Shewhart proposed focusing

[1] https://asq.org/quality-resources/history-of-quality
[2] See the formal definition in the ISO 9000 norm https://www.iso.org/obp/ui/#iso:std:iso:9000
[3] Video in Carsten Corner 1 Interaktive Managementsysteme – eine Revolution
[4] https://www.iso.org/quality-management
[5] Difference between project and processes

on analyzing an organization's production processes rather than solely controlling the finished product, as this approach became more effective in restoring quality at a lower cost and with less time consumption.[6] Deming then applied Shewhart's ideas to improve the quality of US weapons in World War II before helping Japanese companies, most notably Toyota, become more competitive, having great success in doing so.[7] Therefore, Quality Management focuses on processes because they are instrumental in managing both time and money resources efficiently.

Processes are the core of the field called Business Process Management, which aims to improve companies' business processes. According to the most quoted book on the subject [3], it is important to prioritize processes in order to only give them the right amount of attention in relation to the benefit they provide. A way to do so described in the book is to differentiate three types of processes: management, core, and supportive processes. **Management Processes** (e.g., company objectives, vision) provide direction for the company but do not directly generate revenue. **Core processes** are those processes starting and ending with the customers; they are the processes bringing revenue, for example, processes related to customer acquisition. Lastly, processes like accounting, which are needed for the survival of the company but do not directly bring money, are called **Supportive processes**.

We just saw what types of processes can be differentiated and why they are important; however, we still need to understand how people's knowledge comes into play in the process-oriented system Q.wiki.

2.1.2 Aachener Modell and Q.wiki as a solution: empowering employees

In a video,[8] Modell Aachen co-founder Carsten Behrens explains that Q.wiki was created because the founders recognized that quality documentation was mainly used as a proof document and not used by the employees as an up-to-date useful documentation to be improved, and the root problem was that only Quality Management Professional (QMB) where writing in it, meaning the barrier to continuously improve it was high, making the documentation go out of date by design. The goal to achieve a better QMS is therefore not a unilateral management system documentation (top-down approach, outdated). Still, multilateral management system communication (all employees communicate their game rules and best practices), the Modell Aachen founders decided to start with a wiki software to address the need for such a system, therefore starting Q.wiki's development. The core idea of Modell Aachen is therefore to represent processes without losing the big picture: every employee is part of the whole, and every process or work action only makes sense in the context of the bigger system.

[6] https://asq.org/quality-resources/history-of-quality
[7] https://deming.org/toyotas-management-history/
[8] Video: Carstens Corner Episode 1 - Interactive Management Systems

Q.wiki aims to structure business knowledge by focusing on its processes. Processes in Q.wiki are represented as individual wiki pages with a clear responsible employee and clear version control, complying with the recommendations of the ISO 9000 norms. One can think of Q.wiki processes as being in a two-dimensional space: Vertically, there are four zoom levels, from the big picture (process landscape) to the smallest detail (Work instruction). These are the four vertical layers (see Figure 1), which Q.wiki shows with breadcrumbs [9]. The other dimension can be imagined when focusing on processes: how are processes linked between them, on the same level of detail? Currently, processes on the same layer are manually linked with simple links, and if companies want to visualize the relations between them better, they can try to do so with a simple table, or add an image and manually add links to it. If they want a more complex way, they have to invent their visualization method; the system does not provide them with an intuitive way to do so.

2.1.3 Q.wiki's User Workflow

The first page a new user interacts with in Q.wiki is the **process landscape** (see Figure 6). It aims to give a high-level overview of one's organization's core processes. By clicking on one element of this process landscape, the user navigates to the **process overview** (see Figure 7), a list of all related processes to the core process. By clicking on one of these processes, a **process description** (see Figure 2) opens. From there, the user sees the steps for executing the process, who is responsible for it, and what the input and output of the process are. Sometimes, in the process description, they are more detailed, and the user can open a **work instruction**. The difference between a process and a work instruction is that processes take longer and can involve multiple people working at different locations. In contrast, one person can execute a work instruction at a single location within a unit of time.

Q.wiki producers understand work instructions (no figure) or info page (see Figure 8) as leaves of the tree, or the lowest part of a pyramid, the highest grade of detail, or the most zoomed-in part. One can understand these four pages as a 4-layer hierarchy that creates a top-down structure resembling a tree or pyramid, guiding users from strategic views to operational tasks (see Figure 1). However, the user does not see this directly; the pyramid hierarchy is only a mental model taught to the user during onboarding. One can see that Figures 7 to 8 look similar: the software does not visually convey the difference between them.

Q.wiki relies on templates for creating different pages. Under the page, we mean having the choice of creating either a Work Instruction (German: Arbeitsanweisung), an Info Page (German: Infoseite), a process description (German: Prozessbeschreibung), or a process overview (German: Prozessübersicht). See Figure 17. The templates are customizable per tenant (a tenant is a customer company). A surprising aspect of these templates is that the

[9] More on breadcrumbs: https://en.wikipedia.org/wiki/Breadcrumb_navigation

pages can be created everywhere, meaning theoretically, a user can create a work description (conceptually belonging to the lowest layer, the layer 4) under the process landscape (layer 1), which does not make sense according to the Q.wiki concept but is not actively undermined by the software.

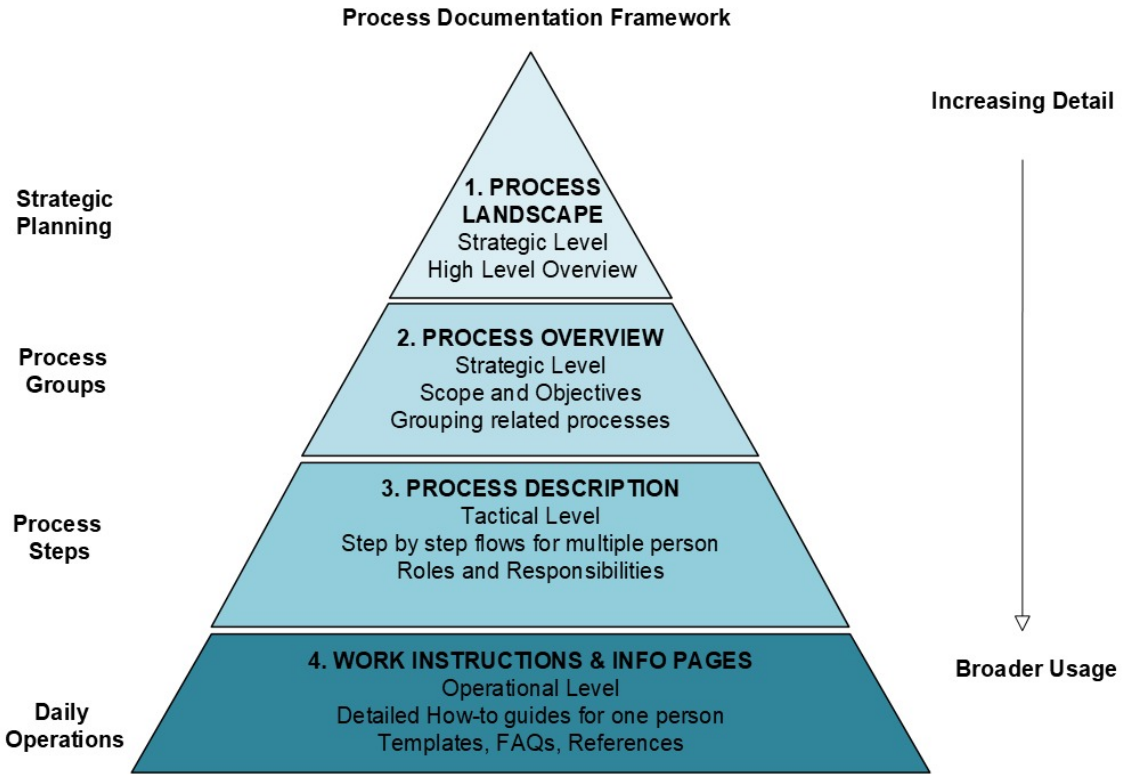

Figure 1: *Four-layer pyramid model of Q.wiki showing hierarchical levels of detail. Figure created by Marine Raimbault based on the 4-Layer Concept from Modell Aachen GmbH.*

Value is created in Q.wiki when processes are **created**, **used (read)**, and **updated**. We can think of user actions as a three-step process, similar to the data scientist's Extract, Transform, Load (ETL) process.[10] First, the user gathers all the information required (Extract: Discover or Find related processes), then the user processes all this information (Transformed: reading related processes, optionally cleaning it by editing it), and finally the user loads the data (assembling all relevant parts to create a new process or updating an existing one).

Let's analyze the steps required before producing value in Q.wiki for the three main use cases. We take the minimum number of clicks needed as a first approximation measure of effort, as this paper does [4]. To read a process, a user must first find it. We differentiate between the discovery of a process (when the user is unaware of the process and therefore does not know where to find it) and the actual finding of a process. To access a given process, we observed that the new user must perform at least two to three clicks before accessing the process (process landscape $\rightarrow$ process overview $\rightarrow$ process description or work instruction). Therefore, in the best case, two to three clicks are required to discover a process. Supposing

[10] https://www.ibm.com/think/topics/etl

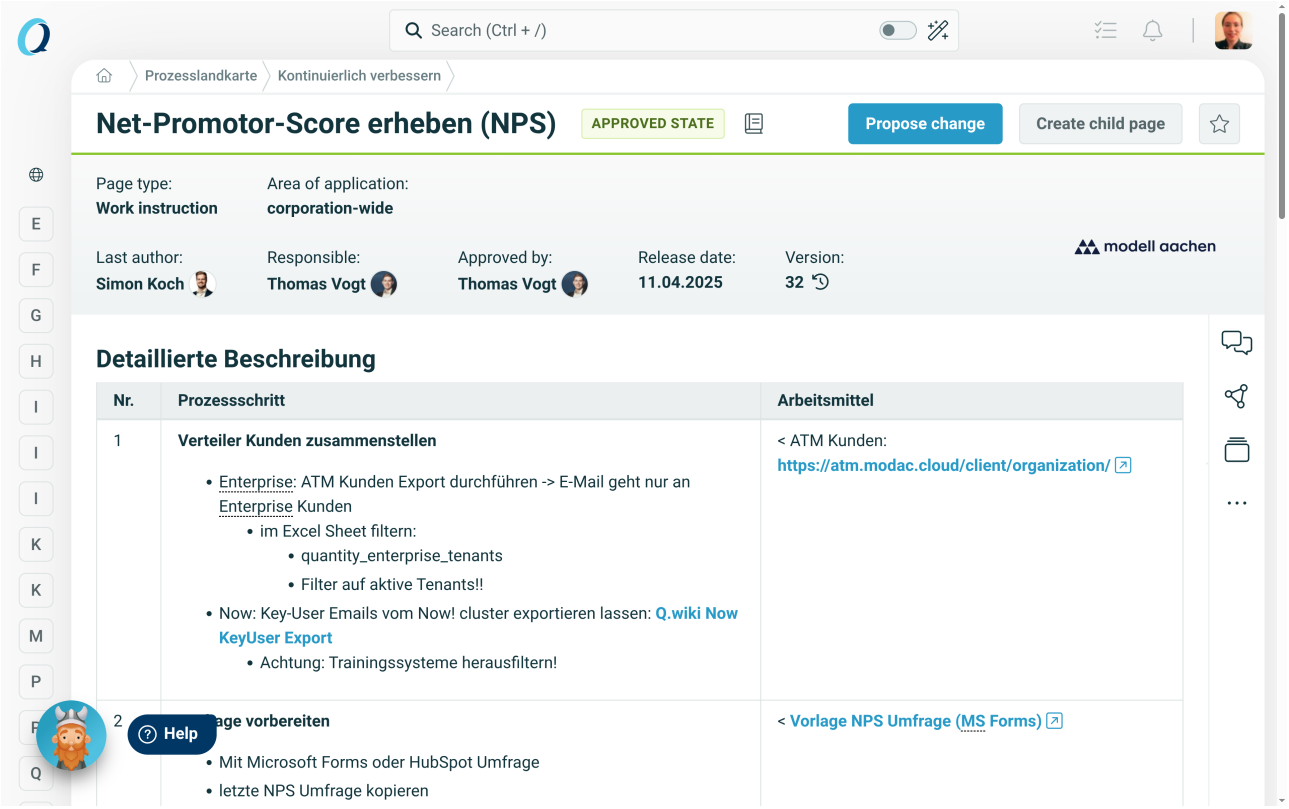

Figure 2: *A Process Description in Q.wiki*

the user already knows the process, they will find it through the search bar and need two clicks (one click in the search bar and one click to select it from the results) to locate a known process. Reproducing this thinking for the other events we obtain:

- **Discovering** a process: minimum two-three clicks
- **Finding** a known process: minimum two clicks
- **Creating** or documenting a process: minimum five clicks (searching for related processes plus three clicks: creating, saving the draft, approving it)
- **Updating** or improving a process: minimum five to seven clicks because we sum the clicks for finding or discovering a process, plus three clicks: one for editing, one for saving, one for submitting a proposal, this does not count the clicks for looking for related processes

We just saw the steps required for the main use cases of Q.wiki. We will now understand what makes software usable and how to reduce the physical and mental effort required to produce value with Q.wiki.

2.2 Cognitive Load in Learning and Problem Solving

In this section, we establish poor learnability as a key usability issue and Cognitive Load Theory as a way to understand it. We will then examine the strategies available to measure cognitive load before determining that visualization techniques, or a better user interface, are

key to reducing cognitive load in human-computer interfaces.

2.2.1 Learnability in Software is Key to Producing Successful Users

Understanding how humans process and learn information is vital to designing usable software systems. One aspect of this is the time required to learn to use a software system referred to as *Learnability* [5], whereby we understand learning as the process by which users acquire insight from the information provided by the system to make productive use of it. Learnability, as Forsey [5] puts it, is a fundamental measure of software *Usability*.[11]

Sierra [6] emphasizes the importance of user mastery for satisfaction and productivity. Assuming that sustained success is the goal of every company, she examines the common denominator of profitable companies. According to her, the key to sustained success is creating successful users, as they are more likely to recommend the product. She argues that when users achieve proud results, they experience a boost in self-esteem, which naturally motivates them to share their success by recommending the product. She claims that empowering users to become product experts is central to achieving sustained business success.

Theoretically, the most effective software would not require effort or learning from users. Yet, increasing requirements for advanced functionality and the growing number of tools we use often make achieving this level of perfection in Learnability difficult. Every day, a typical knowledge worker uses 11 distinct software apps [7], yet only a handful develop actual expertise. Thus, making an interface easy and straightforward to use is a challenge. Therefore, software producers must understand what aspects affect the obstacle to ease of use: the Learnability of software.

We now consider the key reason learning new software interfaces remains challenging for users: cognitive load.

2.2.2 Cognitive Load as a Constraint in Human-Computer Interaction

Empirical studies have shown that the primary barrier to effective learning and problem-solving is cognitive load [8], defined as the mental effort required by individuals when processing information. Under the theory of constraints [9], system optimization should concentrate on the most binding constraint—in this case, cognitive load. Studies of problem-solving have provided evidence that it is experts' sensitivity to recognizable, familiar patterns in problems that marks the difference between experts and novices [8]. These patterns, known as schemas, are mental representations of standard problem types and associated solution

[11] Usability, according to the ISO 9241:2018 norm of Ergonomics of human-system interaction, is *The extent to which specified users can use a system, product, or service to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use*

strategies. The learning process aims to recognize and assimilate these patterns into long-term memory. But what part of the mental effort is necessary for the long-term assimilation of schemas?

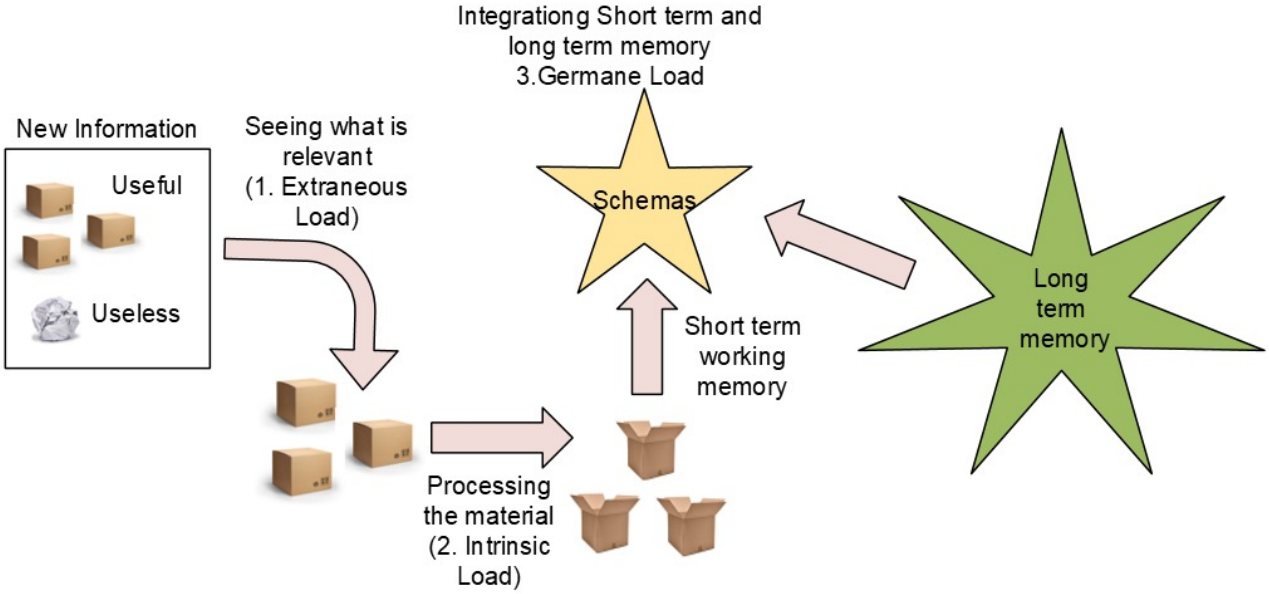

Figure 3: *Cognitive Load has three components: extraneous, intrinsic, and germane. Figure created by Marine Raimbault*

Cognitive Load Theory (CLT), as proposed by Sweller [8] and further elaborated upon [10], identifies three types of cognitive load, which we illustrate in Figure 3:

- **Intrinsic load**: Related to the task's inherent complexity. Reducing it involves simplifying the material.
- **Extraneous load**: Imposed by poor design elements that don't contribute to learning.
- **Germane load**: The mental effort to process, construct, and automate specific patterns called schemas. It involves processing the new information and integrating it with previous learning.

Therefore, the goal is to reduce intrinsic and extraneous load and focus mental effort on the value-creating germane load. Software producers can achieve this by reducing the difficulty of the problem (intrinsic load) and removing everything that does not contribute to learning (extraneous load).

We first thought we needed to measure cognitive load to understand its relevance to our problem, and found various possible ways of measuring cognitive load in the research. However, we realized that these methods are not useful for exploring our current problem. That is why we moved our understanding of those methods to the Appendix. Indeed, while these methods for measuring cognitive load help evaluate different software interfaces, they do not directly teach us how to develop better interfaces. Having observed how scientists assess cognitive load, we will explore practical methods to mitigate it.

2.2.3 Offloading mental effort through Visualization: Cognitive and Perceptual Principles

In this subsection, we will justify our focus on software design, particularly the visual aspect of our software. The eyes are not the only sensory organs through which one can interact with the world. We have four additional body sensors: the ears to hear, the skin to touch, the nose to smell, and the tongue to taste. Although research toward blending these experiences, coined as multimodal Human-Computer Interface (HCI), shows promise for enriching human-computer interfaces to more closely mirror real-world interactions [11], not all of these senses are equally relevant to the core information necessary for learning and problem-solving.

Considering cognitive load theory (CLT), Mayer [10] proposed a cognitive theory of multimedia learning based on the dual channel model. This model asserts that only two channels are relevant in multimedia learning: those that process words and those that process pictures. (1) Raw images (and sounds) are first *perceived* by our eyes (and ears), (2) *organized* into verbal and pictorial models in the working memory, (3) and then *integrated* with prior knowledge stored in long-term memory. Because of the primarily visual and not auditory nature of Q.wiki, we only focus on data from our eyes. Still, we acknowledge that Mayer also speaks of the ears as a means to perceive words.

Information visualization harnesses human visual perception to convey meaning with minimal cognitive load in the working memory [12]. Effective visualization externalizes mental processes and allows users to accomplish more cognitive work in less time [12]. Visualizations reduce cognitive load most effectively when the images reflect how users naturally organize and navigate information [13]. The goal is, therefore, to align software interfaces with natural mental models. The following section examines what those mental models are and how to design our software accordingly.

2.3 Designing Strategy to Reduce Cognitive Load

We aim to develop a viable solution for our problem. For this, we will combine two design methods: first, we identify the iterative cycle of human-driven design as a scientific method for solving user interface problems. Second, information architecture enables us to understand the information structure in software by using different visual tools, such as diagrams or sitemap representations, based on the information we want to extract from it.

2.3.1 Human-Centered Design

Designing software interfaces with a low cognitive load requires a profound understanding of Human-Centered Design (HCD) principles. As Norman puts it, good design is about *finding the right problem and fulfilling human needs* [13]. To support this process, he proposes the

Human-Centered Design iterative cycle, consisting of four steps:

1. Observation
2. Idea generation (ideation)
3. Prototyping
4. Testing

Designers can apply evidence-based strategies to lower users' cognitive effort within the ideation and prototyping phases. Common approaches include:

1. **Constraints** — Limiting options, paradoxically enabling humans to do more [13].
2. **Recognition over recall** — minimizing memory load by displaying visible choices [13].
3. **Clear signifiers** — guiding user attention by using *design cues* (Norman: *signifiers*) to indicate *possibilities* (Norman: *affordances*) [13].
4. **Chunking information** — grouping content into small units, typically around 7 (5-9) items [14].
5. **Proximity Principle** — Keeping related information physically close to where it's needed, rather than requiring users to search for it elsewhere [15].
6. **Clear Visual hierarchy** — making clear what is essential, so the user does not have to do it himself [13].
7. **Progressive disclosure** (also called Visual Scaffolding) — making related information easily accessible without overwhelming the main interface, often through expandable sections or hover states or revealing only necessary information at first, delaying advanced content until users are ready [5, 16].

These techniques are not isolated tricks, but part of a broader effort to align interface design with how humans process and learn information. They aim to reduce cognitive friction by externalizing complexity and aligning interaction flows with users' mental models. According to Norman, mental models are conceptual models formed through experience, training, and instruction. These models serve as guides to help achieve our goals and in understanding the world [13].

We know how to open a door, through pulling or pushing. However, Norman gives an example of a badly designed door having a door handle inviting to push when, in fact, to open it, the human has to slide it. In this example, the human mental model of a door helps us open all objects looking like doors, but confuses us when the object (here the badly designed door) implies a certain function (with the signifier: door handle) but behaves in a way that is contrary to intuition (sliding instead of pushing). Norman thus highlights the importance of using clear signifiers (e.g., a door handle inviting a sliding motion) that give the right cue (the door can slide) for triggering the right intuition (sliding motion to open the door).

Having established human-centered design principles and the importance of aligning software interfaces with users' mental models, we now turn to information architecture theory to understand the structure of information in software.

2.3.2 Information Architecture

Information Architecture (IA) refers to the organization and structure of information within digital systems. According to Rosenfeld et al. [17], IA states that the structure of information not only describes how users can find information, but also how users understand it. The core principle is that **complex digital systems require multiple views** to display different aspects to different audiences and different contexts.

According to the Nielsen Norman group, known as an authority on design,[12] Information architecture has three layers: the visible navigation, the IA structure, and lastly the taxonomies and metadata. First, **the visible navigation (Front stage)** consists of a series of user-facing elements like menus and breadcrumbs that communicate the user's location and the available pathways. The visible navigation should adhere to the user's mental model. Secondly, **the IA structure (Backstage)**, typically shown through a sitemap [13] is the map of all pages of a site or screens in an app and the relationships between them, and is not shown to the user. It is the full plan of the website. Lastly, **taxonomies and metadata** organize concepts through hierarchical classification systems. While sitemaps organize content, taxonomies organize the underlying concepts with emphasis on logical precision.

Rosenfelder differentiates four key aspects of IA: [17].

1. **Organization** systems consider different ways of grouping data like, for example, "exactly" with chronologically, geographically, or "ambiguously" like task-based or audience-based.
2. **Labelling** systems emphasize how important it is to give information a name that speaks the same language as our environment's users while reflecting its content, but scientists consider designing labels to be the most challenging aspect of IA.
3. **Navigation** systems help chart our course, determine our position, and find our way back. Rosenberg differentiates between global, local, and contextual systems.
4. Finally, **Search** is considered important for finding information, but Rosenberg states that not every system needs it.

Possible IA visualization methods include **diagrams** (components and the connections between them), **sitemaps** (showing the relationships between information elements such as pages and other content components), and **wireframes** (depicting how an individual page or template should look from an architectural perspective and content model).

[12] See video from the Nielsen Norman group at https://www.youtube.com/watch?v=v39z0JPeIc8
[13] Understanding Sitemaps

3 Empirical Investigation

3.1 Process to find the solution

This report aims to understand the problem, generate possible solutions, and evaluate those. For this, we will use the methodology of human-centered design we saw in 2.3.1 as a strategy to design in a way that reduces cognitive load, and choose for each step a method that fits our needs:

1. **Observation** observing the behavior of new users: what is the problem? The goal of this step is to generate hypotheses. We chose to use the tool Hotjar heatmap for this.
2. **Idea generation** through expert interview and use of the literature review.
3. **Prototyping** design with wireframe.
4. **Testing** testing different options with questionnaires on cognitive load with internal users.

3.1.1 Observing our users

In this step, we want to understand the questions: How do users interact with the Q.wiki software structure? Where do user spend their time? Where do they click? We chose Hotjar[1] for the observation step because Modell Aachen GmbH already uses it to track user behavior, making it a cost- and time-efficient option. On that website, we can track user actions, such as their mouse movements and clicks. The primary use of this tool is to identify what slows down the discovery of Q.wiki features. We utilized a feature of Hotjar called the heatmap, which displays a color gradient indicating where users click on the page and where the most frequently used areas are. We only use it in the demonstration version of Q.wiki, where QM users test the product before making a purchase decision, because where the users use the system for the first time and thus is the perfect environment to get the users' first impression.

By examining the various heatmaps generated on other pages of the Q.wiki demo website, several observations can be made (see Figures 9 to 12 and their captions). From these observations, we develop the following hypotheses:

- Homepage
  - The user is confused on the process landscape page because its main function is unclear (there are too many options).
  - The left sidebar to change between modules is confusing because it does not convey the structure of the main module: the process module with its four layers
  - There is a need for more **user assistance** instead of just letting the user create "child pages" on the homepage, which does not convey which part of the hierarchy the user creates the page in

[1] Hotjar

- Process Overview
  - The user does not spend a lot of time on the process overview, and either the current page fulfills its goal, or it is a useless additional step before seeing the processes, or **the page displays too much information**
  - The option to display related work instructions on the process page might be valuable to reduce the need for the user to navigate back and forth between the process description and work instructions
- Process Description
  - Since the user only focuses on a specific column, he might benefit from dynamic options to hide irrelevant parts of the page, for example, a possibility to hide parts of the process that he has already completed (checkbox option) according to the **constraint** design principle of letting the user less choice we named in Chapter 2.3.1
- Search page
  - Since the user does not seem to interact with the filtering options, the possible filtering options might be irrelevant to him.

Now that we have observed the behavior of our users, we need to interpret it in the light of the Model Aachen core business, culture, and product objectives. Are we solving the right problem? To know this, we need to involve our domain experts to extract the core underlying problems that are useful to solve, and create specifications for the solution that will really make the business move forward.

3.1.2 Hypothesizing the need to focus on process discovery

We need to focus on a limited subset of our problem because of the limited time scope. Thanks to a discussion with Q.wiki's expert in Product Management, Rico Wilmink, we identified a meaningful subset of the structure problem that aligns with the scope of this work. We focus on only one small part of the problem: **What process comes before, what process comes after**? Namely, how can we inform users of their location? Would the user understand their location if they randomly open a process or a work instruction? How to convey to him the bigger context?

What is the context of processes? Processes are part of an extensive network of processes: there are processes before (predecessors) and processes after (successors). Wilmink reports that our clients are frustrated because they must make a U-turn when navigating the processes. A u-turn in a web app can also be understood as coming back to the same page you were on. We define a u-turn as a vertical change in the layers: going from a different zoom layer, zooming, then unzooming, and finally zooming again to return to the process. We consider these wasteful because they do not lead to a good understanding of the relations between processes, and clients find them frustrating.

Let's revisit our initial problem: the process landscape represents a flow of processes. It is static, created using PowerPoint, and the user who wants to develop it must manually add the links. Then from then on, the user can create *child pages*, navigating between the vertical layers. They can be seen through breadcrumbs. However, the only way that related processes are linked on the same layer (horizontally) is through manually adding related processes as links in the wiki at the top (before, predecessors) and the bottom or end of the page (after, following processes), which is not intuitive because it is not a vertical, but a horizontal change. So we will now focus on this part: how to better represent the relation between processes, so the user can better navigate between those, knowing that at first the user is more used to the classic data file manager, which he does not find in Q.wiki.

There are some constraints we need to take into consideration: user processes are not an easy one-input, one-output process. The rule is many-to-many. As a rule of thumb, we hypothesize that there are usually between 2 and 3 input processes and between 2 and 3 output processes (see Figure 13).

We hypothesize that the connection between processes is the problem we must tackle. In order to verify that, we will ask experts from different departments.

3.2 Expert Interview on the Problem

Understanding the core of our problem means looking at it from different angles. Therefore, we will interview experts from different parts of our domain. First, we will describe the methodology of our interview, and then summarize the findings of each interview. Our experts are coming from three different departments, providing different points of view. Namely, we will interview a sales expert who knows which problem a new user faces before he buys the product, a consultant expert, who knows what problems the user faces once he bought the product and starts to use it more intensively, and finally a product expert, who knows about our product and knows both the strategical vision and tactical aspects of our product.

3.2.1 Motivation and methodology for Expert Interview on the Problem

Although we observed many aspects of our problem, we can not yet pinpoint exactly what the underlying causes are. We need a problem definition, defined as the gap between the current state and the desired state [18]. We observed the *current* state, but we do not fully understand its causes, and we need help to understand the *desired* state. Since we are still in the exploratory phase, we can not use extremely precise questions to interview our experts, but we already have concrete points of discussion. We want a clear definition of the problem from different points of view, so we need a so-called semi-structured interview approach,[2] in which we ask open questions, while having a plan. To define the problem, we

[2] https://www.scribbr.com/methodology/interviews-research/

chose the problem definition process of an article on the ?Toolshero website[3] because it is straightforward. It starts with describing the vision, then describing the problem, and its financial consequences. The article uses the so-called *Lean's 5-times why*, which consists of repeating 5 times *Why* to dig to the root cause of the problem. We adapted the questionnaire from the article using our user observations from Chapter 3.1.1. Our interview process follows these steps:

- First, we will describe the aim of the interview to the experts by explaining: We seek to understand which parts of Q.wiki's process structure users find confusing, intending to create an intuitive product that helps people think in terms of processes and navigate more easily, focusing on process discovery.
- Then we ask our expert to show us how a user discovers information in Q.wiki so he can better visualize what we mean by *process discovery*.
- Next, we ask:
  1. What are the problems there?
  2. What is missing, or what is too much to improve better discovery process?
  3. What is missing, or what is too much to better understand process orientation? (What do I need where?)
  4. What can influence the problem that users do not know where they are, and are confused about what they should do? (current state)
  5. What would the situation be if the software could better convey to users where they are? (desired state)
  6. What would be a good information architecture to present it (hierarchy with pyramid? better present predecessor and successor processes?)
  7. Why is it important to solve the problem? (Depending on the expert: Financially, how much money (sales), time (consultants), or usability challenge (product) does the problem cost?)

To improve readability, we present an organized version of our experts' interviews. We highlight the identified problems in bold and synthesize these findings in the following section.

3.2.2 Interview with Fabian Kröppel, Expert for Sales

Fabian Kröppel is an expert in sales at Modell Aachen GmbH, who knows the cost implications of our problem and the sales challenges when introducing Q.wiki to potential customers.

Kröppel highlights a core struggle users faced when discovering a process in Q.wiki, namely that **users must leave the page to understand where they are in the workflow**; this is the same issue that Rico Wilmink raised in the previous section under the name *u-turn* (3.1.2). They cannot know where they are without leaving the process page, and it is an obstacle to a more seamless navigation. Our expert describes our current way of displaying the location,

[3] problem definition process of the Toolshero website

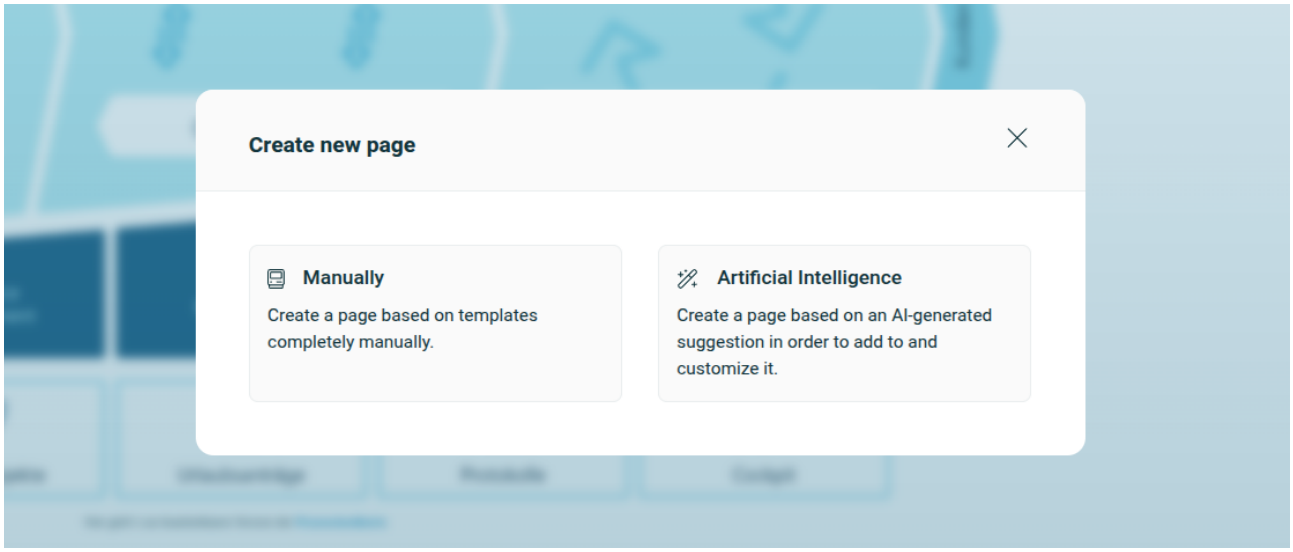

Figure 4: *When creating a page, the user must first decide if they want to do it manually or with AI, an unexpected prompt. Source: Q.wiki*

the breadcrumbs, as too technical and largely misunderstood by our users.

While the current state of having the predecessor and following processes before and after the process page is not a major concern for Kröppel, he describes itself of creating a page and then moving it as very counter-intuitive, and for him the biggest pain point is therefore **constructing and editing new process structures**, not understanding them once they're established.

The page creation process is non-intuitive: Currently, when creating a page, a menu comes asking whether we want to manually create it or with AI , which he states is an unexpected prompt. Instead, users expect to choose the page's destination immediately. **After creating a page, it is extremely difficult to find it**. The user has to go through the search bar to find the parent page, and even if he reaches it, it still has one more obstacle, that is, having to scroll down the parent page to return to his newly created page. The challenge is that customers come from a file system like SharePoint. In such a traditional file system, the location of a file within a folder is clear. However, in Q.wiki, even after finding a parent page (the equivalent of a category), users don't know where their newly created page (element) is located. This forces users to use the search bar to find the parent page, and then scroll extensively to locate their new content. This is therefore a violation of the human-centered idea of aligning with the user's mental model, which we saw in Chapter 2.3.1.

The desired state of our expert is therefore to be more guided in the process creation. Kröppel wishes **a feature for directly creating a sequence of processes** in Q.wiki, meaning creating multiple processes belonging to the same logical layer at the same time, because he states that the current way of creating such a sequence manually involves the massive use of our page creation feature, which as he already explained it, is very non-intuitive. Kröppel wishes more user support for **choosing where to put the page** and requests a

graphical drag-and-drop functionality for easily moving processes between different locations in Q.wiki, because currently **it is very difficult to move pages between different locations**. A possible user workflow for creating a page would, for example, look like the following: First, clicking on ”create process” and then having the choice of where to put it, or not having to choose it during the creation process, and later being reminded to choose a location. Also, he adds a request which was also made by Tim Grafenhorst, head of sales during the preparation for this research, to have an indication (for example with a number) of **how many processes are hidden behind a page** e.g. in the process overview because such an information is useful for auditors and helps them prioritize their work, therefore saving them time. This feature request aligns with the idea of a *Clear Visual hierarchy* we saw in Chapter 2.3.1.

Kröppel describes his ideal solution for this as a sort of mind map view (looking like a tree) to visualize how all the pages are linked together, and having the possibility to drag and drop for moving processes between different locations (see Figure 14) this aligns with the idea of a sitemap we saw in Chapter 2.3.2, with the difference that Kröppel would not display all existing pages at once, aligning with the principle of *Progressive Disclosure* we saw in Chapter 2.3.1.

When asked about the **financial relevance** of our problem, our expert shows that the biggest sales obstacle why Q.wiki misses sales is due to the category *Look and Feel*. This category led to a loss of 280,000 euros in potential Annual Recurring Revenue (ARR) in the last 365 days (state of June 6, 2025). Also, the absence of a tree structure makes us lose more than 80,000 euros (see Figure 16). Although *Look and Feel* is a broad category where many user experiences go into, according to our expert, the confusion about page creation and page position in the whole system is the most important contributor to it. For example, a common feedback is that user struggle to find their page after creating it, and this is categorized as bad *Look and Feel*. In a nutshell, our problem of conveying the structure is the biggest obstacle to more sales, a very costly problem (around 360,000 euros in ARR).

3.2.3 Interview with Sven Schneider, Expert for Consulting

Sven Schneider is an expert in consulting at Modell Aachen GmbH, who knows the challenges our users face when onboarding with Q.wiki after purchasing the product.

Schneider highlights a missing element in the user interface of the process landscape: the **lack of differentiation between management, core, and supporting processes** (German: Führungsprozesse, Kernprozesse und unterstützende Prozesse), which we explained in Chapter 2.1.1. While Q.wiki experts understand these distinctions after being told, the product itself fails to convey them effectively to new users. This is another case of the Q.wiki conceptual model being more implicit than explicit, and therefore a violation of the *Recognition over Recall* design principle we named in Chapter 2.3.1. We will categorize this issue under

the name: **implicit Q.wiki concept**.

Another challenge Schneider raises is Q.wiki's **Expert Domination** on the layer two because of the **complexity of creating a graph** (a flow of processes) there. In the second layer in particular, users can choose whatever representation they want; usually, they either choose a list or graphs. He shows an example where the second layer is a list of processes with no logical order, and another example with a complex graph with many technical terms. Schneider notices that since the experts are usually the ones making more complex graphic overviews, they use technical language, making it hard for ordinary users to understand. This creates a significant problem: the second layer becomes dominated by expert perspectives, rather than serving its intended purpose of helping all users comprehend the content. This is particularly problematic because the second layer plays a crucial role—it is where users discover and understand key processes.

Schneider highlights the need to take Q.wiki's flexibility when designing a solution: **we can not expect all our clients to use our 4-layer concept**. He states that although our model has 4 layers, the majority of our big clients use more than 4 layers for describing their complex company. Schneider's proposed solution is therefore to make a default for 4 layers with the possibility of indicating on each page which layer the page belongs to, while letting admin users the possibility of adding more layers. This goes into the idea of **better default** or more **user guidance** in general.

Another point in the category **more explicit Q.wiki Modell**, Schneider wishes to remove from the table of the process description the predecessor/successor processes. He explains that the table view of processes is our Unique Selling Point (USP) (German: Differenzierungsmerkmal), but points out that the processes before and after should be linked outside the table, for example on top of the process description page, with a way (for example through hovering or through an AI generated summary) to understand the context of the processes before and after without having to navigate to these, instead of having them inside the table. He would then expect the possibility to hover over the predecessor/successor processes and have a **small preview** of these without having to leave the page, similar to the way Wikipedia does it. This would align with the *Proximity Principle* we saw in Chapter 2.3.2, enabling the user to understand the context of processes without leaving the page.

His solutions would be two views, firstly a tree view like Fabian Kröppel (see Figure 14), secondly a graph (see Figure 15) showing the connection between the pages independently of their layer, which customers say they want, although he doubts the actual use of it. Schneider says that to explain those non-intuitive structural parts of our product, around 10 to 20% of the total consultant time is used, around 1 day just for that, if summing all small explanations effort during user training.

3.2.4 Interview with Thomas Vogt, Expert for Product Management

Thomas Vogt is an expert in product management at Modell Aachen who constantly works at improving the product.

Like Schneider, Vogt describes **inconsistencies in the quality of the layer two (process overview)**, the layer that is made by users themselves out of PowerPoint-generated images. Some process overviews are clear and easy to follow, others are hard to understand. There is sometimes an information overload on the process overview layer, and often on the process layer: the users need time to get used to these layers or to find important information in them. According to him, one reason why users favor search over navigation is the abundance of information in some process overviews.

Vogt states that **he does not recognize the process orientation in Q.wiki outside the first two layers**, the first layer always being there, the second layer not always having a graphic representation of the process flow. Aligning with Schneider, Vogt explains that **the process representation in Q.wiki comes more from building up the product through advice, with the consultant, than from Q.wiki itself.**

Vogt, like Schneider, presents the table display of process description as an important feature of Q.wiki, stating that it is a big improvement over the lengthy Word documents that the user comes from; however, **some tables are long**. This is for us a warning sign for higher cognitive load, which we saw in Chapter 2.2.2. This raises the need to add more dynamic recommendations to the software, for example, by warning users when they create processes with too many steps, or automatically hiding those, raising the potential for **more user guidance**.

To him, another major problem is how much Q.wiki **relies on templates** (we saw this in Chapter 2.1.3). Currently, when the user clicks to create a page, he has to choose between different templates. The default that is proposed to him in the process landscape (layer one) is the work instruction (Arbeitsanweisung in German, starting with an A, layer 4) because of the alphabetical order of templates. However, this does not make sense since a work instruction belongs to layer 4 and does not belong as a “child” of layer 1. This raises the issue of too much **flexibility**, which leads to inconsistencies and adds **complexity**, a relation which we saw in inverse relationship in the Chapter 2.3.1: *constraints* are a way of managing complexity, or cognitive load in general, and the point of Vogt is the converse implication, that too much flexibility or the lack of constraints adds complexity. The simplest idea to resolve this issue is to give a **better default**. The Q.wiki user would still have the flexibility of adding any template anywhere, but it would be easier for him to make the right choice, namely in our example, having the possibility of directly adding a process overview (layer 2) from the process landscape (layer 1) without being proposed other templates. This is also a case of more **user guidance**.

On the following point, all our experts align perfectly, namely on the difficulty of creating

or navigating sequences of processes in Q.wiki. A core idea of Q.wiki's process orientation we saw in Chapter 2.1.1, is going away from the folder structure, where to navigate between different files, one has to do U-turns. The navigating procedure in folders is first navigating to a folder, looking at the file, then exiting the folder to enter another one. Opposite to that, our natural mental model of processes is to be able to easily navigate between them with one click, knowing what is before, what comes after, without the need to go back and forth. But such a seamless navigation is not yet built into Q.wiki. Like Schneider, Vogt thus wishes to have the predecessor and successor processes as structured information.

For him, the core problems are threefold. Firstly, understanding where the user is, with a part of a tree, knowing on which layer the user is (orientation). Secondly, inside a process, easily understand what comes before and after (proximity principle). Lastly, displaying processes in an easier easy-to-consume format, hiding the detail level that is not yet relevant (progressive disclosure).

He estimates that 10% of potential users are put off by it and use Q.wiki less often as a result of it (in the words of Vogt, there is 10% less lock-in effect). Solving the issue would then bring 10% more active use of Q.wiki. He explains, however, that the current structural issue of Q.wiki has not been dealt with in depth so far because this problem seems an aggregate of many smaller issues, making it complex to evaluate and difficult to solve at once.

We will now synthesize our findings and present our results.

3.3 Results

Through conducting those interviews, we realized the complexity of our problem. Like Vogt put it (Chapter 3.2.4), our problem is an aggregate of many sub-problems, which alone do not push into action, but together become a highly important issue to be tackled. The hypothesis of Chapter 3.1.2 that the most important problem is the navigation between predecessor/successor processes is refuted: it is only the tip of the iceberg. Solving our problem involves taking it at its root. However, we need to be strategic about it and first solve the subproblems with the highest impact and that are the easiest to fix.

To identify the core subproblems with the highest impact, we decided to plot all the problems into a matrix (see the whole matrix as Figure 18 in the Appendix or the excerpt here 5). Such a matrix is usually called an impact/effort matrix in the literature (See [4]). We created this matrix by listing each problem raised by our expert and then organizing it into two axes: the horizontal axis aims to evaluate the positive impact the subproblem would have on the users according to our expert interview, the vertical axis is the implementation effort based on our perception of what needs to be done to solve each issue. A limitation of this matrix is the difficulty of estimating the implementation effort, which is a known challenge in software (see

[4] https://productschool.com/blog/product-fundamentals/impact-effort-matrix

[5]). A rough estimation was therefore used for this purpose.

The **Do First** category is for high-impact problems that require a relatively lower effort. Three sub-problems fall into it: The non-intuitive page creation as discussed in the section 3.2.2, the difficulty in finding newly created pages as discussed in section 3.2.2, and the need for separating the predecessor and successor processes from the table view as discussed in sections 3.2.3 and 3.2.4.

While solving these issues would have a measurable impact on the product, it does not solve the hardest issue with the highest impact: **the Strategic Projects**, which would require major changes to the process model code. However, starting with solving the do-first category would potentially facilitate the implementation of the strategic projects. For instance, removing the predecessor/successor processes in the table view and, for example, modeling them like Schneider suggests, with a possibility of hovering on them to have a preview like Wikipedia would maybe allow for better navigation and facilitate solving a huge issue like the U-turn issue pointed out by all our experts, that users must leave the page to understand where they are in the process workflow.

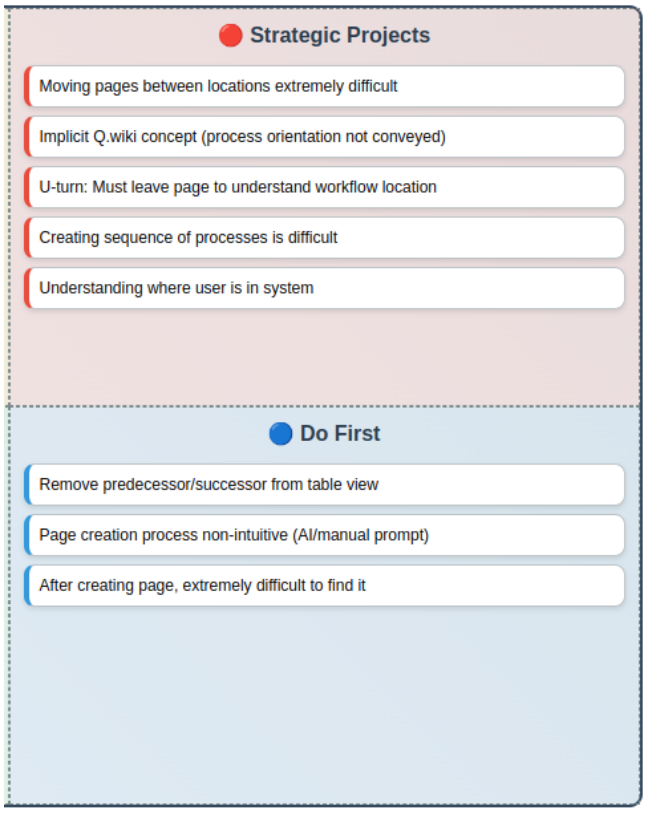

Figure 5: *Excerpt of the impact/effort matrix, which can be found as Figure 18 in the Appendix. The categories Strategic Projects and Do-First are for the subproblems having the highest impact. The Strategic Projects are the top because they have the highest estimated implementation effort, whereas the Do-First projects seem easier to implement.*

[5] Software Estimation Is Hard. Do It Anyway.

4 Summary

We introduced Q.wiki and its context as a process-oriented management system with a focus on interactivity and ease of use. We saw that processes are a core element of Q.wiki and raised a structural problem related to orientation and navigation in those processes. To start solving our problem, we learned that companies wanting Sustained Success must produce Successful Users, therefore optimizing for software usability and learnability. We understood what burden software can put on humans wanting to use it, a learnability challenge called cognitive load. We learned about different principles to manage the cognitive burden in a way that fits with users' mental model, and learned how to use visualizations to offload cognitive load. We discover that the process of simplifying information for human consumption is paradoxically challenging, and the field that addresses this exploratory endeavor is human-centered design. Furthermore, we found that the term structure and generally information is a complex subject which has been studied through many angles, for example, under the study of information architecture, and that many views are needed to address different aspects of information without overloading and therefore confusing our users.

We began exploring the problem space (Effective visualization of structure in order to increase the usability of the software) using the iterative method of human-centered design, and thus started with observing our users. Users appear to experience considerable confusion. However, we could not pinpoint the root cause of this confusion.

Consequently, we asked experts to exactly define their issues from many angles. Compiling the results lead to the fact that there were many subproblems that needed to be solved in order to improve the usability of the product: our problem is an iceberg with multiple parts: Compiling the results lead to the realization that there are many subproblems that needed to be solved in order to better convey the structure of the product. Even though there were many sub-problems with different perspectives from the experts, everyone aligned on the view that this problem was worth solving as it has a high financial relevance. Additionally, the consultants explained that it was time-consuming for them to constantly explain the conceptual aspects (categories of subproblems linked to the implicit Q.wiki model), which, according to our background sections on cognitive load theory, should be part of the software. This problem touches upon the most important domain of process orientation which is the core of the product, making it a huge opportunity at a same time a risk to the business of Model Aachen GmbH. Put in numbers, solving this problem has the potential to bring 360,000 euros in ARR (Kröppel), reduce the training time needed through consultants by 10 to 20% (Schneider), and increase active use of Q.wiki by 10% (Vogt).

Given the complexity of conveying structure in Q.wiki, an impact/effort matrix helped structure the subproblems and identify three high-impact, relatively easy-to-implement solutions as starting points. Strategic projects—the most difficult yet impactful subproblems—require eventual resolution as well.

Bibliography

[1] Rip Stauffer and Debra Owens. “Lasting Impression.” In: *Quality Progress* 45.11 (Nov. 1, 2012). Publisher: American Society for Quality, Inc., pp. 24–29. ISSN: 0033-524X. URL: https://research.ebsco.com/linkprocessor/plink?id=94fcbc7b-f8e1-3658-9257-636b7d6961f4 (visited on 06/03/2025).

[2] Lauri Koskela, Algan Tezel, and Viranj Patel. “Theory of Quality Management: Its Origins and History”. In: ed. by C. Pasquire and F. R. Hamzeh. Conference Name: 27th Annual Conference of the International Group for Lean Construction, IGLC2019 Meeting Name: 27th Annual Conference of the International Group for Lean Construction, IGLC2019 Num Pages: 549596. IRL: The International Group for Lean Construction, July 5, 2019, pp. 1381–1390. ISBN: 978-1-5108-9927-8. URL: https://publications.aston.ac.uk/id/eprint/43139/ (visited on 05/23/2025).

[3] Marlon Dumas et al. *Fundamentals of business process management*. Second edition. Berlin: Springer, 2018. 1 p. ISBN: 978-3-662-56509-4.

[4] Dan Tamir, Oleg V. Komogortsev, and Carl J. Mueller. “An effort and time based measure of usability”. In: *Proceedings of the 6th international workshop on Software quality*. ICSE ’08: International Conference on Software Engineering. Leipzig Germany: ACM, May 11, 2008, pp. 47–52. ISBN: 978-1-60558-023-4. DOI: 10.1145/1370099.1370111. URL: https://dl.acm.org/doi/10.1145/1370099.1370111 (visited on 05/26/2025).

[5] Helen Forsey et al. “Designing for Learnability: Improvement Through Layered Interfaces”. In: *Ergonomics in Design* (Aug. 23, 2024). Publisher: SAGE Publications Inc, p. 10648046241273291. ISSN: 1064-8046. DOI: 10.1177/10648046241273291. URL: https://doi.org/10.1177/10648046241273291 (visited on 05/05/2025).

[6] Kathy Sierra. *Badass: making users awesome*. 1. ed. Sebastopol, Calif.: O’Reilly, 2015. 286 pp. ISBN: 978-1-4919-1901-9.

[7] *Gartner Survey Reveals 47% of Digital Workers Struggle to Find the Information Needed to Effectively Perform Their Jobs*. Gartner. URL: https://www.gartner.com/en/newsroom/press-releases/2023-05-10-gartner-survey-reveals-47-percent-of-digital-workers-struggle-to-find-the-information-needed-to-effectively-perform-their-jobs (visited on 05/12/2025).

[8] John Sweller. “Cognitive Load During Problem Solving: Effects on Learning”. In: *Cognitive Science* 12.2 (1988), pp. 257–285. ISSN: 1551-6709. DOI: 10.1207/s15516709cog1202_4. URL: https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1202_4 (visited on 05/05/2025).

[9] Eliyahu M. Goldratt. *What is this thing called theory of constraints and how should it be implemented?* Great Barrington, Massachusetts: North River Press, 1990. 160 pp. ISBN: 978-0-88427-166-6 978-0-88427-085-0.

[10] Richard E. Mayer and Roxana Moreno. "Nine Ways to Reduce Cognitive Load in Multimedia Learning". In: *Educational Psychologist* 38.1 (Jan. 1, 2003), pp. 43–52. ISSN: 0046-1520, 1532-6985. DOI: 10.1207/S15326985EP3801_6. URL: https://www.tandfonline.com/doi/full/10.1207/S15326985EP3801_6 (visited on 05/13/2025).

[11] R. Sharma, V.I. Pavlovic, and T.S. Huang. "Toward multimodal human-computer interface". In: *Proceedings of the IEEE* 86.5 (May 1998), pp. 853–869. ISSN: 1558-2256. DOI: 10.1109/5.664275. URL: https://ieeexplore.ieee.org/document/664275/ (visited on 05/13/2025).

[12] Colin Ware. *Information visualization: perception for design*. Fourth edition. Cambridge, MA: Morgan Kaufmann, an imprint of Elsevier, 2021. 538 pp. ISBN: 978-0-12-812875-6.

[13] Donald A. Norman. *The design of everyday things*. Revised and expanded edition. New York, New York: Basic Books, 2013. 347 pp. ISBN: 978-0-465-05065-9.

[14] M. S. Mayzner and R. F. Gabriel. "Information "Chunking" and Short-Term Retention". In: *The Journal of Psychology* 56.1 (July 1963), pp. 161–164. ISSN: 0022-3980, 1940-1019. DOI: 10.1080/00223980.1963.9923710. URL: http://www.tandfonline.com/doi/abs/10.1080/00223980.1963.9923710 (visited on 05/16/2025).

[15] Yanxia Liang. "Application of Gestalt psychology in product human-machine Interface design". In: *IOP Conference Series: Materials Science and Engineering* 392.6 (July 2018). Publisher: IOP Publishing, p. 062054. ISSN: 1757-899X. DOI: 10.1088/1757-899X/392/6/062054. URL: https://dx.doi.org/10.1088/1757-899X/392/6/062054 (visited on 06/13/2025).

[16] Soyoung Park. "A Study on Visual Scaffolding Design Principles in Web-Based Learning Environments". In: *Electronic Journal of e-Learning* 20.2 (Feb. 14, 2022). Number: 2, pp180–200. ISSN: 1479-4403. DOI: 10.34190/ejel.20.2.2604. URL: https://academic-publishing.org/index.php/ejel/article/view/2604 (visited on 05/05/2025).

[17] Louis Rosenfeld, Peter Morville, and Jorge Arango. *Information Architecture: For the Web and Beyond*. Google-Books-ID: dZaJCgAAQBAJ. "O'Reilly Media, Inc.", Sept. 9, 2015. 485 pp. ISBN: 978-1-4919-1355-0.

[18] James O. Coplien and Gertrud Bjørnvig. *Lean architecture: for agile software development*. Repr. with corr. Chichester: Wiley, 2011. 357 pp. ISBN: 978-0-470-68420-7.

[19] Ali Darejeh et al. *A critical analysis of cognitive load measurement methods for evaluating the usability of different types of interfaces: guidelines and framework for Human-Computer Interaction*. Feb. 19, 2024. DOI: 10.48550/arXiv.2402.11820. arXiv: 2402.11820[cs]. URL: http://arxiv.org/abs/2402.11820 (visited on 05/12/2025).

[20] Thomas Kosch et al. "A Survey on Measuring Cognitive Workload in Human-Computer Interaction". In: *ACM Comput. Surv.* 55.13 (July 13, 2023), 283:1–283:39. ISSN: 0360-0300. DOI: 10.1145/3582272. URL: https://dl.acm.org/doi/10.1145/3582272 (visited on 05/12/2025).

[21] Gregory L. Murphy. *The big book of concepts*. 1. MIT Press paperback ed. A Bradford book. Cambridge, Mass.: MIT Press, 2004. 555 pp. ISBN: 978-0-262-63299-7 978-0-262-13409-5.

[22] Allan M. Collins and M. Ross Quillian. "Retrieval time from semantic memory". In: *Journal of Verbal Learning and Verbal Behavior* 8.2 (Apr. 1969), pp. 240–247. ISSN: 00225371. DOI: 10.1016/S0022-5371(69)80069-1. URL: https://linkinghub.elsevier.com/retrieve/pii/S0022537169800691 (visited on 05/16/2025).

[23] B. Shneiderman. "The eyes have it: a task by data type taxonomy for information visualizations". In: *Proceedings 1996 IEEE Symposium on Visual Languages*. 1996 IEEE Symposium on Visual Languages. Boulder, CO, USA: IEEE Comput. Soc. Press, 1996, pp. 336–343. ISBN: 978-0-8186-7508-9. DOI: 10.1109/VL.1996.545307. URL: http://ieeexplore.ieee.org/document/545307/ (visited on 05/16/2025).

[24] Colin Ware. *Information visualization: perception for design*. 3d edition. Interactive technologies. Waltham, MA: Morgan Kaufmann, 2013. ISBN: 978-0-12-381464-7.

[25] B. Johnson and B. Shneiderman. "Tree-maps: a space-filling approach to the visualization of hierarchical information structures". In: *Proceeding Visualization '91*. Visualization '91. San Diego, CA, USA: IEEE Comput. Soc. Press, 1991, pp. 284–291. ISBN: 978-0-8186-2245-8. DOI: 10.1109/VISUAL.1991.175815. URL: http://ieeexplore.ieee.org/document/175815/ (visited on 05/13/2025).

[26] Hans-Jorg Schulz. "Treevis.net: A Tree Visualization Reference". In: *IEEE Computer Graphics and Applications* 31.6 (Nov. 2011), pp. 11–15. ISSN: 1558-1756. DOI: 10.1109/MCG.2011.103. URL: https://ieeexplore.ieee.org/document/6056510 (visited on 05/13/2025).

[27] Mohammad Ghoniem, Jean-Daniel Fekete, and Philippe Castagliola. "On the Readability of Graphs Using Node-Link and Matrix-Based Representations: A Controlled Experiment and Statistical Analysis". In: *Information Visualization* 4.2 (June 2005), pp. 114–135. ISSN: 1473-8716, 1473-8724. DOI: 10.1057/palgrave.ivs.9500092. URL:

https://journals.sagepub.com/doi/10.1057/palgrave.ivs.9500092 (visited on 05/13/2025).

[28] Carolina Nobre, Marc Streit, and Alexander Lex. “Juniper: A Tree+Table Approach to Multivariate Graph Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (Jan. 2019), pp. 544–554. ISSN: 1077-2626, 1941-0506, 2160-9306. DOI: 10.1109/TVCG.2018.2865149. URL: https://ieeexplore.ieee.org/document/8454344/ (visited on 05/13/2025).

Appendix

Figures

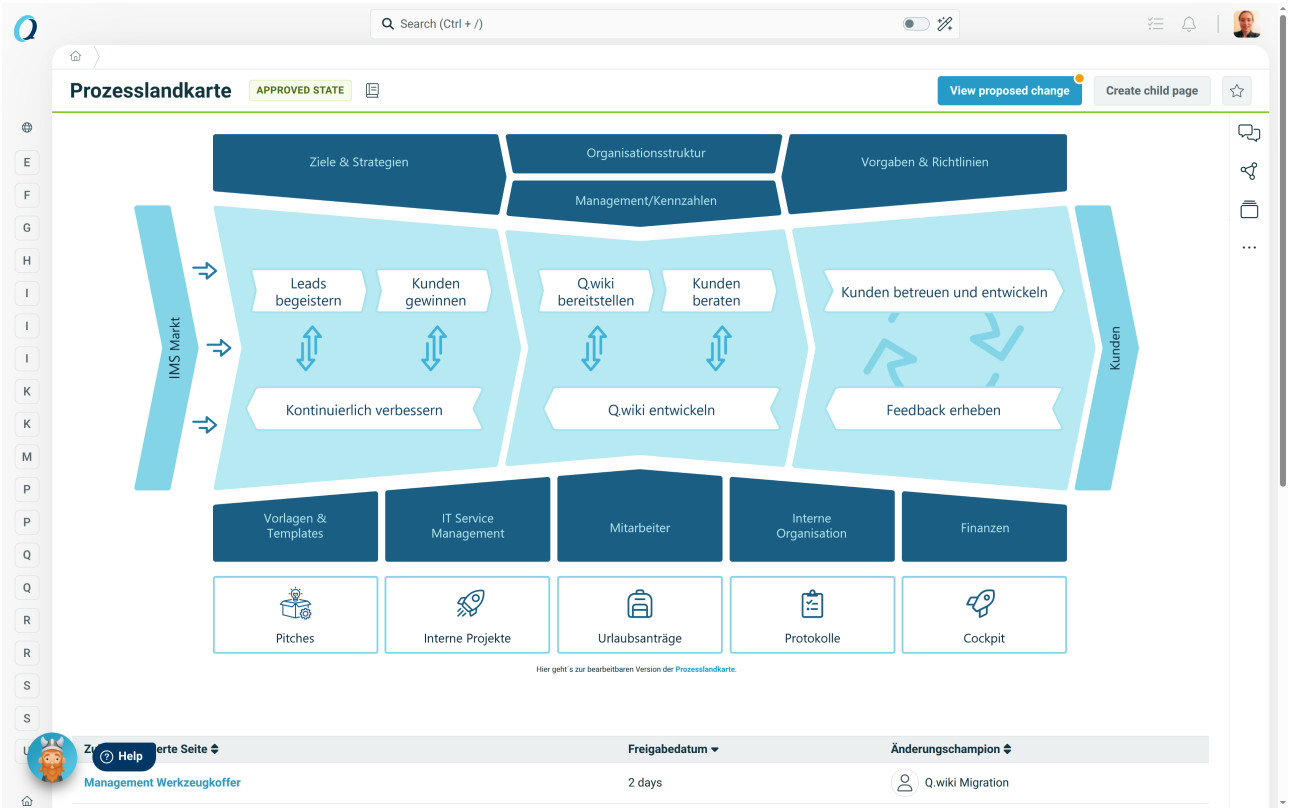

Figure 6: *The Process Landscape in Q.wiki*

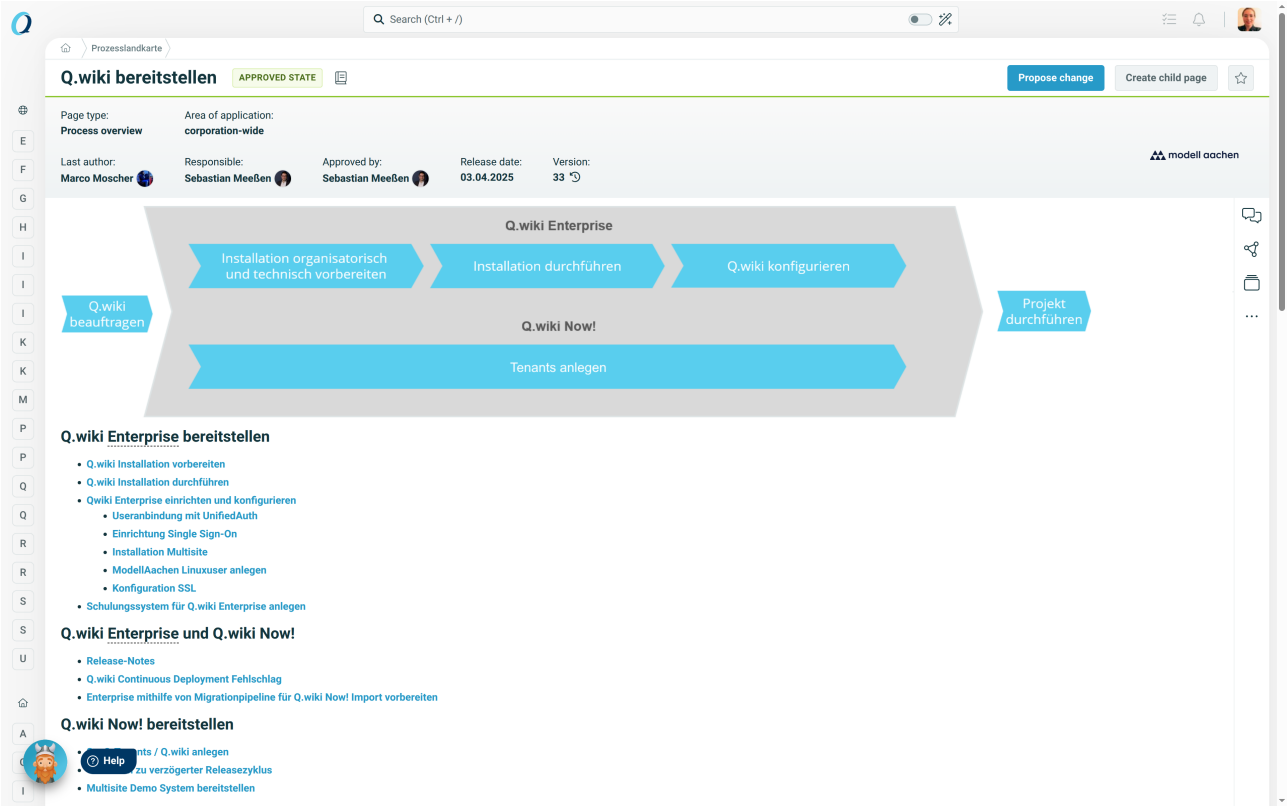

Figure 7: *A Process Overview in Q.wiki*

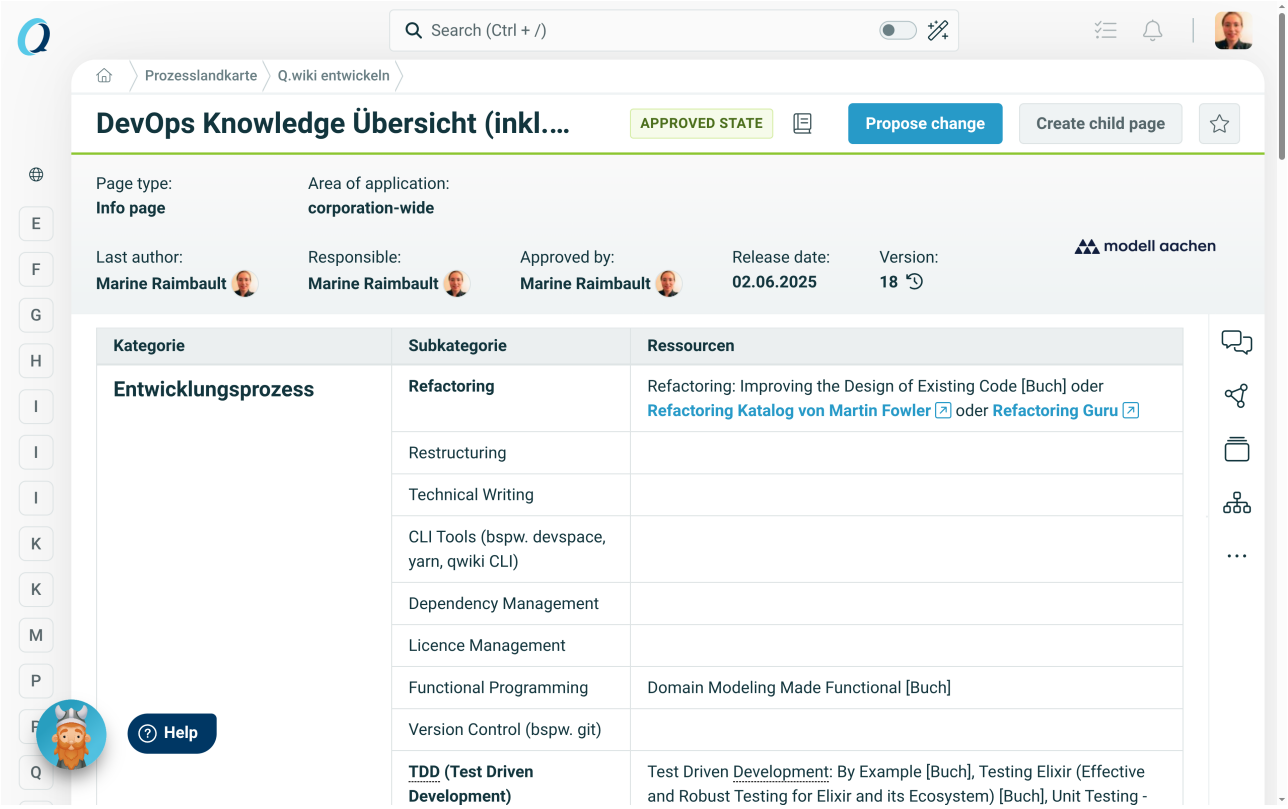

Figure 8: *An Info Page in Q.wiki*

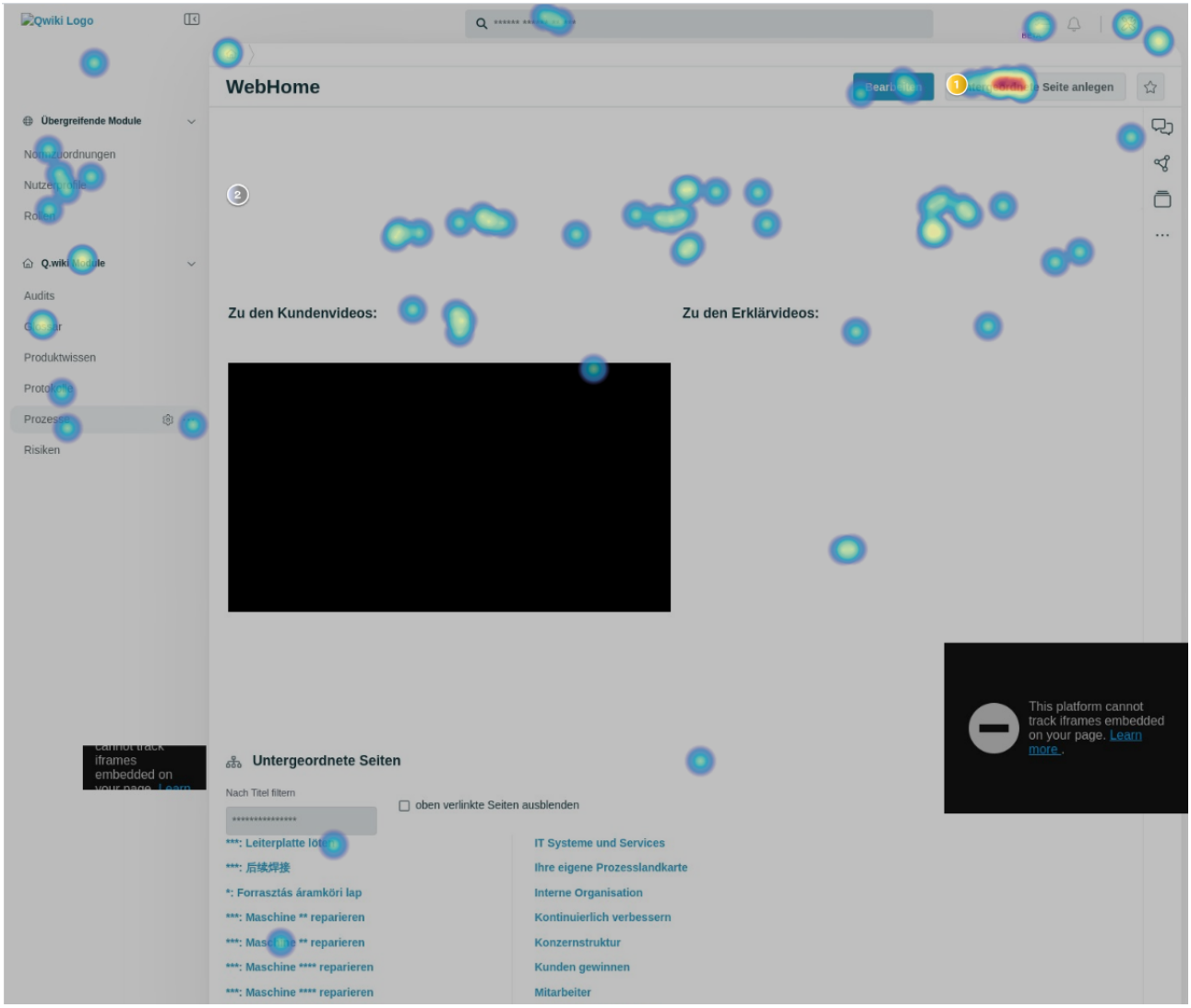

Figure 9: *The heatmap of the homepage shows that users focus on the top right "add page" button and spend a lot of time in non-important spots in the middle of the screen. Average time on page: 2:30 minutes.*

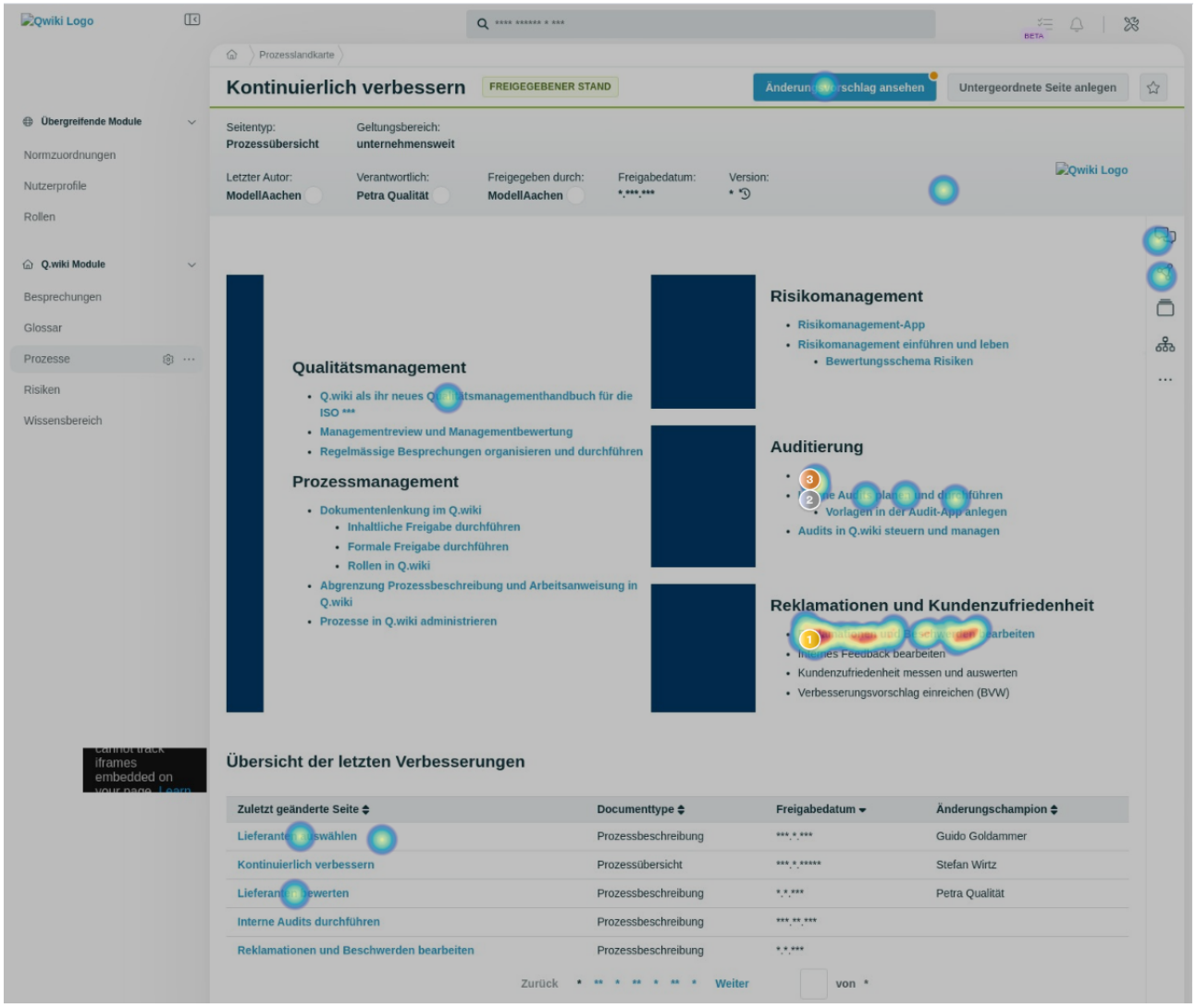

Figure 10: *The heatmap of the process overview shows that users do not use the module sidebar on the left, but use the comment and relation sidebar on the right. Average time on page: 0:59 minutes.*

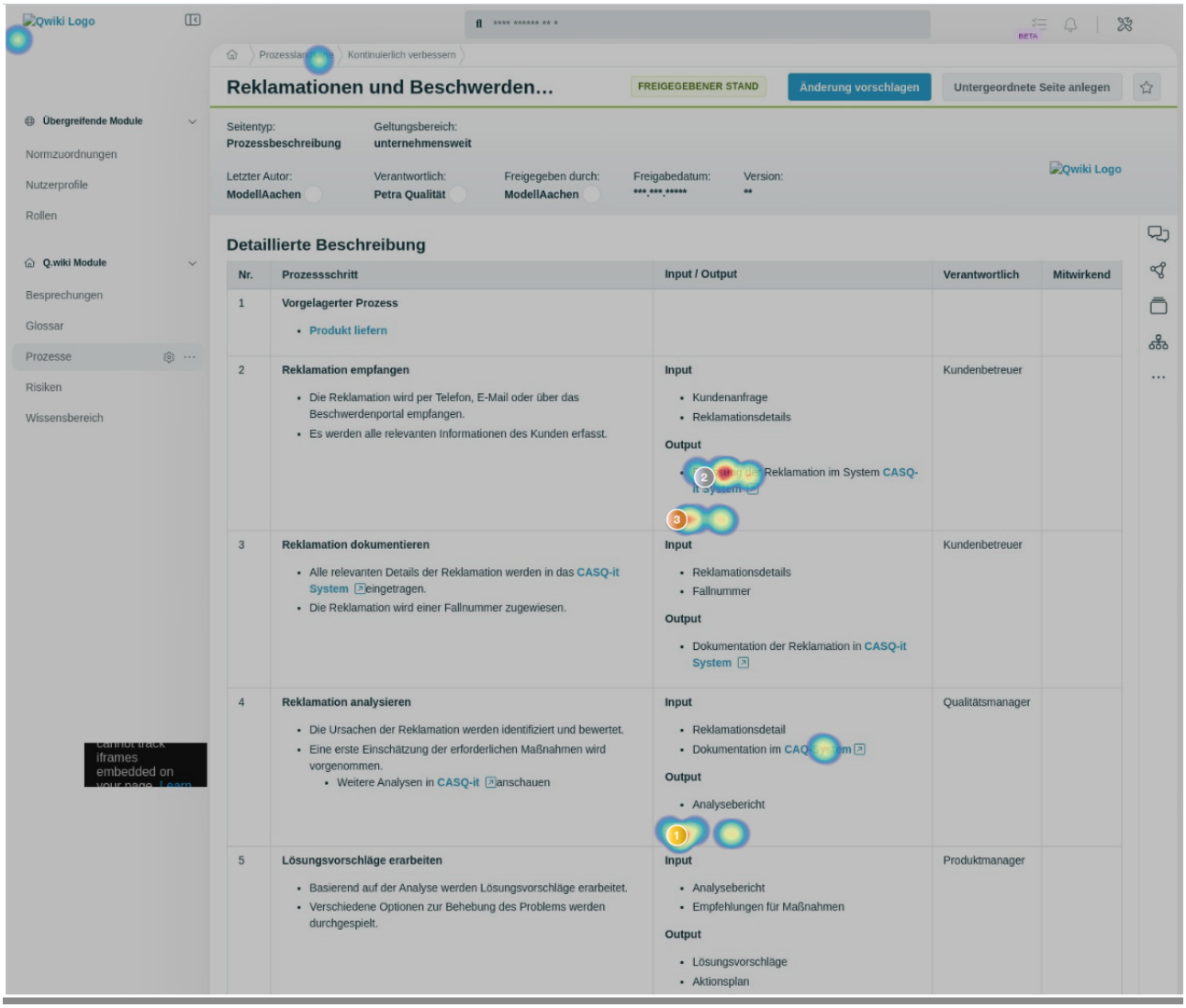

Figure 11: *The heatmap of the process descriptions shows that users mostly use the page description to then click on pages that are found below. Average time on page: 2:52 minutes.*

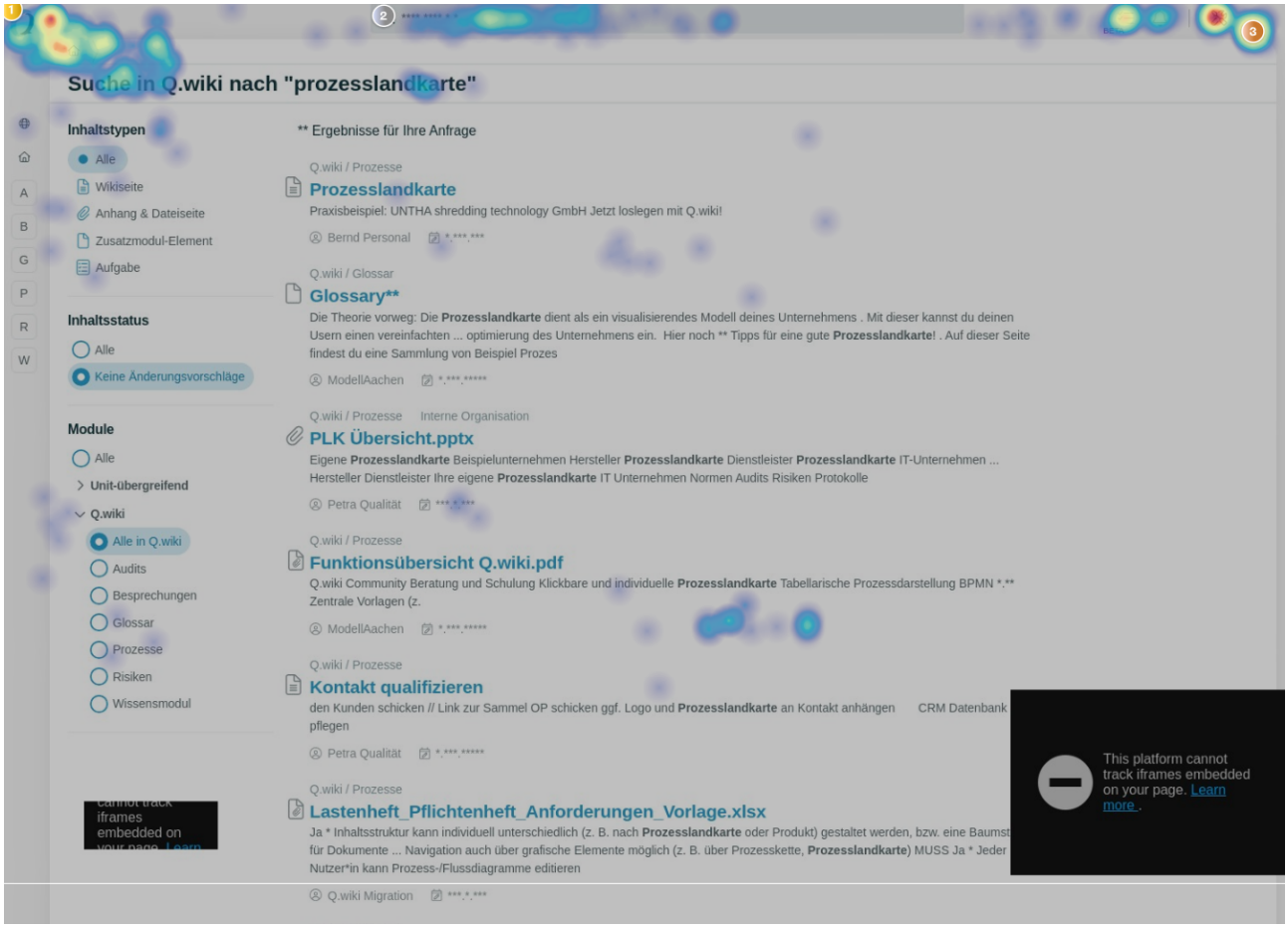

Figure 12: *The heatmap of the search page shows that users focus on three spots: the top left button for the homepage, the search bar, and the top right for their profile. The filtering options on the left were not used. Average time on page: 2:09 minutes.*

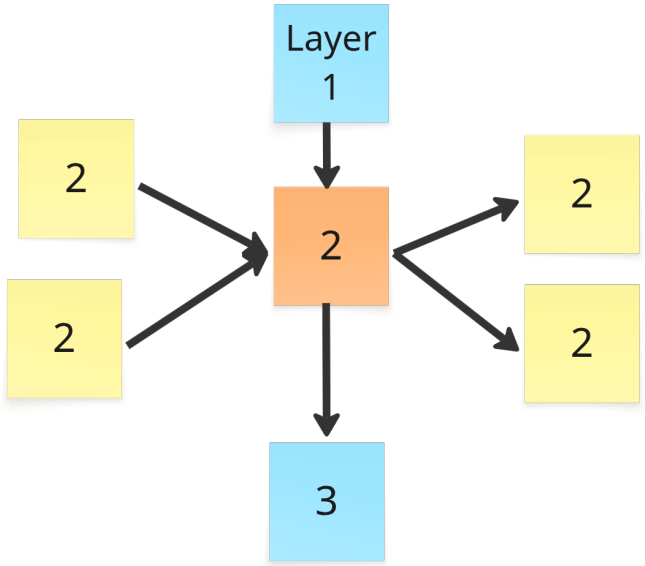

Figure 13: *Representing the process flow. Each post represents a process. The numbers represent the layers, or vertical layers. The orange process is the one the user is currently at. The yellow post-its represent that for each process, they can be any number of processes coming before (triggering this process) and any number of processes that the current one leads to.*

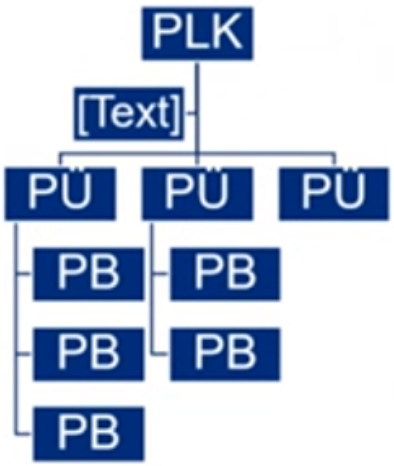

Figure 14: *A tree-like view of processes showing the relations between all the pages, from the process landscape (German: Prozesslandkarte PLK), to the process overview (German: Prozessübersicht PÜ) and the process description (German: Prozessbeschreibung PB)*

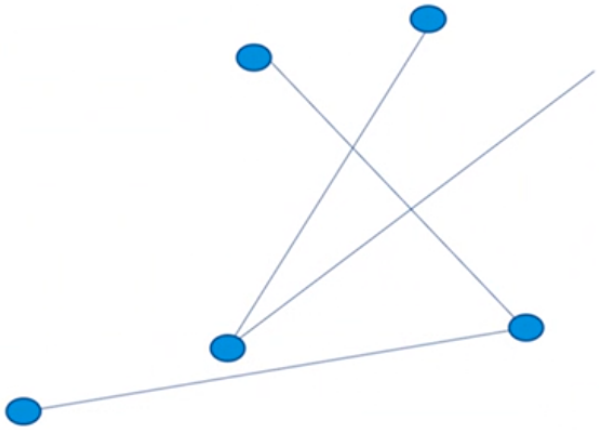

Figure 15: *A graph view showing connections between processes.*

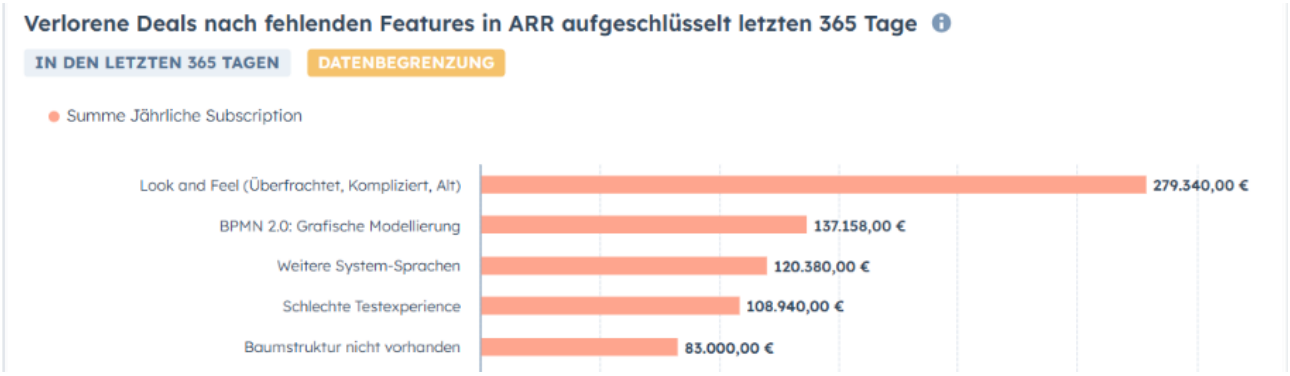

Figure 16: *Lost deals due to missing features. Look and feel ranks first (279,340 euros) while lack of tree structure ranks fifth (83,000 euros), totaling over 360,000 euros in lost ARR.*

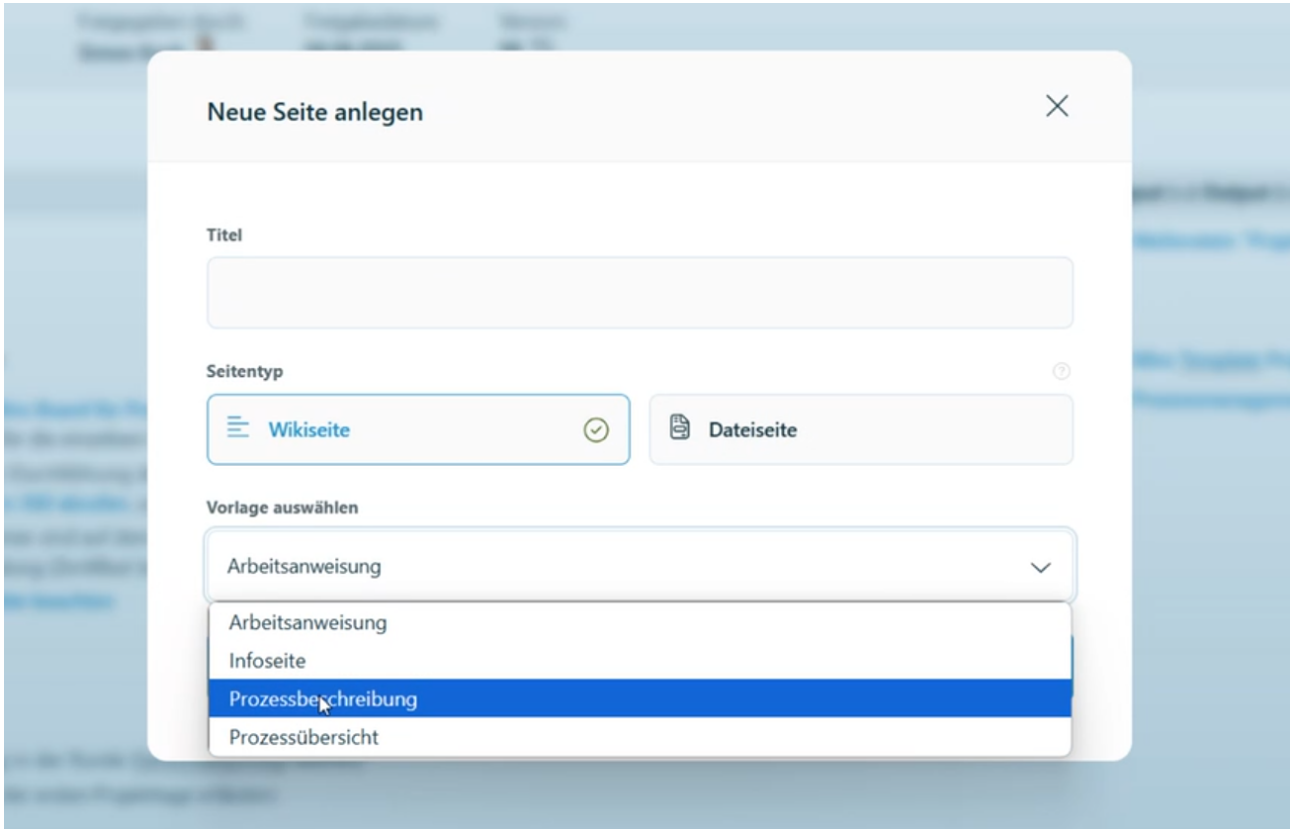

Figure 17: *When creating a process in Q.wiki, the user has to choose between different templates.*

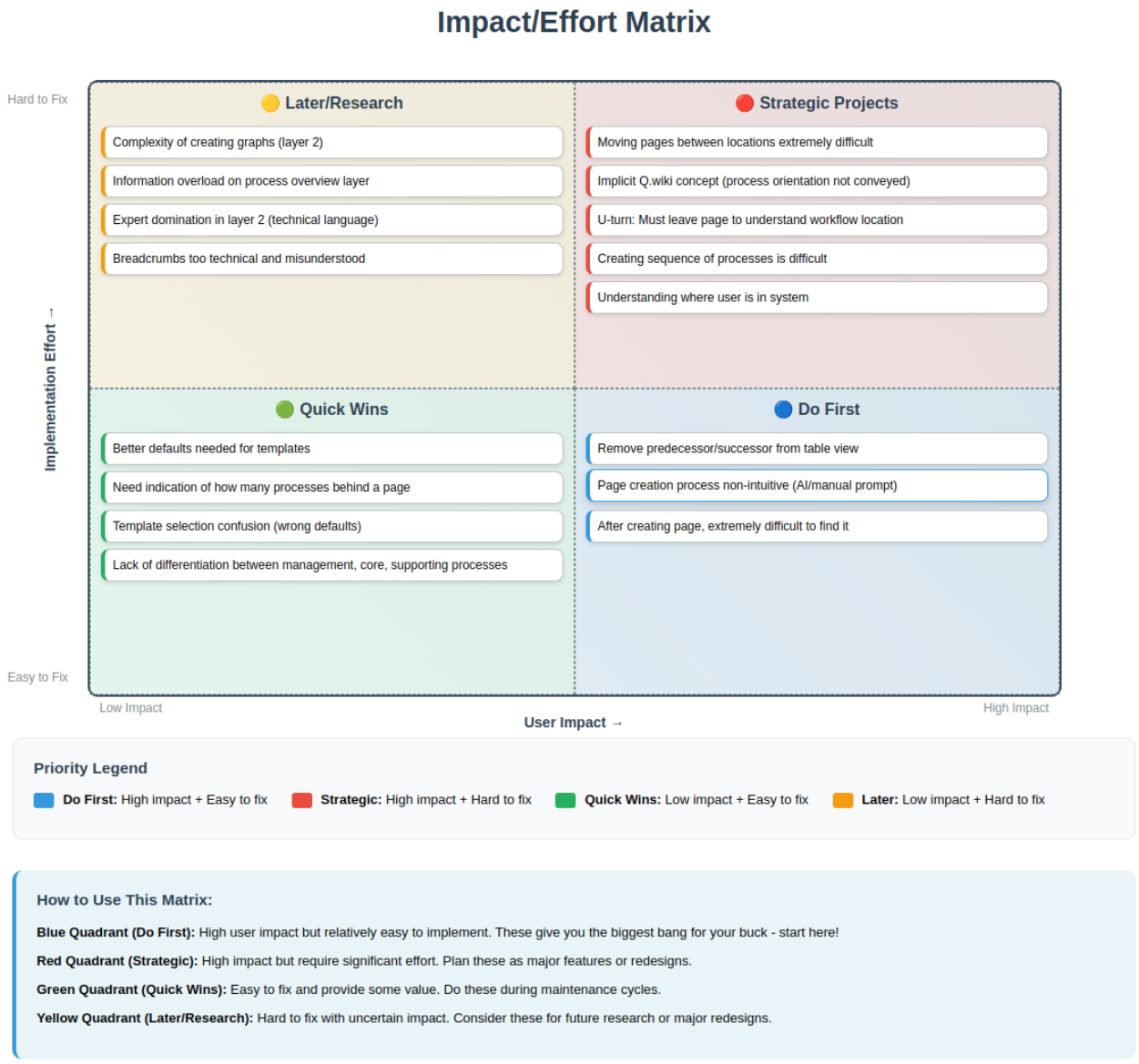

Figure 18: *Impact/Effort Matrix identifying the problems having the highest impact. Created by Marine Raimbault*

Two Removed Chapters

Strategies to Measure Cognitive Load in Software

Cognitive load can be measured through different methods, which offer unique advantages depending on the specifics of the chosen experiment [19]:

- **Self-reported questionnaires.** Instruments such as the NASA Task Load Index (NASA-TLX) enable participant assessment and provide a subjective measure of cognitive load. The NASA-TLX specifically evaluates six dimensions [19]:
  1. How much mental effort was required? (Mental Demand)
  2. How much time pressure did you feel? (Temporal Demand)
  3. How much physical effort was required? (Physical Demand)
  4. How hard did you work to finalize the task? (Effort)
  5. How successful were you in completing the task? (Performance)
  6. How disappointed, Were you bored or annoyed while completing the task? (Frustration Level)
- **Performance-based measures.** These include:
  - Dual-task methodology: adding a second task (e.g., rhythmic tapping) while performing a primary task. Worse secondary task performance suggests higher cognitive load on the primary task.
  - Time-on-task analysis: Longer duration spent on specific interface elements suggests higher cognitive processing requirements.
  - Error rate comparison: Analyzing mistakes made while completing identical tasks across different interface designs.
- **Physiological indicators.** These provide continuous, objective measurements:
  - Eye-tracking metrics: Prolonged fixation duration on interface elements indicates increased processing demands.
  - Heart rate variability (HRV): Decreased HRV during short tasks correlates with increased mental stress.
  - Electroencephalography (EEG): Direct measurement of brain activity patterns associated with cognitive load fluctuations.

Despite the widespread adoption of questionnaires like NASA-TLX due to their simplicity, Kosch et al. describe them as "*an inherited rather than an efficient tool*" and recommend researchers prioritize more objective physiological measurements when feasible [20]. Darejeh et al. [19] suggest that optimal cognitive load measurement during usability testing should satisfy two key criteria:

1. it should not introduce participant discomfort
2. it should enable continuous measurement throughout the session rather than only at completion.

Their analysis recommends task performance metrics, time-on-task measurements, dual-task paradigms (particularly tapping methods), mouse movement analysis, facial expression monitoring, and linguistic feature analysis as primary methods. They further indicate that self-report questionnaires provide useful complementary information, although they should be asked at the end of the tests to not interfere with users' experiences.

Techniques for Visualizing Hierarchical and Networked Structures

To make compelling visualizations, it is essential to align the data structure with how people naturally think and navigate through information. Researchers from different disciplines use two structural paradigms: hierarchies and networks. Hierarchical structures support categorization and abstraction, similar to how humans can simplify complex tasks regarding nested relations with each other [21]. Network structures, in contrast, reflect how knowledge is connected and retrieved through associative links [22]. These two models are fundamental to numerous software and organizational tools and the foundations for designing visualizations for reduced cognitive load [23, 24]).

By focusing on these two structural paradigms—hierarchies and networks—we can identify visualization strategies that mirror users' mental models, making complexity more approachable and reducing unnecessary cognitive load. Several established approaches exist for visualizing hierarchical and networked structures:

- Hierarchical visualizations such as listings, outlines, and tree diagrams effectively convey parent-child relationships, but they do not scale well to large information structures [25]:
  - *Listings* present detailed content but make it difficult to grasp the overall structure;
  - *Outlines* structure structural and content information, but limited screen space restricts their effectiveness;
  - *Tree diagrams* offer intuitive visual layouts for small hierarchies, but they use screen space inefficiently and become impractical for large datasets. Researchers created the website treevis.net to provide a comprehensible overview of the existing types of Tree diagrams [26]
- Network visualizations effectively show connections between elements [27]:
  - *Node-link diagrams* for small graphs and
  - *matrix* representations for bigger ones
- Hybrid approaches: Combined techniques that integrate hierarchical and network elements to represent complex organizational structures [28]