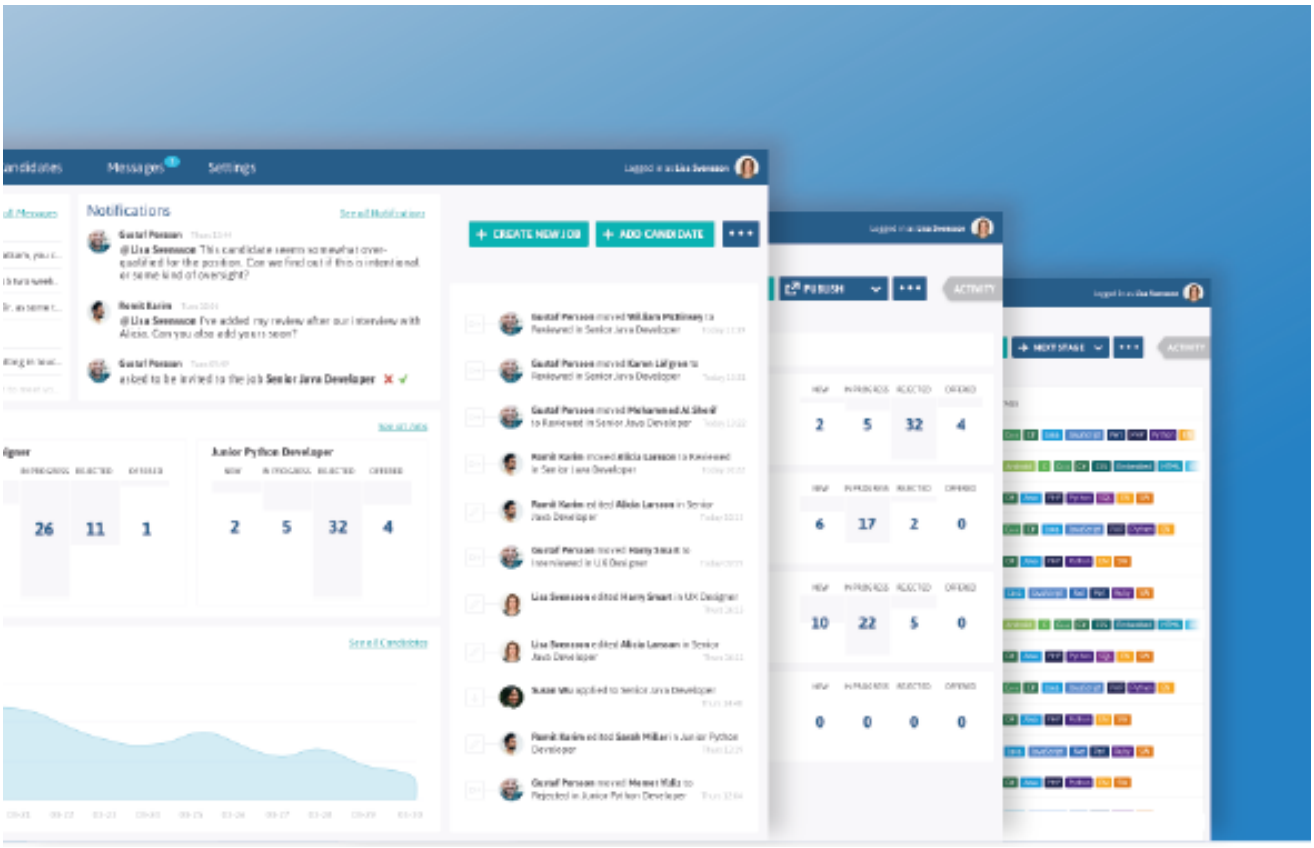
CHALMERS
UNIVERSITY OF TECHNOLOGY



Designing Honeycomb

Patterns for Awareness and Collaboration in Applicant Tracking Systems

Master's thesis in Interaction Design and Technologies

JOHANNES LUNDQVIST
ANNA WEISS

Department of Applied IT
CHALMERS UNIVERSITY OF TECHNOLOGY

Report No. 2016:121

Designing Honeycomb

Patterns for Awareness and Collaboration in Applicant Tracking Systems

Johannes Lundqvist
Anna Weiss

Department of Applied IT
Chalmers University of Technology
Gothenburg, Sweden 2016

Designing Honeycomb
Patterns for Awareness and Collaboration in Applicant Tracking Systems
JOHANNES LUNDQVIST
ANNA WEISS

Technical report no 2016:121
Department of Applied IT
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover:
Three screens of the Honeycomb applicant tracking system. Read more about it in section 6.1.

Gothenburg, Sweden 2016

Abstract

This master thesis investigates how to design applicant tracking systems (ATS) to facilitate awareness, collaboration, and efficiency (ACE). For this purpose, an HCD process has been undertaken, encompassing primary and secondary research, and implementation and evaluation methods. As a result of the investigation the prototype of the Honeycomb ATS has been designed. Based on the prototype, fourteen design patterns that enhance the collaboration and awareness of teams in CSCW applications have been developed and the accompanying ACE pattern model has been established. While these results have been evaluated against each other, and yielded an effective mutual integration, more work needs to be done in the future to test and verify those patterns in a broader context.

Keywords:

Applicant Tracking System, Patterns, Awareness, Collaboration, Collaborative Web Application, e-Recruiting, Groupware, CSCW, Interaction Design.

Contents

Acknowledgments

We would like to thank our supervisor Staffan Björk for letting us partake of his experience and expertise, and helping us find the right path through the master thesis jungle.

Thank you to the team at Software Skills for enthusiastically participating in the many methods and for all the great feedback. Especially to Henrik, who started everything, and for opening your office to us that has become our home in the past few months.

Thanks also to the "bachelor kids", Erik, Linus, Oskar, Adrian, and Martin, for not giving up in the face of all the design changes, and for breathing life into Honeycomb.

We also want to express our gratitude to all the other interview and test participants, everyone who gave us tips and feedback, and the friends who have reviewed our report. And thank you, IxD class of 2016! It has been great!

Abbreviations

ATS	Applicant Tracking System
ACE	awareness, collaboration, and efficiency
CMS	Content Management System
CSCW	Computer Supported Cooperative Work
CMI	Computer-Mediated Interaction
GDD	Goal-Directed Design
HCD	Human-Centered Design
HCI	Human Computer Interaction
HR	Human Resource
ISO	International Standardization Organisation
IXD	Interaction Design
RIA	Rich Internet Application
SME	Subject Matter Expert
UI	User Interface
USP	Unique Selling Proposition
UX	User Experience

1

Introduction

This master thesis project was conducted by two students of the master program Interaction Design and Technologies at Chalmers University in Gothenburg, Sweden. It was mandated by Software Skills, an IT recruitment company, also based in Gothenburg, who have commissioned the concept and design development of an Applicant Tracking System (ATS). This is a comprehensive application whose purpose is to handle everything related to the hiring process: from advertising new job positions, to managing a large number of applicants and their data throughout the recruitment procedure (Holm 2012). These systems can come either as dedicated desktop software or as web applications and are used by both recruiting companies and internal human resource departments. Although many ATSs exist and are marketed as highly complex and powerful applications, a substantial portion of them are heavy and interface and interaction components struggle to keep up with the new standards that have been made possible through recent technology developments in web standards and internet speed, as found during the pre-study of this project.

Also, while there is literature concerned with the business effects of e-recruiting (for example Holm 2012, Bartram 2000, Feldman et al. 2002, Furtmueller 2010, Geutal et al. 2005, Koong 2002, Thompson et al. 2008) – recruiting supported by electronic means – we found that there is a lack of research surveying and analysing the tools from a usability perspective. It is a substantial undertaking, as there are many aspects that should ideally be part of such an investigation. For one, it means to look at the systems and diagnose their strengths and shortcomings as tools, to recognize spaces for innovation. Apart from the existing solutions themselves, the circumstances where they are embedded need to be examined. This means a survey of user groups, stakeholders, and the organizations where they are deployed, and to identify their underlying goals and needs. Thirdly, the problem scope also requires attention to related practical and theoretical fields, therefore this thesis also looks to other web applications, as well as to the research field of Computer Supported Cooperative Work (CSCW) (see Guerrero and Fuller 2001; David et al. 2003).

There are two important aspects that we have identified as crucial for an ATS: The obvious one is the information management within the system. Simply said, an ATS is nothing more than a better Finder or File Explorer, where candidate information is stored and handled. Whether it is different job postings or different hiring stages, these are simply folders that the candidates are assigned to. The rest of the access functions – sorting, searching, filtering – are just accessories. So what makes an ATS interesting? It is the other decisive element of the recruitment process that we have recognized in our research: The team. And for a team to perform together dynamically the key is information sharing and communication (Schümmer and Lukosch 2007). The world is currently experiencing an upheaval in the way people work (Stone and Deadrick 2015) and consequently also what their workplaces and tools look like. With buzzwords like *Industry 4.0* emerging (Brettel 2014), and co-working spaces becoming evermore popular (Johns 2013), the traditional office jobs are subject to revision, too. This includes overall flexibility of when and where people work, manifested through the popularization of remote work and adaptive hours. If companies want to be able to offer these new models and reap their benefits, they need to plan for how to accommodate these temporal and spatial disparities. On a higher level, others name the rise of the knowledge economy and of globalization, together with an increasing diversity in the workforce as factors that influence the concept of work (Stone et al. 2015). Digital tools will play a significant role in adapting for the changes to come (Johns 2013). It is for these reasons, the stakeholders of this project saw a notable innovation space and the need in the realm of collaborative systems that allow teams to work synergistically. The domain of e-recruiting has tremendous growth potential and we endeavored to find specific ways to tap into this potential. To tie all of these aspects of team collaboration, hiring, and technology together and provide the findings in a useful format, this master thesis was set up to identify patterns that aim to help with the design of collaboration–facilitating software – more specifically applicant tracking systems.

Patterns are a way to organize and consequently enable recycling of solutions. This approach is used in various fields, for instance in IT development, where chunks of code are stored for later use to speed up future development processes (Gamma et al. 2002). Similarly, design patterns are used to store knowledge and experience and can take different forms: The range covers everything from how to tackle research issues in the early design process all the way to address particular features of a design (van Welie et al. 2003). Defining a way to categorize patterns supports understanding and discoverability for other designers when reutilizing the knowledge (Carlsson 2004). By packaging our research results in this manner, we intended to make it easier for others to practically employ the findings in applicable situations.

1.1 Purpose

The purpose of this thesis was to examine the current state of web applications by focusing explicitly on ATSs, as they present a complex set of elements and interactions while

having a specific purpose. From a general interaction design perspective, an appropriate user experience is warranted, where the resulting product does not just function correctly but is also enjoyable to use. In the initial phase of the design process, the founder of Software Skills and main stakeholder, Henrik Enström expressed the desire for an "open system which does not require permission to perform actions" and that it would be "better to show who did what" and be able to reverse mistakes if needed (Enström 2016). We therefore set out to explore designing a system targeting flat organizations rather than hierarchical ones, where usually operational activites have to be approved by higher ranking staff. Innovative solutions to support this approach were therefore requested in order to exhibit unique selling points and display novelty to clients, who represent the secondary stakeholder group. To satisfy both primary and secondary stakeholders, the system should address efficient collaboration by enhancing computer mediated communication and minimizing tedious manual labor.

ATSs are often used by many different users, with diverse roles and intentions; it is therefore crucial to support communication and sharing of relevant information in order to effectively coordinate the work towards a common goal. This thesis therefore explores the following challenges: How can software help to establish a common and democratic knowledge base? More specifically, how can software make it easy – or even enjoyable – to input information and keep it up to date? How can it support collective decision-making and teamwork? How can it ensure that each individual is given ample agency, while still warranting accountability? All of these also pertain to conveying awareness of other users within a system.

In aggregate form, the research question is thus:

How can applicant tracking systems be designed to facilitate awareness and efficient collaborative work across a team?

1.2 Deliverables

The thesis was planned to result in two main deliverables that are presented in detail in the results section in the report to follow: The theoretical effort was to focus on finding either patterns or guidelines that would help with informing the design of an ATS, with the conceptual, interactive, and visual layers that are part of this sort of dedicated web application. As a practical artifact, we aimed to conceptualize and designed an ATS that is underpinned by the theoretically conceived patterns or guidelines. This product was to be delivered in the form of detailed mockups for each view with an accompanying sitemap to visualize connections and dependencies. Vice versa, we intended to use the design to evaluate the findings from the theoretical track. Also, it should be mentioned that the concept and design we were to deliver was to be implemented up onto a certain point by a group of students as part of their bachelor thesis project in software engineering, also at

Chalmers University. As their project timeline coincided with ours, we were to work with them to accommodate their needs for new input so they had ample specifications to work with, and to avoid idle time.

1.3 Delimitations

We did not plan on adapting the design for responsive mobile use. While we acknowledge the importance of making all services available across multiple platforms, the time limitation allowed for the detailed development for one platform only, if done properly. It would however be the next step in the process if the project was to extend beyond the stated time period for this particular thesis.

We did not intend to design the external part of a job seeker applying to a job advertisement, because this is already part of the current website of Software Skills. However, we did plan to look at these existing elements and kept them in mind to integrate well with our proposed design.

Our thesis did not plan to include the implementation of the designs into code. The solution was to be programmed by a group of bachelor thesis students as first iteration, and further programming improvements and adding future functionalities were to be done by Software Skills to release the product.

2

Theory

A fair amount of academic research into the realms of design patterns has been undertaken in the past three decades. Numerous pattern libraries have been proposed and their merits investigated by several authors, and the discourse on patterns as a concept is an ongoing subject. Another significant component of our theoretical backbone is Computer Supported Collaborative Work (CSCW) (see Guerrero and Fuller 2001; David et al. 2003), which pertains to software that is meant to assist different people working together, and the associated technical and social demands that should be met by the system. The following section will first present a definition of what web applications – as opposed to websites – are, and subsequently delve into a discussion about the concept of patterns and introduce some relevant pattern libraries and their categorizations. Finally, a summary of the CSCW field and an overview over organizational structures will be presented.

2.1 Web Applications and Rich Internet Applications

Software applications that are built using web technology and are made available via web browsers are becoming a common phenomenon today. They are frequently referred to as Web Applications or Rich Internet Applications (RIA), as Vora (2009) terms it. RIA share many of the characteristics of traditional desktop application software. On the other hand, the difference between web applications and more traditional websites is sometimes difficult to ascribe clearly. In general however, websites are content oriented and designed to facilitate consumption of static information, whereas web applications depend on interaction, user input and data processing (Vora 2009). Thus, applications are recognized by their capability of presenting individual output data.

There are many reasons for designing applications for web rather than desktop. Bychkov (2013) reports that the ease of access is a significant factor, since users do not have to install the software on their computers and can access the content and functionalities

from a variety of devices as long as internet connectivity is available. Also, the development of applications for web holds an advantage in implementation, because it can be deployed to almost any user, as web standards are improving. Smith suggests that fewer versions have to be created, rather than unique versions for every operating system. It is also easier to introduce updates that will become available to the user as soon as they access the service anew, instead of having to first download and install the improved version. In terms of interaction design, it holds the advantage of having a lower barrier for use, because it does not require installation (Smith n.d.). Furthermore, most users today are familiar with web browser functionality and terminology, which makes the experience more intuitive. Despite these benefits of web applications, they also present drawbacks. One major challenge is security, as web applications are exposed to large numbers of users in the internet community, and thus expand the threat (Smith n.d.). Applications housed on the web require an internet connection and will often cause performance issues at low connectivity levels (Smith n.d.).

Vora sees one of the bigger challenges for interaction design of web applications to be the loosely coupled web architecture, which is the delay in communication between web browsers and servers. This limits the set of interactive controls, even if much of it can be solved with scripting languages today. A problem that emerges from this is the lack of design standards on the web, which can confuse users of different operating systems and platforms (Vora 2009).

2.2 Design Patterns

The idea of a pattern language to enable the reuse of established solutions to recurring problems is commonly attributed to Christopher Alexander et al., who published a book titled "A Pattern Language: Towns – Buildings – Construction" in 1977. In it, he proposes 253 patterns for the domain of city planning and architecture, covering larger concepts related to, for example, creating local centers, and how this can be achieved with a range of smaller patterns, such as *Activity Nodes, Shopping Street, Promenade,* or *Nightlife* (Alexander et al, 1977). Two years later, Alexander abstracted the meaning a pattern language can carry, which is eloquently summarized in the following paragraph:

> "As an element in the world, each pattern is a relationship between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain spatial configuration which allows these forces to resolve themselves. As an element of language, a pattern is an instruction, which shows how this spatial configuration can be used, over and over again, to resolve the given system of forces, wherever the context makes it relevant." (Alexander 1979, p. 247)

Since then, patterns have been applied to a wide range of other domains, from software engineering (Gamma et al. 1994 and others) to gameplay design (see for example Björk

et al. 2003). In the firstly mentioned field, it is documented that Ward Cunningham and Kent Beck were inspired to apply this concept to a project where they thus standardized recurring parts of the object-oriented programmed code and user interface in 1987 (Kruschitz et al. 2010). In the discipline of Human-Computer Interaction (HCI) – the umbrella of interaction design and therefore this thesis – the first design pattern language for user-centered interfaces was released in 1996 by Tod Coram and Jim Lee (Kruschitz et al. 2010). Since then, numerous pattern collections for design and HCI have been delivered, as exemplified further below in section 2.2.3.

2.2.1 Pattern and Guideline Distinction

Guidelines are a similar concept to patterns, however they differ from each other in several ways, mainly in their form and organization. As aforementioned, "patterns explicitly focus on context and tell the designer *when*, *how* and *why* the solution can be applied"(van Welie et al. 2001). Guidelines on the other hand, are usually more verbose and more linked to a specific scenario. They often propose several solutions or pointers of what to keep in mind when working on a given aspect. Patterns offer concise information about how to solve a specific problem. Together, they are embedded into a pattern language, where their relationships and interdependencies play an important role. Making the connections explicit helps to contextualize and understand a given pattern, eliminating improper applications of solutions. Guidelines on the other hand, do not include an organization, use context, or rationales – which leaves the designer, who wants to make use of the guidelines, with less instruction of how to apply these in practice (Carlsson 2004). We therefore determined that the concept of patterns is better suited to document the results of this master thesis project.

2.2.2 Pattern Collections in HCI

Alexander's patterns have been seminal for a wide range of applications, as for example, Apple's "Human Interface Guidelines" also cite Alexander as an influence (Borchers 2001). Prominent among the HCI design pattern collections is Jenifer Tidwell's "Designing Interfaces" (2005), which focuses on user interface (UI) design and is still being updated since the first version of the collection was released on the internet in 1998 (then under the name "Common Ground"). Other widely recognized collections are, for example, "Patterns in Interaction Design" by Martijn van Welie (Welie.com) and "Designing Web Interfaces" by Bill Scott and Theresa Neil (Scott 2009). Pawan Vora's "Web Application Design Patterns" (2009) presents yet another thorough look at problems and how to solve them with design patterns. He groups them according to specific areas within a web application, specialized apps and overarching requirements, and the names of those provide an indication of the collection's exhaustive range: forms, user authentication, application main page, navigation, searching and filtering, lists, rich internet applications, social applications, internationalization, accessibility, and visual design (Vora 2009).

The industry has also put forward patterns, for example "Web UI Design Patterns 2014" by Zuberi et al. (2004), but they differ on some counts, which is likely rooted in the fact that they are very feature-focused and largely keep to a low abstraction level. Instead of asking *why* a certain pattern should be used and *how* they relate, it is more of a listing of existing UI elements. Such industry examples are generally more up to date with their examples, but when compared to for instance Tidwell's collection (2005), the actual problems that are addressed are largely the same. Partially responsible for this is that pattern collections are optimal forms to maintain and expand over time, because of the defined underlying structure. Also observable in the undertakings of practitioners, is the adoption of the pattern concept for newer devices like smartphones. Due to the differing demands of screen size and touchscreen technology, separate pattern libraries have been put online that sport hundreds of screenshots of apps of, for instance, sign up screens, newsfeeds, or empty states (see for example Pttrns.com and Mobile-patterns.com). However, these industry examples are less descriptive and are more categories of different examples, where the user who wishes to use the solutions has to put more work into evaluating the underlying concepts, consequences, and danger spots themselves. Also, how the patterns relate to each other is not made explicit.

The popularity of design patterns, evident in the sheer quantity and continued effort into create exhaustive collections, may be attributed to the advantages they bring. For instance, the lack of standardization and limitations of controls mentioned in the previous section. Patterns can complement style guides to provide consistent interface designs, which improves the overall usability of web applications. Design patterns are abstract descriptions of recurring problems of use and lead the way for designers on when, how, and why a solution can be used. A pattern focuses on the problem within the context and can be applied in different situations to embody principles and strategies for desirable design (Wentzlaff et al. 2006). The idea is thus to "prevent reinvention of the wheel" time and again (Wentzlaff et al. 2006). However, if implemented in the wrong context they may cause more harm than good (Zuberi 2014). Therefore, it is important to document patterns systematically and they can be a useful timesaver in the design process. There are few guidelines of how to document design patterns, but they generally include a *problem summary, proposed solution, examples* to communicate the pattern and *usage* description of when and why a pattern is applicable for use and how it *relates* to other patterns on different abstract levels (Vora 2009).

Using patterns does not only improve the designer's productivity, but also improves the reusability of principles, leading to more consistent interfaces both within and across applications, which ultimately leads to users' familiarity of various interfaces. Patterns can also be particularly useful when having distributed design teams so that different solutions for common problems are eliminated. Patterns can also facilitate communication as they create a common terminology among designers. Moreover, it supports effective teaching of design principles to novice designers (Zuberi 2014).

As Vora (2009) notes, there is no standardized way for the makeup of a pattern – what Kruschitz et al. refer to as the pattern *form* (2010). Alexander's format for his architectural patterns (1977) consisted of an image, an introduction, a title that conveyed the essence of the problem, a more detailed description of the problem, a solution accompanied by explanatory diagrams, followed by a final paragraph that mentioned related patterns. Another form that was first introduced in the book "Design Patterns: Elements of Reusable Object-Oriented Software" by Gamma et al. (1994) and has been influential on subsequent pattern collections consists of: Pattern name and classification, intent, also known as, motivation, applicability, participants, collaborations, consequences, implementation, sample code, known uses, and related patterns. Tidwell's chosen form is more succinct and comprises the seven headings: Name, image, what, use when, why, how, and examples (2005).

Gamma et al.'s detailed form is appropriate for the domain of software engineering, as patterns for programming have other requirements than patterns for interaction design, where the exact implementation differs from project to project (Kruschitz et al. 2010). It is therefore necessary to deliberate carefully and be aware of what a pattern collection's purpose and context are, so that a suitable form can be determined accordingly. Generally, assigning a pattern a particular mould, helps standardize the overall collection and makes it easier to evaluate, compare, and apply the patterns.

2.3 Pattern Languages

A collection of patterns, if coherently structured and complete, make up a pattern language (Kruschitz et al. 2010). A design pattern language is an interconnected structure that is determined by how the patterns work together to solve complexes of design problems. When designing large systems, there can be many patterns at work to solve problems in different areas of the system. It is therefore helpful to categorize those patterns all the way from the broader social context in which an interactive system is used, down to the lower levels of detailed interaction. This aids with locating a specific pattern at a later point in time and to understand the problem space (Dearden et al. 2006). Furthermore, the process of structuring a pattern language ensures completeness (Hübscher et al. 2011). When Alexander introduced the concept of patterns in 1977, it was aimed towards the field of architecture, where scale – in terms of abstraction level – provides a useful organizing principle. However, in HCI the categorization is not as clear. While scale is important, designers may also need to address problems in terms of technology, tasks, and information. This makes the organization structure more convoluted (Dearden et al. 2006). To enable the use of such collections, it is essential to provide descriptions of a given problem and of how the patterns should be utilized to address those problems. A summary that includes how the patterns connect and a glossary explaining the terminology can improve the understanding of others even more.

2.3.1 Hierarchical Pattern Structures

There are a variety of ways to categorize patterns systematically within a language. The overall structure of the patterns is defined by how the individual patterns are related. One categorization method is Alexander's idea of scale, which can be adopted for HCI if *scale* is translated to *scale of problems* instead of geometry (van der Veer 2003). There is a *scale hierarchy* of problems when taking a top-down activity design approach. The top level is where designers gain understanding of the users, tasks, stakeholder requirements, and business goals and subsequently burrow down to specific actions in the interface.
The top-down approach uses three different pattern relationships: *Aggregation, specialization,* and *association.* Aggregation means that one pattern contains one or several other patterns. This is a form of a *has-a* relationship. For example, "getting overview" contains or *has a* "sitemap" and "breadcrumbs" pattern, making up the higher level pattern, as shown in figure 2.2 (van Welie et al. 2003). There are also specializations of patterns, which means that a pattern inherits the basic idea of a pattern and expands on it. This is called a *is-a* relationship, where one pattern is a more detailed and thorough version of another. For example, the "advanced search" can be considered a more sophisticated version of "simple search"(van Welie et al. 2003). The last relationship is called association and is neither a *has-a* or *is-a* connection, but rather a *related-to* affiliation. When designing a specific experience there might be several patterns applicable for one purpose. The patterns may be associated because they commonly occur in the larger context of the design challenge or simply because they are alternatives to solve the same kind of problems (van Welie et al. 2003). These can be summarized to *containing, complementing*, or *contradicting* relationship types.

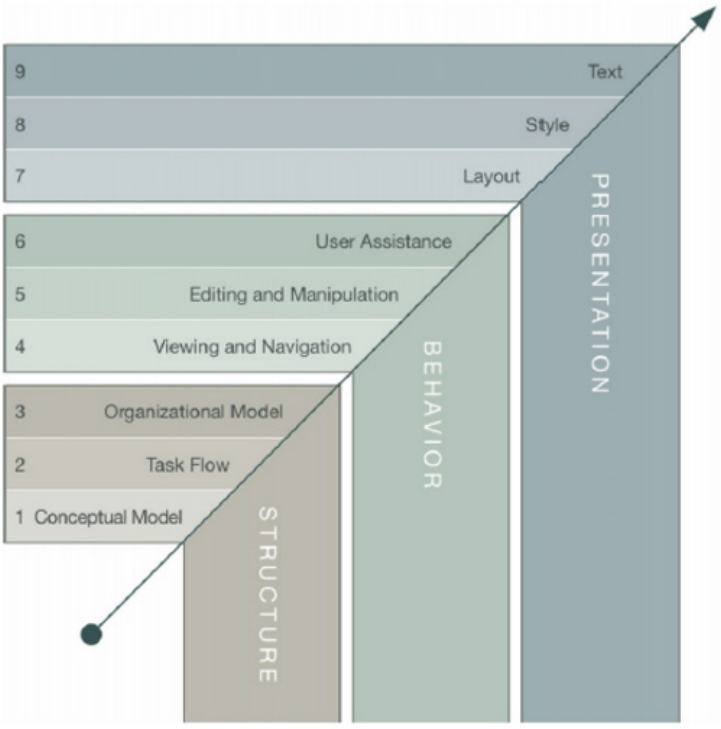


Figure 2.1: A pattern structure emerged from the design process proposed by van Welie et al. (2001).

A logical structure also supports designers to discover where patterns occur, so they can use them for their work. This also helps to identify possible gaps in the collection, consequently making it easier to contribute and expand. There are two major approaches of categorizing patterns: from the pattern creator's view, which typically is developed alongside the design process, as these are answers to design challenges as they are met (see figure 2.1); and from the designer's view, who wants to use the patterns, where it is often easier to have the structure arranged from a hierarchical perspective (see figure 2.2). The hierarchical scale (figure 2.1) is divided into different layers of patterns when going from high level patterns down to lower level patterns. The different levels used in this model – structure, behavior, presentations – are rough demarcations of typical stages encountered in design (van Welie et al. 2001).

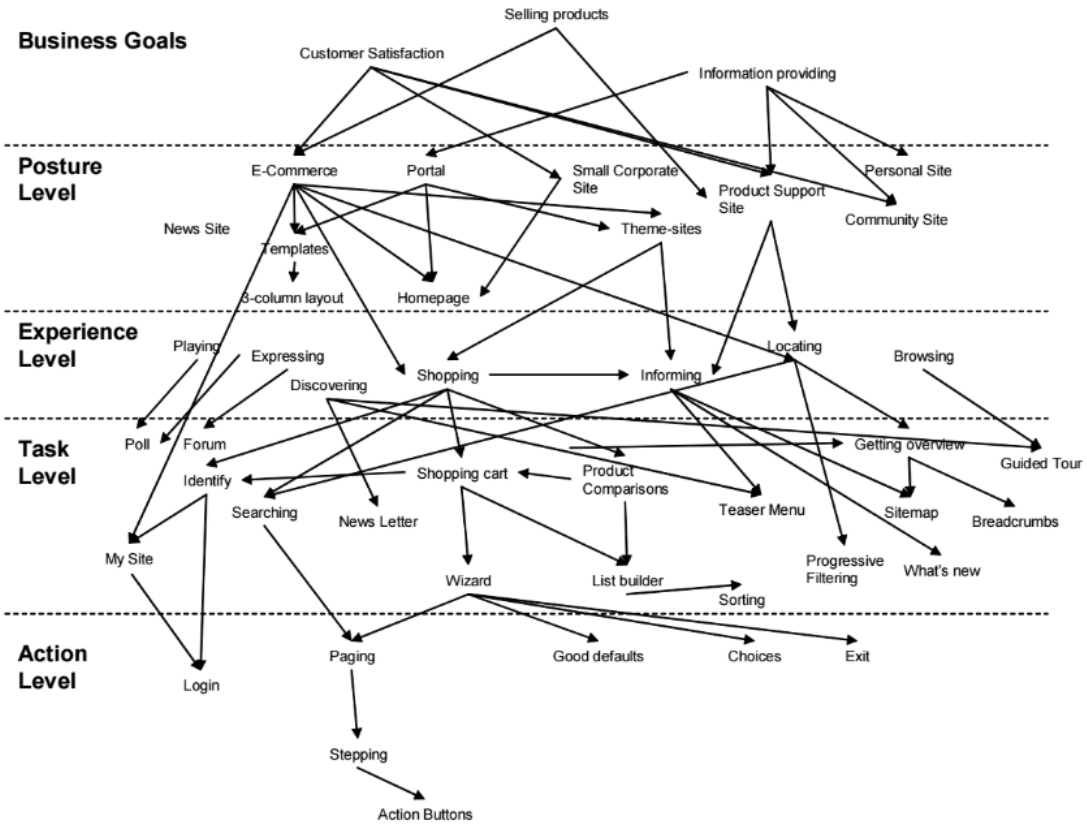


Figure 2.2: Example of how patterns for an e-commerce web site can be organized (van Welie et al. 2003).

Posture level patterns are the patterns that are required to achieve business goals or other personal or social goals. Design processes typically start off by informing the designers of what kind of site is needed to meet the goals. Websites with comparable purposes are often also similar in their structure and this is called the site's *posture*. Hence, posture patterns are: the site structure; the elements that typically make up the homepage; and the main experiences such a site offers. Use of lower-level posture patterns can support the definition of the platform since it is common for a site to have multiple purposes (van Welie et al. 2001).

A common subsequent step in the design process is to determine the main user goals and tasks. This involves the *user experience* (UX), and is how these goals and tasks are accomplished, how users perceive the site, and how this gives the user a sense of achievement (see Cooper 2014, as discussed further below in section 4.2 Goal-Directed Design). This level describes the *experience* and the patterns needed to support this experience. Furthermore, it is important to balance all the experience patterns to create a consistent impression and understanding for the user. Van Welie et al. (2001) considers a task pattern to generally describe a series of interactions needed to solve a problem. Whereas posture level and experience level set the context details for the site, task patterns are applied to translate those into concrete features. Finally, the *action level* holds the lowest pattern layer of building blocks. The action patterns describe specific uses of interactive elements that are only meaningful because they are part of features stipulated by the *task level* (van Welie et al. 2001).

2.4 Computer Supported Cooperative Work

As defined by Guerrero and Fuller (2001), Computer Supported Cooperative Work (CSCW) is a research area within the field of Human-Computer Interaction (HCI) (see for example Preece et al. 2015 for a definition of HCI). The research is focused on a type of software that deals with multi-user applications. These are sometimes referred to as groupware and are meant to help with completing shared assignments from separate workstations. Guerrero and Fuller (2001) characterize collaborative systems as being employable for situations where individuals work across various temporal and spatial settings. The challenge is to bring geographically distant people together and to coordinate asynchronous tasks, and

	Real time	Asynchronous
Communication	• Telephone • Video conferencing • Instant messaging • Texting	• Email • Voice mail • Blogs • Social networking sites
Information sharing	• Whiteboards • Application sharing • Meeting facilitation • Virtual worlds	• Document repositories • Wikis • Web sites • Team workspaces
Coordination	• Floor control • Session management • Location tracking	• Workflow management • CASE tools • Project management • Calendar scheduling

Figure 2.3: One way of grouping similarities and differences between different conceptual collaboration technologies (Grudin and Poltrock n.d.).

also to mediate the individual work among users of the system (Guerrero and Fuller 2001; David et al. 2003).

There are many ways of organizing CSCW software and figure 2.3 displays one example. Here, human behavior – which is a vital factor of collaboration – constitutes three categories: *Communication, information sharing,* and *coordination.* These strategies for collaboration occur either in *real-time* or *asynchronously*, which creates six groups of collaborative behavior (Grudin and Poltrock n.d.).

Communication is *how* users of a system share information with each other and shapes the style of collaboration within the system, which is largely dependent on the timing of communication. *What* information is shared within the system affects user behavior and interaction patterns. Information can be accessible at all times, unfolding over time or after certain events occur. Parts of data can also be limited for certain users either by the system or by other users. Coordination corresponds to *when* users perform tasks together (Lundgren et al. 2015). These actions can either be accomplished simultaneously or by linked efforts complementing each other's work. Coordination is important to ensure progress and minimizing redundant work (Lundgren et al. 2015, Dourish 1992). Examples of web applications for some of these are explained later in section 3.5.

2.4.1 Awareness in CSCW

One of the central concepts of CSCW is awareness. Awareness in CSCW is the understanding of the activity of others in order to work together (Wang et al. 2012). The research field of awareness in CSCW studies what users need to be aware of in order to coordinate and carry out interdependent activities (Tenenberg et al. 2015). The approach of how to maintain user awareness differs widely depending on the type of groupware. Real-time distributed work might require direct access to other users' actions in order for efficient collaborative work. Meanwhile, asynchronous co-located applications could seek to protect users' real-time actions, since knowing where others are working can constrain one another (Tenenberg et al. 2015; Gross and Koch 2006; Gutwin et al. 1996). As Schmidt explains, *awareness* can carry a lot of different meanings and "depending on the context it may mean anything from consciousness or knowledge to attention or sentience, and from sensitivity or apperception to acquaintance or recollection" (Schmidt 2002). In order to concretize awareness in CSCW, Dourish and Bellotti offer a more framed definition:

> "Awareness is an understanding of the activities of others, which provides a context for your own activity. This context is used to ensure that individual contributions are relevant to the group's activity as a whole, and to evaluate individual actions with respect to group goals and progress. The information, then, allows groups to manage the process of collaborative working"(1992, p.107).

Category	Element	Specific questions
Who	Presence	Is anyone in the workspace?
	Identity	Who is participating? Who is that?
	Authorship	Who is doing that?
What	Action	What are they doing?
	Intention	What goal is that action part of?
	Artifact	What object are they working on?
Where	Location	Where are they working?
	Gaze	Where are they looking?
	View	Where can they see?
	Reach	Where can they reach?

Category	Element	Specific questions
How	Action history	How did that operation happen?
	Artifact history	How did this artifact come to be in this state?
When	Event history	When did that event happen?
Who (past)	Presence history	Who was here, and when?
Where (past)	Location history	Where has a person been?
What (past)	Action history	What has a person been doing?

Figure 2.4: Workspace awareness relating to the present (top) and the past (bottom) (Gutwin and Greenberg 2010).

Further, they assert that awareness is central to coordination efforts and information exchange, both of which are vital parts to collaborate effectively (Dourish et al. 1992). Figure 2.4 above contains examples of attempts to clarify and structure various forms of awareness that ought to be addressed in groupware (Gutwin and Greenberg 2010).

3

Recruiting, Systems, and Collaboration

This section introduces the aspects that are relevant for the research question from a practical standpoint: It begins with a short history on recruiting as a vital business process, then formally defining what an ATS entails, followed by an introduction to the company and the connected stakeholders. This also includes an outline of how the existing workflow of Software Skills is structured. Subsequently, the results of benchmarking other ATSs will be described, followed by the presentation of some collaborative web applications.

3.1 Brief History of Recruiting

By briefly outlining the steps that led to the current status quo of how new employees are hired to companies, some of the values and motivations of recruiters and human-resource departments may be uncovered. Generally, recruiting can be categorized into two sections: *Active* – where candidates actively seek for employment, and *passive* – where candidates are scouted for a specific position (TalentBin n.d.). During the Second World War, as a significant portion of the workforce left their jobs to join the war effort, either as soldiers or in other capacities, these positions were more difficult to fill. This led to the emergence of employment agencies that attempted to fill this vacuum by searching for *passive* candidates. Subsequently, the creation of resumés gained in popularity as it was an effective way to summarize one's skills and experience (Sigmund n.d.).

In the pre-computer era, newspaper listings and postings on bulletin boards were the most common way to advertise job openings. On the *passive* side of recruiting, the specialized agencies had a lot of resumés stored in physical archives that were tedious to use and maintain. The telephone was the fastest tool available for finding passive candidates. With the onset of the computer age, some of this data was digitized – leading to early forms of Applicant Tracking Systems – and email was added to the range of possible communication channels for outreach. From the mid-nineties, online services were made available on the world wide web, such as Craigslist and Careerbuilder in 1995, or Monster in 1999

(TalentBin n.d.), which acted as widely accessible job boards for *active* searchers. With the strong growth of economy close to the turn of the century, the demand for qualified workforce increased rapidly (Thomas 2000), and soon these platforms were used by professionals as a means to search for suitable candidates. This widened the search range and established the concept of "headhunting". With the introduction of professional social media platforms – most renowned among them Linkedin that was launched in 2003 (LinkedIn) – recruiters and potential employers now had access to millions of profiles of *passive* candidates. Nowadays, the recruitment landscape has expanded to include meta search engines that are specialized to find possible fits for positions by crawling all sorts of social media sites, professional and special interest platforms (TalentBin).

Some believe that this development is expected to continue, so much so that paper resumés will soon be a thing of the past and all data will be digitized. The various online presences of both *active* and *passive* candidates, as well as potential employers, will be the main go-to sources to find and select both future employees and workplaces (Fallon 2016). Therefore, smart systems are needed to aggregate and store this information, and make it accessible in an effective way.

3.2 E-recruiting and Applicant Tracking Systems

E-recruiting is defined by Holm (2012) as "the organisation of recruitment process and activities, which, by means of technology and human agents, facilitate time- and space-independent collaboration and interaction in order to identify, attract, and influence competent candidates" (p. 245). Lee (2007) explains *e-recruiting* as benefiting from making "use of a central database and an array of Web-enabled integrated applications" (p. 82).

Compared to the many articles that have been written on the effectiveness of electronic and online means for job and candidate searching (for example Bartram 2000, Feldman et al. 2002, Furtmueller 2010, Geutal et al. 2005, Koong 2002, Thompson et al. 2008), the exploration on how digital tools can support both job seekers and recruiters *after* the first contact has been established, may appear not to be as extensive. However, the diversity of angles discussed on this matter illustrate the multidimensionality associated with e-recruiting. These specific aspects include for example: issues of digital resumés (Furtmueller et al. 2011), candidate relationship management (Holm, 2012), and architecture of holistic e-recruiting systems and the integration of front-end job advertisements and back-end human resource management systems (Lee 2007).

Holm scrutinized the work procedure of three big recruiting companies in Denmark, and based on this delineated the standard recruitment process to be split up into different main tasks, which branch out into various subtasks respectively (shown in figure 3.1): Firstly, the *identification of candidates*, which is the stipulation of requirements that satisfy

the prospective job position. This entails the subtasks of developing a job description and determining "the appropriate pool of applicants" (Holm 2012, p. 244). Secondly, the task of *attracting applicants* is made up of identifying suitable recruitment sources – such as newspaper ads or LinkedIn – and subsequently initiating the outreach. Thirdly, the result of said outreach will need to be addressed by *processing incoming applications*, corresponding to the subtasks of cataloging and registering the applications themselves, and reviewing the information contained. This task also involves making a pre-selection of eligible candidates. Lastly, *communication with the applicants* takes place, meaning notifying applicants of their status (including not-shortlisted ones) and setting up interviews with the chosen ones.

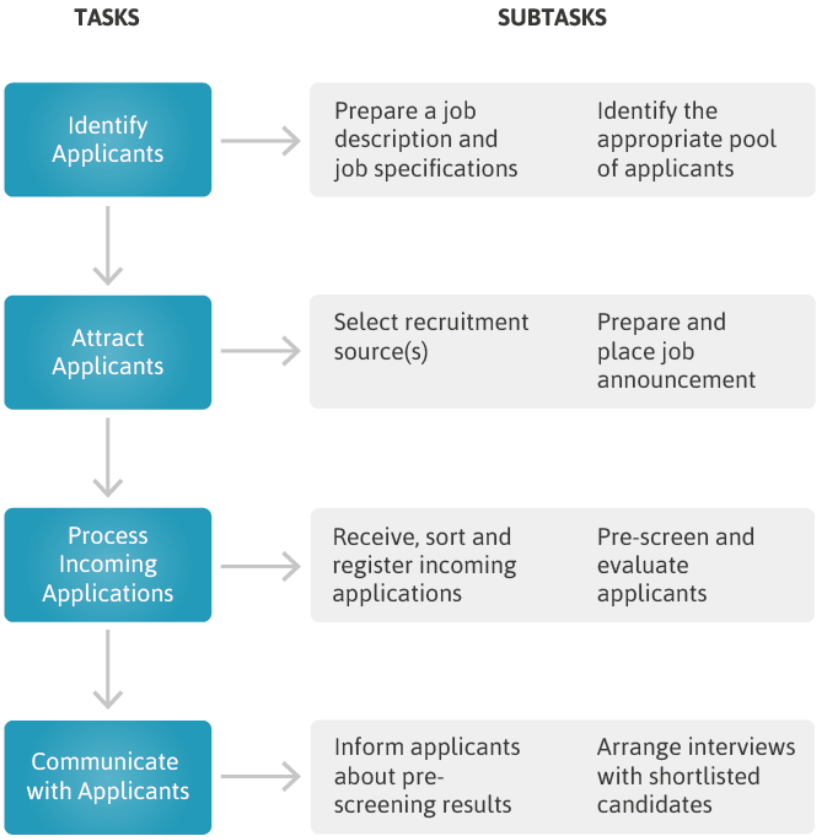


Figure 3.1: Recruiting process based on Holm (2012).

Holm found that all of the companies in the study relied on electronic automation tools for the tasks of *attracting applicants, processing of incoming applications* and *communication with applicants* – a substantial part of the hiring process. These results also validate the importance of such tools as they have become an integral part of the recruiting landscape. Applicant tracking systems, as these tools are commonly called, help reduce time spent analyzing applicant information, as they help to identify and highlight suitable applicants at any stage of the hiring process. These systems are often purposed to collect all job applications for a given position, storing all submitted documents and information for the applicants, and keeping track of communication and interviews in the same place (Lee 2007). Furthermore, it is common to store data about the applicant to be able to include them in future job openings, but also data from the various hiring processes to

gain information to further streamline the recruitment process. ATSs vary in size, features, and complexity, but they often incorporate options to set up rules that filter applicants automatically, typically by parsing applicant resumés to identify and weed out unsuitable applicants, which is determined by keywords in the resumé text.

Essentially, companies can benefit to varying degrees from using an ATS in their recruitment process depending on their former solution and method of hiring. Using an ATS can trim the cost per hire due to the reduced administration time and at best enhance the quality of hire (Lee 2007). Other motives may be to improve communication internally and thus facilitate better workforce planning and budgeting.

3.3 Software Skills

Software Skills is a recruitment agency located in Gothenburg, Sweden, founded in 2012. Their initial business idea was to specialize in hiring people abroad and bring new talents to Sweden. However, the market for such a service proved limited because most firms prefer Swedish speakers. A new, unique selling proposition has been developed in the meantime: programming tests that are used to evaluate an applicant's skills for various programming languages. Today, Software Skills provides personality tests, logic tests, and programming tests. The tests are both used in-house and are also sold to external companies for use in their recruiting processes.

Henrik Enström, the founder of Software Skills, describes his company's purpose as identifying and testing talented developers and other IT experts for their client companies. As analyzed during our pre-study, Software Skills has also developed an in-house applicant tracking solution that is comprised of third party services – namely Trello (Trello [1]) and Google Drive (Google Drive n.d.). Trello is used to manage the applicants throughout the hiring process, while their files, such as CVs and resumés are automatically stored on Google Drive. This has been offered to customers, however clients have shown hesitation to opt in, as it is not an integrated, holistic system built by Software Skills from the ground up, but an adapted system. To further expand the service segment, the company aspires to provide a coherent ATS alongside their recruitment services.

3.3.1 Current system

The current solution works reasonably well for Software Skills internally, but with the expected rise of candidate numbers it is going to reach its limits soon, most notably because the manual labor intensive operation and having to rely on human memory often. However, the current system's functionality and structure are valid as they have grown organically with the recruiters' routines and experiences. The system is based on Trello and Google Drive and makes use of their APIs that are connected to Software Skills' web platform for job postings. Posting a new job automatically adds a new Trello

board. Similarly, a new application automatically creates a Trello card with the applicant's name, which is subsequently dealt with manually by the recruiters. When the candidate uploads files such as their resumé, a corresponding folder structure is created in Google Drive. In turn, Trello sends information to Google Sheets that is the basis of the statistical data of the recruitment process, such as the source of the candidate (headhunted or job ad respondents), and if they were hired or not.

3.4 Competitor Applicant Tracking Systems

As can be seen from the history of recruitment, the act of hiring is one of the most vital undertakings of any company. It is no surprise therefore, that many have endeavored to devise tools that support this process and Software Skills is not alone in its aspirations. These tools range from smaller systems with limited capabilities, to very feature-laden applications that accommodate a large number of candidates, for numerous positions, to be handled by a big team with different levels of authority. Examples of software that are built to manage the application process for small to medium sized companies include the ATSs Jobylon, Teamtailor, and Workable.

Jobylon is a Swedish startup, whose core functionalities are creating and sharing new job openings, and subsequently oversee the hiring stages and applicants for the respective job. There are some operations that allow for collaboration among a team of recruiters. The search and filter options – needed to find applicants in a given list – and integration with other third party service providers are basic but adequate. Their unique selling proposition (USP) is to easily create appealing job postings that speak to job seekers in stories, instead of just a "list of requirements" (Jobylon n.d.).

A somewhat bigger system is provided by Teamtailor, also a Swedish startup company, who – apart from above listed core functionalities – provide features such as statistics and generating reports, and calendar integration (Teamtailor n.d.). Most notably, they devote more options that pertain to user roles and respective access rights, automating communication, and bulk actions. Both Jobylon and TeamTailor exhibit a contemporary design, with clean interfaces, generous text sizes and click areas, bright colors, and responsive design. Since they are both developed in Sweden and have most of their customers in the region, these two systems are some of the most serious competitors for Software Skills.

Workable is a product conceived in Greece, and we found that it is comparable to Teamtailor in terms of size and purpose. Apart from the aforementioned functions, it supports more small scale activities for collaboration, such as voting on the suitability of a candidate with thumbs up or down that is tied to the specific team member, who has given the vote. Also, it offers a more seamless integration with online job boards such as Monster or Career Builder (Workable n.d.).

There are many other systems, a major share of them originating from the United States, such as Jobvite, Greenhouse, CATS, or Jazz. A few are based in other places, such as RecruiterBox in India. Jobvite is the most widespread in use of these, but all of these are very powerful systems that include many additional features. For example, parsing resumés to extract information from uploaded documents, integrated options for sourcing candidates from social media and professional platforms, intricate and customizable search filters, or built-in video service for online interviews (Jobvite n.d., Recruiterbox n.d.), which makes them able to handle large numbers of applicants, but can also make them laborious to use.

3.5 Collaborative Web Applications

In this section, we take a broader perspective and examine how web applications – that are not ATSs – solve issues related to collaborative work by means of their functionalities. According to the matrix introduced in section 2.4 (see figure 2.3), where the relevant fields are *asynchronous information sharing* and *asynchronous coordinatio*n, the following examples are sorted according to the following sub-categories that we see as significant: *Document Repositories, Wikis, Team Workspaces, Workflow Management, Project Management,* and *Calendar Scheduling* (examples shown in figure 3.2).

3.5.1.1 Document Repositories – Example: Dropbox, Google Drive

Document Repositories are applications that facilitate the storage and organization of a variety of files, and since these are stored on servers, it is simple to make these available to multiple users. Many services exist in this category, such as Droplr, Microsoft's OneDrive, or Box, which places heavy emphasis on the security of files. However, the most used services among private individuals, but also for professional purposes, are Dropbox and Google Drive. Usually parties are invited to folders via their email, and then have access to the contained documents. These can then be viewed and downloaded, and depending on the permissions, can also be moved, deleted and so on. To keep track of these changes many services use activity logs that record which actions have been taken and by whom. One key advantage is the file synchronisation, meaning that all involved individuals have access to the same instance of a file, and thereby avoiding discordant information levels.

3.5.1.2 Wikis – Example: Wikipedia, Wikia, Confluence

A wiki by definition is a website where users may have roles of both author and editor for the overall site structure (i.e. adding, linking, and deleting pages), as well as changing the content on these pages. This concept was first implemented in 1995 by Ward Cunningham in his WikiWikiWeb system, that he developed as a documentation tool for programmers (Ebersbach et al. 2008). Since then, his idea has spread due to it making content editing simple and therefore possible for anyone with internet access. This in turn allows users to collaborate asynchronously and facilitates the democratization of knowledge and

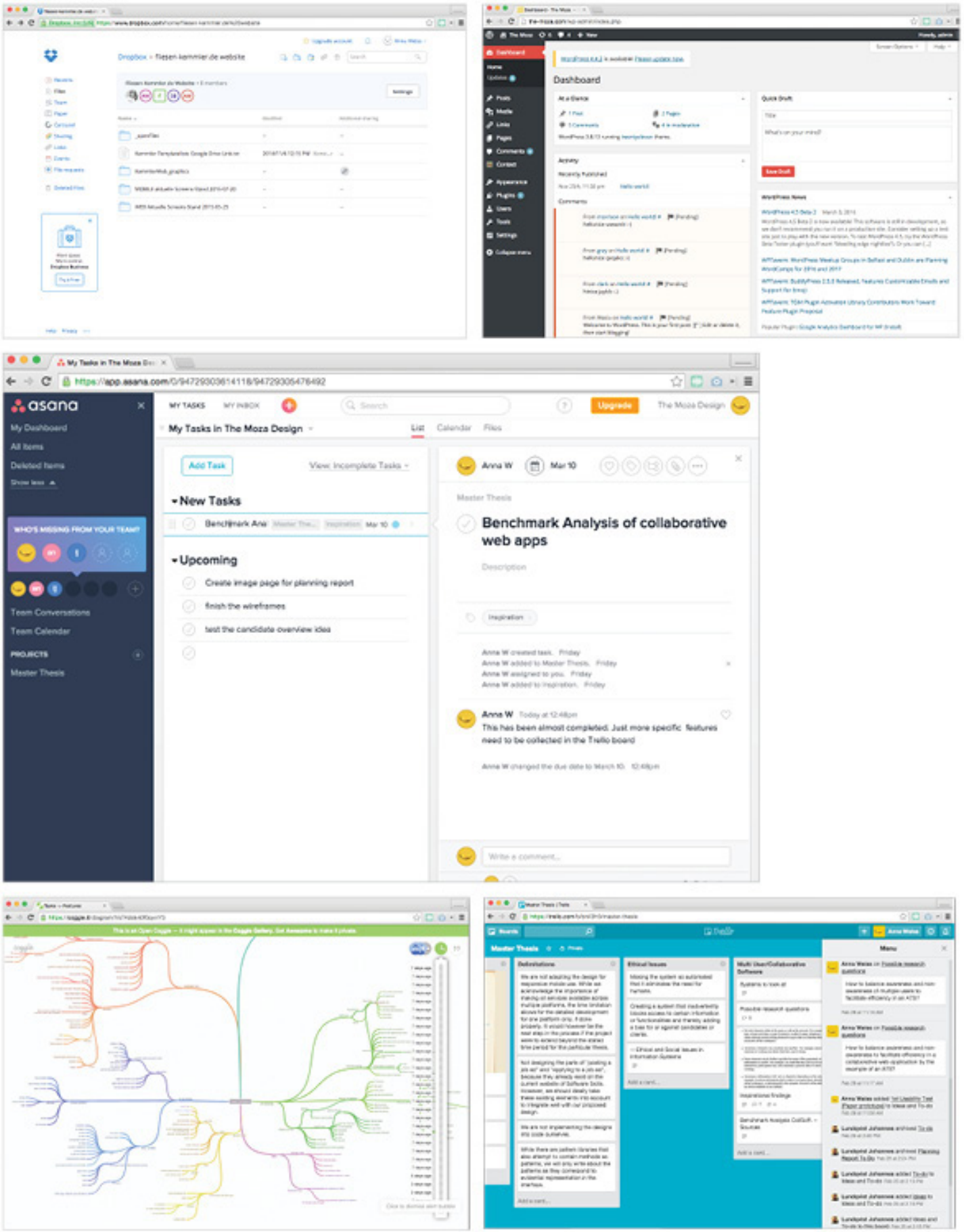


Figure 3.2: Collaborative web apps Dropbox, WordPress, Asana, Coggle, Trello.

expertise. Wikis can be either public for anyone to use, others are private and can mean a way to structure and access intelligence that belongs to a specific company, for example. The most renowned instance is Wikipedia, the open encyclopedia, where more than 38 million articles in over 250 different languages are available (Wikipedia 2016).

There are services that provide the technical frameworks to support in creating one's own wiki, such as MediaWiki, Wikia, Wikispaces, or Confluence, which is a professional product most often used to maintain information on intranets. These systems are specifically designed for collaboration and according to Ebersbach et al. (2008) commonly carry the following capabilities: *User roles*, often distinguishing in access rights according to reader, author, wiki admin, and web admin; *History*, where all previous versions of an article are preserved; *Recent Changes*, a dedicated page that tracks changes so admins can monitor edits more conveniently; and finally *SandBoxes*, which are areas for new users to test their new found powers without actually affecting the structure.

3.5.1.3 Team Workspaces – Example: Google Docs, Figma, Coggle

Team workspaces can be loosely described as productivity applications where individuals can work on one and the same file either in real time or turn based. This is called *semi-synchronous* by Dourish and Bellotti (1992). The instantaneousness in which the information has to be relayed poses a particular challenge for programmatic implementation, but also pertains to interaction design in how the users are represented to each other in the system and what they can do at a given time (Imine 2009). Google Docs, an online text editor enables simultaneous editing from multiple machines – as do its brethrens Google Sheets and Google Slides. To this end, the web applications show the changes as soon as they occur – to the human eye anyway – with a different color and icon assigned to each user currently online. Sideline discussions about the work to be done can be conducted in a separate chat window. By clicking a user's avatar, one can jump to this user's current location in the document and see what they are working on at the moment, which pertains directly to awareness as introduced in section 2.4.1.

Other features pertain to asynchronous collaboration ambitions, for example the commenting function, where parts of the document can be highlighted and notes be added, which in turn can then be responded to or resolved by others. By clicking on "all changes saved in Drive" the user can access the revision history that reveals the executed changes – again color coded according to the person responsible. Also, Google Docs allow users to switch between three different modes: The normal editing mode, the suggesting mode, and the viewing mode. The suggesting mode enables users to propose changes, that can subsequently be approved or rejected, instead of becoming instantaneously a part of the text. Other team workspace applications are for instance Coggle, an online mind mapping tool (Coggle), or Figma, a vector-based design editor that will roll out its simultaneous editing feature in 2016 (Figma).

3.5.1.4 Project Management – Example: Asana, Basecamp, Wrike

A project management tool can cover many different aspects of a project, helping with planning and a sorts of information sharing requirements. One such product is Asana. Various teams can be set up that work on different projects. Features such as assigning tasks to oneself or to team members, managing deadlines, calendar functions, and creating performance reports are incorporated. These are the core project management components. However, additional demands are also addressed by enabling attachments and file management; Communication is covered by allowing users to add comments on tasks, and features like private message inboxes and open discussion threads (Asana). A similar product is for example Basecamp, with its six pillars representing message boards, real-time chat, automatic check-ins (automatically showing which tasks people are currently working on), to-do lists, file storage, and centralized schedule (Basecamp). Basecamp's USP is its "Clientside" feature, which channels all input to either the team internal communication and planning, or to the branch that the client is privy to as well. The underlying idea is to keep the client in the loop and all correspondence in one place, while ensuring that sensitive material is not shared unintentionally. Wrike is yet another project management tool, that – among the standard functionalities – offers customizing task and workflows, which is strikingly similar to what corresponds to adapting the hiring process in a number of benchmarked ATSs. Wrike also puts emphasis on real-time reporting, especially visualizing completed, active, and overdue tasks of everyone – "the team temperature" as they call it (Wrike).

3.5.1.5 Calendar Scheduling – Example: Doodle

Many of the project management tools contain calendar options, but there are services whose very purpose is to collaborate on scheduling matters. Well-established Doodle allows to create polls for determining a time where individuals are available for a given commitment, by simply creating a matrix of time slots versus relevant parties, where said parties can then mark their vacancies (Doodle). The resulting overview can help reach a democratic consensus.

It is also valid to mention that there are products that cover many of these sections under one umbrella. As seen above, many project management applications do so, but also solutions like eXo Platform, which describes itself as a "social collaboration solution" (Doodle) and that can be set up to perform as a company portal, with focus on what they label knowledge management. This "enterprise social network" offers components such as collaboration workspaces, forums, wikis, FAQ sections, calendar, document management, and be adapted to work as a CMS (Doodle).

Likewise, some applications clearly accommodate collaboration (= *communication, information sharing,* and *coordination*) among several users *asynchronously*, but were not built specifically for this purpose or they do not clearly fit into one or the other category. Trello is a widely known project organizing tool, whose basic structure consists of boards, which

can be filled with lists that in turn contain cards. The premise is to be able to set up and adapt one's desired project architecture intuitively, through drag and drop interactions. The smallest entities – the cards – can hold a multitude of information types, such as descriptive texts, images and other attachments and links, checklists, and comments. They can be endowed with color labels, due dates, polls, or set to visually convey aging when content has been neglected (Trello [1]). The actual collaborative aspect is addressed through actions relating to members, where for example comments can be addressed to individuals, or turn on notifications for specific cards, to learn about change in real-time. Trello Teams – available to subscribers of paid plans – offer team pages to manage user permissions and assign certain boards to pertaining groups. While the UI displays a minimum of action buttons to make it simple for the beginner users, Trello is a comprehensive tool, whose expert users may access powerful functionalities, for example special operators to refine searches, or markup options to format text contained in the cards (Trello [2]).

Also Content Management Systems (CMS) are generally intricate programs that support the maintenance of a website by several parties. One widely-used example is WordPress. Originally a blogging platform, it enables the handling of content and even structure of an installed website. An essential feature is controlling user roles, as in many cases many individuals have access to the backend, and while some may only be tasked with editing text and media content, others need to be able to adjust design or functionality. WordPress has also grown to incorporate features like version control (WordPress).

4
Methodology

In the interaction design community and similar fields, many researchers and practitioners alike have concerned themselves with how to best organize the process to eventually arrive at a valid design solution. Processes like the Human-Centered Design (HCD), Goal-Directed Design (GDD), Activity-Centered Design (ACD), and Participatory Design (PD) have been developed and copious amounts of methods experimented with. It is therefore not surprising that these many schools of thought diverge – and even contradict – on numerous counts. To navigate this meandering terrain, a brief introduction is given to both HCD and GDD, from which we will take inspiration for our own design process outlined in the planning chapter. This is followed by brief descriptions of a wide range of methods that correspond to the stated process.

4.1 Human-Centered Design

Interaction design related fields, such as Usability Engineering and Human-Computer Interaction, important in the 1980s and 1990s (Williams 2009), founded the basis for what is commonly called Human-Centered Design or User-Centered Design. Since then, the ISO standard titled "Human-Centered Design for Interactive Systems" (this title was first used in 2006 (ISO 2016)) has been put forward. In it, common standards for this approach to design have been outlined, and in summary the principles stated are: The Human-Centered Design (HCD) approach should have comprehension of "users, tasks and environments" (ISO 9241-210, 2010) at its core, meaning the users are involved during the entire process and evaluation is informed by user feedback; the development phases occur iteratively; and finally, the design team should encompass a wide scope of skills and backgrounds. The motivation for employing HCD is to develop products that are easy-to-use and effective, thus attracting a wide and loyal customer base. Apart from being popular and therefore financially successful, a product or service that is designed with awareness of the user will also reduce stress, errors, and resulting support costs.

With these advantages in mind, many practitioners have sought ways of how these principles can inform planning this iterative process and choosing appropriate methods in the wild. Among them is the renowned IDEO organization, who has identified and named three different stages: *Inspiration, ideation, implementation* (IDEO.org 2015). The *inspiration* phase incorporates research activities and can contain such methods as different kinds of interviews, immersion and ethnographic undertakings, and approaches to analysing various existing solutions. The learnings from the first step will flow into the subsequent *ideation* phase, where various brainstorming methods take the center stage. After narrowing down the collection of generated ideas, it is time for the *implementation* phase where some of the ideas are implemented as lo-fi prototypes and then tested as early as possible. After *evaluating*, the stages will be run through several times to refine the solutions as best as possible. The LUMA Institute has also put forward three stages that each corresponds to a set of suggested methods: *Looking, understanding, making* (LUMA Institute 2012).

4.2 Goal-Directed Design

Another school of thought is Goal-Directed Design, whose establishment is largely due to the efforts of Alan Cooper and his associates and is largely based on practical experience rather than theoretical underpinnings (Williams 2009). One of the foci is to research the topic in question and affected users thoroughly and subsequently extract their goals on three different levels: The most immediate layer is termed *Experience Goals* and pertains to the emotional response a user exhibits when engaging with a product. This visceral part of the experience asserts itself in phrases like "feeling smart and in control" or "feeling cool or hip"(Cooper et al. 2014, p. 76) and is contained in the product's look and feel. The middle layer, the *End Goals*, is probably the most tangible as it refers to the immediate tasks that the user wants to complete. "*Life Goals* represent the user's personal aspirations that typically go beyond the context of the product being designed."(Cooper et al. 2014, p. 77) Understanding these underlying desires allows to create a stronger bond to the product, which can be addressed through the overall design and branding. In Cooper's way of thinking "how the user wants to feel", "what [she] wants to do", and "who [she] wants to be" are the three questions that can inform the design and make it a more successful product (Cooper et al. 2014, p. 76).

In practical terms, the proposed process is divided into six parts that in turn have specific sub-activities. First is the *research* phase, which starts with determining the project scope, followed by auditing already existing components and a review of relevant literature, then stakeholder interviews, and concluding with user interviews and observations. These findings are then used in the second phase, *modelling*, which entails the creation of personas, and other models, for example workflows. The subsequent *requirements definition* phase consists of determining context scenarios and requirements, according to the needs of the previously identified persona. Then it is time to tackle the *design framework*, which incorporates defining elements, the experience framework, and key path and validation

scenarios. This is followed by the *design refinement* phase, where all the details of the design are fleshed out. Finally, the last phase design support is initiated, where modifications according to practical concerns from the technical or stakeholder perspective are accommodated (Cooper et al. 2014).

4.3 Methods

The GDD process is split up into more distinct phases than what is defined for the HCD process, and it places less emphasis on iterating through all stages several times. Instead, the methods are strung together in a way that is meant to support informed decision making earlier with less of the trial-evaluation-iteration that is favored in HCD practice. This is perhaps due to the fact that GDD seems to formally incorporate more activities that pertain to meeting the business requirements of a given project. However, since the HCD process is more flexible it can certainly be planned in a way to accommodate a business-focused course as well.

However, in terms of the specific methods used, these two approaches are not so different after all (Williams 2009). The following methods are a compilation from different sources, and ordered in the sequence they could be conducted (but do not have to be). They were selected to showcase a variety of different techniques that help gain insights, create new ideas, implement solutions, and evaluate their validity. Correspondingly, they are grouped according to *inspiration, ideation, implementation,* and *evaluation.*

4.3.1 Inspiration

As Cooper argues, it can be very helpful to interview what he calls Subject Matter Experts (SMEs) at an early stage in a design project as the designer can tap into this rich resource of expertise in the given area. Furthermore, it is also valuable because they are often – and in our instance they are – expert users, meaning they are skilled at interacting with the product, in this case the ATS. Cooper distinguishes between interviews with stakeholders, SMEs, users, and customers. These interviews can help establish a common language between the intended users and the design team, and the personal involvement can help with adopting the new system. To establish the different user goals (for the GDD process), Cooper suggests interviewing both present and possible future users (Cooper et al. 2014).

Apart from the potential participant groups, also the interview approach has to be selected. Baxter et al. distinguish between *unstructured, structured*, and *semi-structured* interviews. Unstructured interviews are open-ended and provide the opportunity to figure out the 'lay of the land' and to find out nuanced – qualitative – information, because they are suitable for follow up questions. Structured interviews can provide quick answers to specific questions, that will be consistent across a number of participants, which makes it efficient to evaluate and results in quantitative data. A blend of these two types is the semi-structured

interview, which incorporates both standardized questions, and possibilities for improvisation to investigate further. Accordingly, this can yield both quantitative and qualitative information (Baxter et al. 2015). When preparing interviews it is especially important to avoid leading questions. Nielsen calls this the query effect: "People can make up an opinion about anything, and they'll do so if asked. You can thus get users to comment at great length about something that doesn't matter" (Nielsen 2010).

Interviews yield good results in terms of providing an introduction to the field and reveal explicit and obvious matters, observing the users can call attention to implicit aspects. This can often also identify issues with the current system that the users have already grown accustomed to and have established workarounds that have become part of their daily routines. By watching the product being used (accompanied by *thinking-aloud* or narrating the actions) the designers can discover the system states and behaviors that are odd or complicated, and subsequently process these insights into useful pointers into what the current solution lacks (Cooper et al. 2014). *Pure observation*, or *fly-on-the-wall observation*, is a method where the designer tries to blend into the background as best as possible, with the intention of not disturbing the environment and the protagonists in it, so as to be able to experience the situation in a preferably pristine state (Baxter et al. 2015). This is different in the *contextual inquiry* method, which can be described as a user interview in the natural habitat. Here, the designer is in fact supposed to engage in a dialog, and can ask questions about specific artifacts or procedures. It is important however, to not make this a show-and-tell session, but to encourage people to engage in their regular activities (LUMA Institute 2012).

Personas are synthesized from the preceding qualitative research results and are essentially constructed characters that exhibit traits that align with members of the actual target group. These attempt to define the persona's behavioral patterns, goals and motivations, skills and experiences, relations to other people or entities, and any demographically related information. The thus conceived fictional characters can then be referred to throughout the remainder of the process to inspire, inform, and evaluate design decisions. Evident from the presented GDD process and also from his domain prominence in this regard, Cooper is an outspoken advocate of making personas an integral part of the design process. He argues that it can help with feedback and keeping the design coherent. Also, it makes it easier for the design team and clients to explain decisions and prioritize according to the user requirements (Cooper et al. 2014). As much as Cooper, and others (Adlin et al. 2010, Preece et al. 2015, Nielsen 2013) see personas as the panacea and allow them to play a major role in their process, there are also voices criticizing this concept.

Chapman et al. (2006) identify the "methodological weakness" that it is inherent with personas: There are no criteria or standards that could validate whether a presented persona's characteristics are appropriate or not. "How many users does a given persona describe? How important are those users for the project? If several personas represent several slices of the population, how many people fall between those slices?"(Chapman

et al. 2006, p. 634) Practitioners are ideally supposed to synthesize their personas from empirical data, but – as Chapman et al. argue – these "scattered data points" can inform the development of personas only so far, and are in turn not verifiable without a tremendous effort. Furthermore, to warrant a comprehensive coverage of all characteristics, usually a whole set of personas is required (Preece et al. 2015). Chapman et al. conducted a quantitative evaluation that correlated the number of attributes of a persona to how likely it would be to match real groups of people. They conclude that the more characteristics are stipulated, the less probable it is to hold true in the real world (Chapman et al. 2008), and since these user models are commonly endowed with a number of features and surrounding narratives to provide a *holistic* picture, it is difficult to couple these to real world users. To attempt and corroborate these models through aforementioned gathering of empirical data can quickly become a time and money consuming feat (Flaherty 2015).

Others take a higher-level perspective on the whole issue of how to incorporate the user into the design process, to which the persona approach belongs. Redström warns against relying too much on methods that model members of the declared target group. 'Declared' is the appropriate word, because these are all constraints stipulated by the marketing or design team, but what about the humans who use a product or service, who fall outside of this circle? Redström states that "a 'user' is something that designers create" (Redström and Johan 2006, 129), not an entity that exists outside in the real world, which ties in with what Chapman et al. found in their aforementioned quantitative analysis. Furthermore, socio-cultural and technological developments confine this state of understanding to a small window in time, because it cannot account for newly emerging needs and resulting adaptation and appropriation of a given design. Norman, an esteemed contributor to design, HCI, and their related processes, even claims that listening to users too much can be the cause for unsuccessful designs, because they will fail to address the actual problem. He writes: "Look at those detailed scenarios and personas: honestly, now, did they really inform your design? Did knowing that the persona is that of a 37 year old, single mother, studying for the MBA at night, really help lay out the control panel or determine the screen layout and, more importantly, to design the appropriate action sequence?" (Norman 2005, p. 18).

4.3.2 Ideation

There is a wide range of ideation methods that are usually conducted by the design team after some domain knowledge has been gained, i.e. after the inspiration phase. Often another round of ideation is performed for more detailed aspects of a design solution – a testimonial to the iterative design process. A common activity is some form of *brainstorming*, where the group tries to produce as many ideas as possible in a set amount of time. For this to work effectively, an atmosphere that is friendly yet intense needs to be created, where wild ideas are encouraged, which means the team is to adhere to a set of rules: "Silly stuff" should be allowed (Kelley 2001), and no ideas be judged at this time. Instead, participants should build on each other's statements to disband their own thinking

patterns and think into new directions (IDEO.org 2015). It is also a useful practice to engage in some warmup exercises like word games to relax the group beforehand (Preece et al. 2015).

These sessions are sometimes aided with artifacts or further specifications of how to collect these ideas. One way to undertake brainstorming is *3-5-6 brainwriting*, where everyone writes down their own ideas silently on separate pieces of paper in a short time and then switches notes to build on each others' ideas. When some ideas already exist, they can be more diverged on by asking *5 why questions* in a row, or use *abstraction laddering* (scaling the issue up or down), to identify new aspects (LUMA Institute 2012). Key to all of these exercises is having well thought-out questions to pursue that are interesting – ideally even unconventional – and spark the group's creativity.

Rapid Prototyping is a quick and efficient way of creating a tangible version of the design to convey an idea, rather than be exact in all details, and test in order to get feedback from users (IDEO.org 2015). Sometimes referred to as *rough and ready prototyping*, the low-fidelity means the artifacts – which can come in the form of sketches, paper prototypes, storyboards, rough code prototypes, or click dummies – are fast to create and therefore do not require a lot of resources or time (LUMA Institute 2012). This method can be useful at all stages of the design process: Preece et al. (2015) suggest that it can function as a brainstorming method where the prototype is manipulated live to spark new ideas, to try out and evaluate a holistic idea, or to figure out specific aspects of a design. It is also useful to note here that these representations – which ever state they may be in – can be an essential communication tool to have when talking to people about the design, whether they may be fellow group members, outsiders, or stakeholders.

4.3.3 Implementation

Through the gathered input from different sources, be it user interviews, stakeholder provisions, literature reviews and so forth, some demands to the system to be (re)designed will have already transpired hitherto. These, together with additional results from evaluative methods described further below, will need to be collected and formalized, in the form of *design requirements*. Cooper et al. (2014) argue that these are not features and not specifications, but determine the need of a product. In other words, the *what* should be construed before the *how*.

The *card sort* method, as described by Baxter et al. (2015), involves noting the various components that are planned to be in the system on index cards each, and are then grouped and arranged by workshop participants. This helps the designers to understand how others – preferably members of the target group – think these elements should be placed in relation to each other, meaning it ensures that the product serves the mental model of the users (Baxter et al. 2015). This can also be conducted as part of a participatory design workshop where expert users and potential users, together with the design team, work to

create a solution. To further flesh out the relationships between components, scenarios can be employed.

Scenarios, as they are employed for Goal-Directed Design according to Cooper, allow the designer to "define a product's behavior" (Cooper et al. 2014) and the function hierarchy according to the user research and can be improved and built upon iteratively. Herein, Cooper distinguishes between three kinds: *context scenarios* are to focus on the persona's circumstances and their motivations and life goals; *key path scenarios* expound the paths the users will typically go through, beginning with the most common key paths and then becoming increasingly detailed; Lastly, *validation scenarios* can be used as an evaluative method and are often phrased like what-if questions, trying to find faults in the current version (Cooper et al. 2014). Here, the concept of key path scenarios is arguably the most important when it comes to developing the core architecture of a piece of software, and in order to represent these paths *flow charts* are a common choice. Also known as *schematic diagrams*, these show the different components of a product and how they connect. This could for example be all the views of a smartphone application with arrows indicating where each button tap would lead the user, ensuring a sensible navigation structure. It can be considered some form of a paper prototype and can be highly beneficial to figure out all the functional details and consequently build a sturdy skeleton that accommodates all previously identified key path scenarios and required use cases, and eliminates dead-ends (LUMA Institute 2012).

As a first visual implementation of a product, *moodboards* can be used to manifest hitherto accumulated ideas about the design. A moodboard is a collection of images, sketches, textures, or any kind of visual representation, that as a whole present the atmosphere the planned result should convey (Cassidy 2011). This technique is an effective way to gather inspiration and use these to reach a common understanding in the design team and with the client. A more refined sort of mood board – and a possible succeeding activity to focus what has been agreed on in the team – is the *style tile*, proposed by Samantha Warren (n.d.). Here specific color swatches, possible font combinations and text treatment, image styles, and other elements are combined to provide a focused reference throughout the subsequent design process.

An immediate relative to rapid prototyping is *prototyping* – no rapid here – which denotes a high-fidelity product and which usually happens at a later stage, when the finer points are to be visualized and implemented. Preece et al. (2015) explain that a hi-fi prototype strives to be complete in its components, functionality, and interactivity, and should have the near-final look and feel. Since this is time and cost intensive, this should not be used as the first testing tool, but when as much as possible has been ascertained.

4.3.4 Evaluation

Evaluation is an important aspect of any design process and should take place throughout, whether it is to take stock of research insights, converge ideation results, or to assess the functionality of prototypes.

Concept mapping can be an effective method to organize complex topics. At first glance a concept map may look very similar to a *mind map*, because they share the characteristic of an interconnected web of keywords. While a mind map is just that, during concept mapping the second stage is to label the links (the connecting lines) with describing words, as defined by Luma Institute (2012). This adds another layer of information of knowledge that remains often implicit otherwise, and thinking in this second layer forces the participants to spell out these connections, which facilitates the common understanding in the group (LUMA Institute 2012). This can be a useful method at the inspiration stage to gain an overview over all the hitherto collected results and connect the various forms of knowledge to each other.

Luma Institute (2012) presents *Affinity clustering*, often also called *KJ method*, as involving writing all findings of a given topic on a post-it each and having the group members one after another arrange the items according to similarities. After discussing and rearranging the sticky notes accordingly, the thus formed clusters need to be named appropriately. This establishes common themes that can then inform the next steps, such as areas for further research. Affinity clustering also works well after brainstorming sessions to map out and structure the results (LUMA Institute 2012).

The *heuristic review* method can be regarded as an efficient checklist-style approach to evaluate systems against, and includes the following ten principles that should ideally be adhered to: "1) Match mental model; 2) Minimize perceived complexity; 3) Use consistent form, words, and actions; 4) Provide a sense of place; 5) Account for user and environmental constraints; 6) Anticipate needs; 7) Use clear and concise language; 8) Give feedback about actions and status; 9) Prevent errors and provide graceful recovery; 10) Strive for appropriate and minimal aesthetics" (LUMA Institute 2012, p. 22).
These axioms can be applied to assess the completeness and consistency of either existing products in the beginning of a project, or as an interim step in the development process as a basis for the next iteration of the prototype.

As soon as some form of prototype exists, even as lo-fi as in paper form, it can and should be tested, as advocated by several authors, for example Cooper et al. (2014) and Preece et al. (2015). Again, there are a variety of methods, that each have their advantages and disadvantages, which relate mostly to effort versus thoroughness of yielded results. While some valuable feedback can be received from more informal feedback sessions, where the designer presents sketches or wireframes to a person not involved in the design process, it is usually limited and not detailed. The upside is that this sort of evaluation does not

involve much planning or resources. A full-scale *usability test*, on the other hand, needs more meticulous preparation, appropriate participants, and the sessions are usually conducted in a more formal manner (Cooper et al. 2014). These tests are an effective means to find out some of the following things:

> "Organization—Is information grouped into meaningful categories? Are items located in the places customers might look for them? First-time use and discoverability—Are common items easy for new users to find? Are instructions clear? Are instructions necessary? Effectiveness—Can customers efficiently complete specific tasks? Are they making missteps? Where? How often?" (Cooper et al. 2014, p. 140)

Another form of verification, that is often combined with a usability test, is the so-called *think-aloud testing*. Baxter et al. (2015) describe it as prompting the participant to narrate her actions and thoughts as she explores and interacts with the presented product. This can help the design team to identify weak points in the current prototype, by easily understanding how the user perceives the design and learning what seems logical or difficult for the users.

Rather than testing a whole system, there is also a technique to evaluate a very specific part of the interface, called *AB testing*. As specified by Nielson (2012), usually two designs of a feature, for example two variations of a call-to-action, are built into a website and some users are exposed to option A, while others are redirected to option B. The more effective design is consequently established through conversion, bounce rates, or how long it took for the user to select the option in question. While being relatively low in cost and risk, the tested features usually have not a large scale impact on the system (Nielson 2012). This is a common practice to improve small features of an existing website or application, but can of course also be employed to determine whether a solution is effective or not during the development of a service.

To truly test a system with its complex technical behavior is often a difficult feat, as the time and expertise required to built a fully functioning prototype *just* for testing is often not viable. To circumvent the need for a testing environment that is essentially a fully fledged product, or part thereof, the so-called *Wizard of Oz* method has been developed and is readily used in the HCI community (Dow et al. 2005). Essentially, the idea is to make test participants perceive the system to respond to their actions, where in reality a member of the design team – the wizard – creates the appropriate responses. This can for example be achieved through an additional computer connected to the machine with the testing interface (Green et al. 1985).

5
Process

This chapter describes the undertakings in chronological order that were conducted to create both the pattern collection and the Honeycomb ATS prototype. It is noteworthy at this point to emphasize the approach to our working process: Developing a pattern collection and a system prototype in parallel – where findings of one artifact inform the progress on the other and vice-versa. This cross-fertilization between the conceptual product and the practical implementation continued throughout the thesis project. So even though the Process chapter is split up according to the five iteration cycles with an additional cycle that summarizes the final pattern establishment, the activities overlapped and informed each other.

Drawing from our previous domain knowledge, the initial plan had been to observe the general HCD process, and work in several iterative cycles. We were aware of almost all of the methods described above, with the exception of style tiles, but the methodology research provided the depth of knowledge needed for selecting and arranging the process purposefully. In terms of time, the general prescription had been to spend the first four weeks, out of the total twenty working weeks available, on researching and writing a planning report. Similarly, four weeks at the end were recommended to reserve for writing the final report. The remaining twelve weeks were to be used for the process, implementation, and further research if needed. More detailed planning was to be undertaken when the scope of the project was more defined and the pre-study was completed. The resulting plan is presented in section 5.1.5 Planning. Throughout the project we also ensured to accommodate the group of bachelor thesis students and provided them with specifications and results of wireframe and designed screens, so that they could program the backend and frontend accordingly. All of the activities part of the process were recorded and described in a process diary, to help us keep track of interim findings and the decisions that we took at each stage.

5.1 Pre-Study and Planning

This section describes the preliminary research activities, including the study of secondary literature sources, and carrying out stakeholder and competitor research in the form of interviews, contextual inquiry, and benchmark analysis. This first iteration concludes with the detailed plan that we adhered to for the project, including finding the patterns, designing the prototype, and documenting everything in this report.

5.1.1 Literature Study

There were many topical aspects we needed to familiarize ourselves with first to be able to stake off the scope of this project sensibly. For this purpose, we first listed an extensive set of topics and surveyed related material superficially and subsequently centered in on the aspects that transpired as the most relevant. The topics on this list included literature about applicant tracking systems, the practice and profession of recruiting, web applications, desktop software, interface design, data visualization, the theory of patterns and pattern languages, methodologies and processes, and frameworks. Also, we needed to learn more about the very specific situation of our primary and secondary stakeholders. The goal of this was to determine exactly what we could and would deliver according to the resources and time available, and to break down which building blocks were needed to further our cause. A significant part of the research was centered around the different definitions and structure of pattern languages (see section 2.3). It transpired that miscellaneous views and purposes exist on the matter, and our conclusions on the subject are presented in the theory section. At the beginning of the project we had also not yet decided on whether to deliver patterns or guidelines, and therefore we explored both concepts as part of our literature study (see section 2.2.1). After learning about the merits of both, we decided to go for patterns, where the conciseness and the emphasis on the relations between patterns seemed more suitable for creating a purposeful result. The other influential field of research was CSCW, which brought our attention to the issue of awareness and led to the final formulation of the research question.

Another focus area was to immerse ourselves more into the world of hiring workforce. We were planning on tapping into the resource directly available to us and learning from our stakeholders as much as possible through interviews, contextual inquiry, and later on ideational and evaluative methods. In preparation for these endeavors, we searched for secondary literature on topics such as the history of recruiting and e-recruiting, current industry articles on both hiring and being hired (so both the employer side and job seeker side), and the state of recruiting tools such as ATSs. As the number of scholarly articles on these topics was not as plentiful as we would have liked, we diversified on the type of source, and proceeded to extract some of the expertise from articles written for advisory or sales purposes. However, the knowledge found therein still primed us for the sort of problems that are common within this line of work, and led to the formulation of some of the subsequent interview questions. Yet another large share of the literature study was spent on revisiting previously encountered material on design methods. As we already had a

general idea of how to go about the design process, it was a matter of selecting the appropriate methods in a sequence that aligned to the goals set for the project. This included, for example, finding arguments for and against using given methods in papers with both qualitative and quantitative evidence to support their claims, and thus elect and plan for the most suitable techniques.

5.1.2 Interviews

To get an initial understanding of the company, goals, job descriptions, workflow, and what it means to work with an ATS we started our process by conducting semi-structured interviews. Our hope was more than to understand the current situation, also to detect critical areas in need of improvement.

The interviews began with general and open questions to establish a connection with the interviewee and to make them feel at ease. As the interview went on, we gradually progressed to more specific questions, and concluded with a demonstration of the current tool, where the interviewees demonstrated different aspects they had been talking about previously. Overall, these interviews were fruitful, and every subsequent session yielded more detailed results than the preceding one, as we were able to dig deeper based on the thus far gathered insights. Also, we purposefully refined and added to the prepared script as our understanding expanded.

These interviews presented us with a broad understanding of many aspects: from the recruiters' background, the business goals of Software Skills, to their workflow, and their expectations for our thesis results. A difficulty at this stage was to obtain detailed knowledge of how they work and the problems they experience with the system. Some specific results that accompanied our reflections along the subsequent process were created here. Among these were, for example, realizing the importance of the job interview and the human factor of personal impressions. How can this be resembled in an e-recruiting system? Also, the issue of collecting and storing these subjective pieces of information in places where they could be found by colleagues – as a response to this specific case the pattern of *shared annotations* (see section 6.3.6) was conceived. Another aspect that several of the interviewees discussed in detail was the search and filter functionality that was unsatisfactory in the current system, which led us to explore corresponding solutions in detail at later stages.

When analysing the overall findings, we realized how much the recruiters work as a team, how much they discuss issues that appear on a daily basis, and that trying to find solutions together connotes a substantial share of the work. At the same time, we noted down quotes such as "I use some lists [for my candidates] that the others don't" or "I add extra notes to myself", which meant that individual flexibility and personal settings not affecting the entire system are also important. These insights in turn led to the focus on

building a system that supports these very issues, and the idea to look for patterns that satisfy this set of requirements and also led to the final form of the research question.

5.1.3 Contextual Inquiry

To get an insight into how the recruiters at Software Skills work with the system on a daily basis, we undertook the contextual inquiry method. The purpose was to reveal their actual workflow, common problems, and to understand how and with what tools they collaborate. Through interviews for example, it is more likely to get a description of the optimal workflow rather than the way it actually occurs. Since the current workflow and tools have grown organically with the growth of the company, we believed that the contextual inquiry could yield a lot of useful insights on what is important and learn about the reasons why. The method was conducted straight after the interviews to take advantage of the flow of discussion.

It transpired that the recruiters communicate mostly outside of their current system, with the help of third party email programs, whiteboards, daily meetings, paper notes et cetera. Meetings, where recruiters discuss their work progress and agree on daily and weekly tasks, are held every morning. On top of this, whiteboards are used to communicate the status of clients, distribution of work and responsibilities, and daily tasks for the team as a complement to the Trello solution. To display the status of various things, different markers are used which are sporadically and manually updated. An effect of this way of working is that their workflow is very modular and adaptive. An example of this was the use of Trello which allowed for the recruiters to set up new boards and lists suitable to their current process. Another consequence of the freedom is that it opens up for many possible mistakes and misunderstandings among the recruiters. For instance, if a candidate card is to be moved to the rejected list, it automatically sends the applicant a rejection email. These ejectorseat levers are too easy to hit by mistake in the current system, and unfortunately candidates have indeed been rejected by mistake on several occasions.

These findings made us realize that a lot of this manual labor could be automated in a new ATS solution. For example, we decided to automate parts of the setup process, where every process has the most common hiring stages added, possible tags regarding a candidate's expertise are also decided in the here. These are later requested from the candidates, meaning that the recruiters do not have to add tags to every single candidate manually. When a candidate is rejected a general email template is presented for the recruiter to use. The template has certain highlighted phrases that can be modified to make the letter more personal. Furthermore, their routines and use of surrounding resources indicated that the existing system lacks easy means to communicate actions and intentions, and to keep data updated and complete. The idea of an integrated chat function was born as a result. Their inability to use Google Drive to get hold of statistics about their process due to the complexity of finding the right information gave us the idea to provide the information in a more intuitive and enjoyable way. When this finding was discussed with Henrik, a

dashboard solution which provides general statistics of the current process was conceived. Further insights, such as how recruiters rate candidates and the problem of incurring individually biased decisions emerged, and let to the formulation of the requirements which were used later on.

5.1.4 Benchmark Analysis of ATS

Surveying and evaluating other ATSs proved to be quite a lengthy but also important step into understanding both the scope of our practical project and an outlook on our theoretical deliverables. We created a list of all different kinds of ATSs and ordered them according to size, i.e. capacity to hold candidates and breadth of functionality, in order to determine their respective target audience. Then we selected a few for each size and also according to country of origin, to gain insight into a wide range of distinctive systems. Also, we included both established systems and ones that were newly-released with more current design and specific niche features. For each ATS, we explored the interface and capabilities, and recorded the most noteworthy features through screenshots accompanied by written annotations. We analyzed eleven systems in detail (CATS, Greenhouse, Jazz.co, Jobvite, Jobylon, RecruiterBox, TalentBin, Teamtailor, Uptrail, Upwork, Workable) (refer to 3.4 Competitor Applicant Tracking Systems for a description of the results), two of which were not ATSs per se: Upwork is more of a freelancer finding platform, but it has many similar functions to an ATS. TalentBin is a metasearch engine to find passive candidates in social media platforms and forums, but the search and filter options and candidate handling and so forth were relevant nonetheless. The results are described in section 3.4 competitor applicant tracking systems.

As a last subactivity, the functionalities were collected and positioned against the ATSs in a matrix, to summarize and quantify the results. This also highlighted the most common components of these systems, and thereby provided an indication what the central functionalities of all ATSs are, but also the areas where USPs could be found.

During this phase, we analyzed the behavior and differences of various activity logs. We mainly looked into what information was shown, what possible interactions were provided, but also how the log changed over time and how it depended on the user's navigational movements on the web application. The data gathered here was stored to be consulted later when it was time to determine the exact makeup of the activity log to be included in our design. More than analysing the logs, we looked for inspiration of useful functionality and solutions by investigating how other systems support awareness, collaboration, and efficient workflows. Furthermore, we identified if and how other systems supported direct user communication, collective rating, possibilities to schedule appointments, automatic emailing, system settings for teams and users, among others.

5.1.5 Planning

Having determined the research question, established fundamental knowledge in the theoretical and practical areas that pertain to ATSs, web applications, and design patterns, and having expounded a range of methodological approaches, we set up a detailed project plan. The overall process model was to be iterative, as stipulated by the HCD ISO standard (ISO 9241-210, 2010) but components were to be borrowed from other schools of thought that have been introduced in the methodology section above. Overall, five iterations were planned that each cycle through the stages of *inspiration, ideation, implementation*, and *evaluation*. The first two iterations placed more emphasis on inspiration and ideation, during which the bulk part of the research and user studies were to be conducted. Likewise, the later run-throughs were more focused on implementation activities. The design process was planned to be concluded with some smaller iterations of testing and polishing the design in its final stages as needed. Our theoretical results – constructing the pattern language – were worked on throughout and be evaluated by means of the practical implementation.

"Because personas resemble real people, they're easier to relate to than feature lists and flowcharts," (Cooper et al. 2014, p. 64) is Cooper's opinion on mentioned methods. After researching the pros and cons of personas however, we felt that for our case, where not an entire brand of a product is conceived from the beginning, this sort of *relatability* is not the core concern. Also, the ATS is a very framed system that is intended for a very specific context and very specific purpose. It is not equatable with a novel lifestyle product that helps its consumers rectify their personal identity issues. So even though we used many of Goal-Directed Design's recommendations for *user research, requirements, scenarios*, and details for the design implementation, we planned to also rely on other activities, such as *affinity clustering* and *style tiles* to guide us through the design process. Overall, our approach was to be more cyclic – as is common for HCD processes – and therefore not as linear as the general GDD process. In terms of inspiration methods, we foresaw that a lot was rooted in the determined requirements, and that the patterns and the design would have to answer those *requirements* specifically. Therefore, the emphasis would not be so much on diverging with ideas, but more on finding specific solutions to the requirements by following a stringent research regimen and focusing on the converging points of those findings. For specific issues, for instance ideating on collaborative issues and interface elements, we determined that the ideation methods *mind mapping* and *rapid prototyping* respectively were more result-oriented, rather than brainstorming techniques such as extreme characters or what if scenarios.

For scheduling these activities in an appropriate time frame, especially in light of allotting ample time towards the end for *testing* and evaluation, and for implementing those results, we established the plan shown in figure 5.1.

Week	Description	
1-4	Intitial planning and Research	Literature Study, Interviews, Contextual Inquiry, Benchmark Analysis, Planning Report
5-6	Second iteration cycle	Affinity Clustering, Establish Requirements
7-8	Third iteraction cycle	Benchmark Analysis, Brainstorming, Card Sort, Concept Mapping
9-11	Fourth Iteration cycle	Paper Prototyping, Evaluation, Wireframes, Usability Tests, Requirements Check
11-14	Fifth iteration cycle	Collect Awareness Patterns, Style Tiles, Design Implementation, Validation Scenarios
15-16	Final iteration and wrap up	Improve Design Implementation, Improve Patterns
17-20	Documentation	Report Writing, Animated Prototype if time

Figure 5.1: Project weeks and respective planned activities.

5.2 Establishing Requirements

During this second iteration cycle, we drew from the findings of the previous stage and specified the hitherto collected results from the preceding literature study, interviews and inquiries, and the benchmark analysis. First we diverged with an ideation session, sorted and evaluated the results directly after, which finally resulted in creating the first draft of the requirements.

5.2.1 Affinity Clustering

We undertook the affinity clustering in order to collect and organize the requirements of the system that was to be built as one of the results of this project. Inspired by the 3-5-6 brainwriting method, we began by writing every aspect that came to mind indiscriminately on a separate post it. As the flow of ideas started to dwindle we started discussing with each other and together generated yet more points to add. Here are some random examples of items that were put forward: “Ask a fellow recruiter a question”, “Add a new user to the system”, “Make it easy to integrate information from other places”, “Show progress”, “Promote shared understanding of a candidate”, or “Communicate clearly what actions are possible at each point”. Some points were very specific requirements that corresponded to exactly one action needed in a specific place in the system, while others were more overall pointers to be kept in mind throughout the design process. As shown in figure 5.2, we sorted the notes on the whiteboard by relation to another in the first iteration and three main topic clusters emerged already at this stage: General UX concerns, general awareness considerations, and thirdly, specific recruiting system requirements. In the latter case, it surfaced that it was possible to sort the items almost chronologically according to a workflow of overview screens, setup screens, followed by more detailed parts of the system. This structure was a constructive prerequisite for the subsequent activity of formalizing the requirements. With this activity we were able to stipulate the overall workflow that the ATS needed to support, and where information and functionalities and settings should be available in a given screen. Furthermore, the actions that corresponded to each element were also identified.

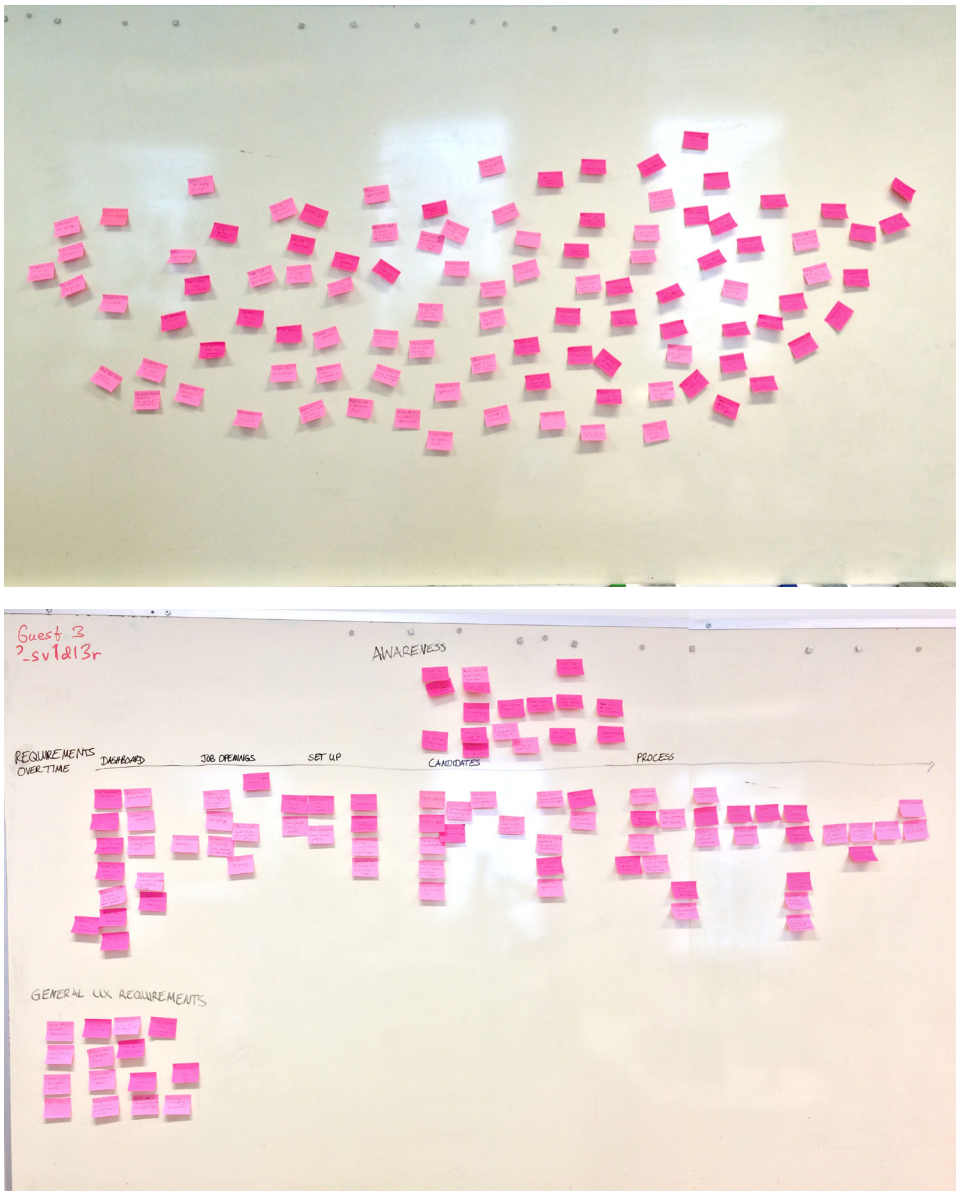


Figure 5.2: Affinity Clustering stage 1: Unordered post-its, and stage 2: Categorized ideas.

5.2.2 Establishment of Requirements

With the results from the previous method at hand, an exhaustive list of all possible requirements was created. For this purpose, we used the collaborative web application Coggle, previously introduced in chapter 3.5 Collaborative Web Applications. The main branches were based on the groupings of the affinity clustering, and stipulated as “Dashboard”, “All Candidates”, “Job Openings”, “Candidate Single View”, “System Users”, and “Tasks”. Then all respectively connected functionalities and parameters were added successively in a drill-down fashion. When all items from the affinity clustering that were a specific feature were exhausted, we revisited our benchmark analysis results document to ensure all previous findings were included. We also provided the list, with prioritized screens and functionalities marked, to the bachelor thesis students, so that they could plan and work accordingly.

The method sparked a lot of discussion. We identified a number of tricky edge cases and challenges that the system needed to support, related to, for example, awareness, permissions, duplicate artifacts in multiple places, and the system-wide implications for some features. Furthermore, the stipulation of building for flat organizational structure warranted that nothing is ever deleted directly, and instead finished processes and removed candidates are first archived to be able to reactivate elements if needed as an error reversal method. Also, the need for teams was determined, which enables sectioning off certain information and places in the system. The various teams can access different processes, which discourages openness and awareness to some extent. However, this *non-awareness* that we identified, promotes efficiency and prevents users to receive irrelevant information. It is possible for everyone to join teams, making the pattern still a relatively democratic feature.

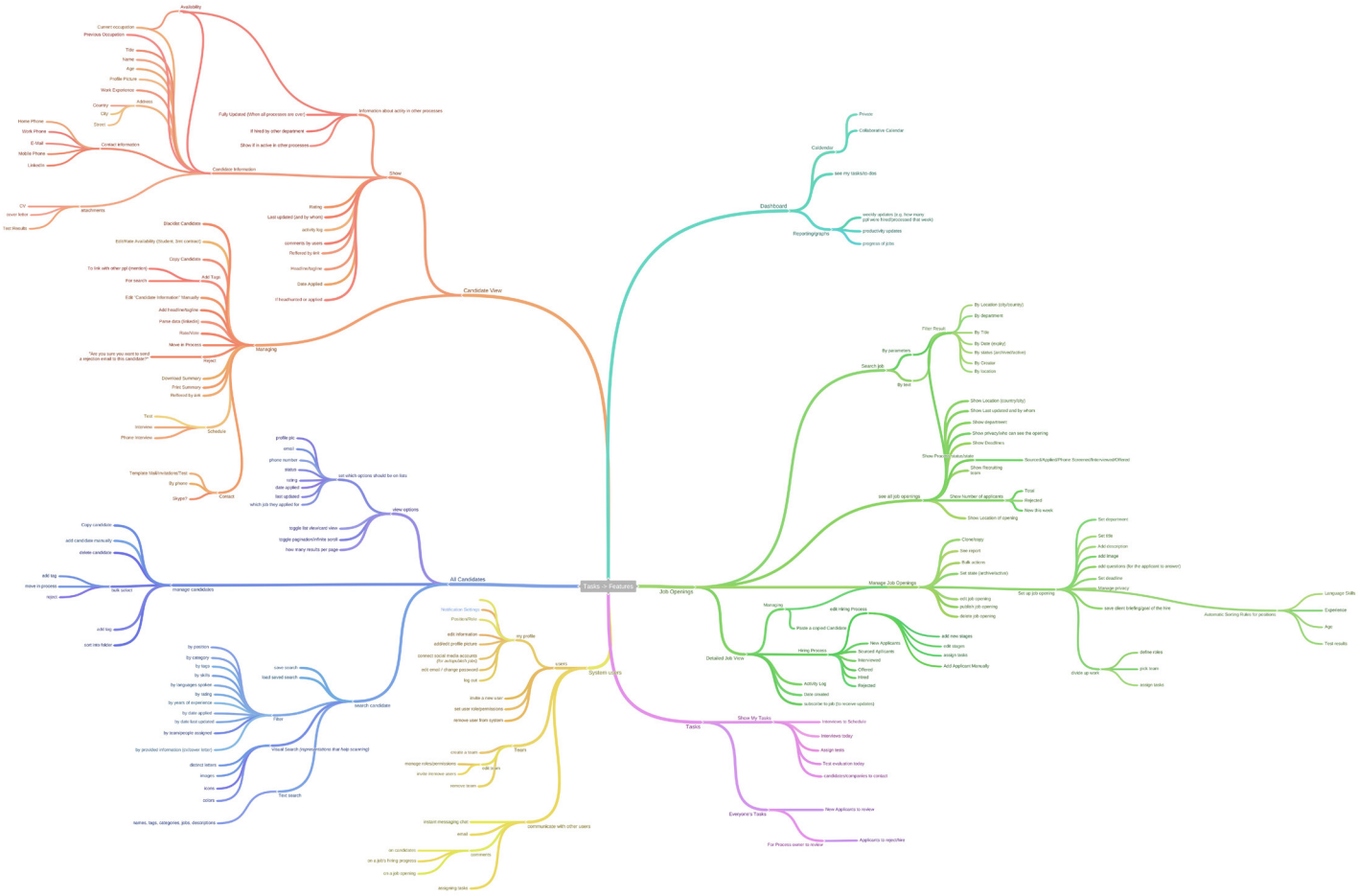


Figure 5.3: Requirements mind map (see Appendix 2 for higher resolution of this image).

The tool Coggle lend itself very well to the activity, it helped us think of all the details that we previously missed and store them in one place. Also, it was possible to dynamically add and resort the items easily and visually. Therefore, we kept revisiting the online document as we found more points to include. In subsequent stages, this document provided a thorough reference to check for consistency and completeness.

5.3 Finding Awareness and Collaborative Patterns

The third iteration cycle of our process focused on getting started with the patterns that we had set out to determine. At the beginning of the iteration, we were quite unsure of how to find a starting point that would lead on a productive path. Therefore, we consulted the resources on patterns we had collected during the literature study, and decided to use the questions presented in figure 2.3 the awareness table as a basis for the initial ideation method. This meant that we decided to deviate from the project plan a little by switching around the order of concept mapping and the second round of benchmarking. Also, while attempting to conduct the concept mapping we realized that the activity and result should more justifiably be called mind map, as the connecting lines did not warrant any further labeling. Subsequently, further inspiration methods were used to gather more insight into how other systems address these issues, and further literature provided additional guidance. The cycle concluded with a document of rudimentary notes on possible patterns, that was revisited and added to at later stages.

5.3.1 Mind Mapping of Awareness Issues

To launch further into the investigation of awareness aspects and how a system can support collaborative work, we mind mapped all branches around the keyword *awareness*, based on the table by Gross and Koch (2006). These questions were: What are the goals of [different people and roles in the system]? What can others do? Who is responsible? Who is in this process? What are others doing? How are we [as a team] doing? In answering these, we attempted to find ways to satisfy these concerns in an ATS. For example, for the latter the suggestions included features that could be shown on the dashboard, such as productivity updates, highlighting upcoming deadlines or tasks to be completed, show hiring progress of jobs, weekly evaluation for the team, and personal weekly reports. Similar productivity indicators could be shown in different parts of the system, specifically for that part or the affected team, or also adjacent sectors and teams. These indicators could both be applied for either real-time feedback of who is doing what and where, or a summary aggregated over a certain time span. Also, some other secondary issues surfaced, such as showing *intention* of a user to others currently in the same location.

As more and more possibilities came to mind, we realized that adding all of these features into a system would lead to a state of *hyperawareness*, as we called it, and while it may be effective in communicating team activity and all related data, it may well be detrimental to productivity. What effect does seeing all of this information have on the individual user? Some possible consequences would perhaps include that users would feel under surveillance, or that the constant stream of real-time data about other people's actions would distract, overwhelm, and pressure users. Therefore, we created a separate mind map, centered around *non-awareness*. Here, we listed the ways awareness could intentionally be discouraged, and allowing users to focus on their own work instead For example, instead of showing who was currently working on an item, only indicating that *someone* was

working there. Or even one step more extreme into non-awareness, would be the system only denying access to the item in question, but not disclosing that another user was there.

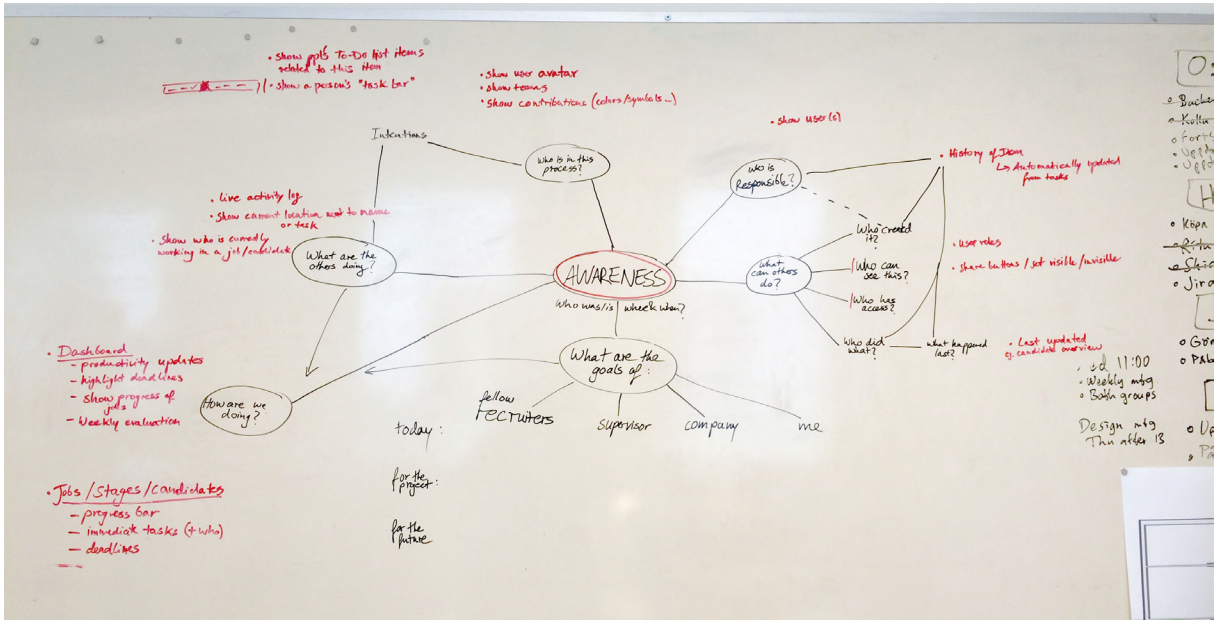


Figure 5.4: Mind mapping questions related to awareness within an application.

We had expected to get more specific features out of this mind map exercise, instead we found more overarching, deeper issues connected to how information provided will affect users in a system. It materialized that we needed to find this balance in our system, of showing enough awareness to collaborate effectively, but not too much to distract users from their work.

5.3.2 Benchmark Analysis of Collaborative Apps

To supplement the results of the preceding awareness mind map, we decide to deviate from the process plan and take some time for a closer look at other collaborative web applications and software to learn more about how the industry addresses these issues. Therefore – similarly to surveying the state of the market in terms of competitor ATS – we also analyzed the landscape of collaborative web applications. Many such web services exist, catering to a myriad of different purposes. Therefore, a large collection of different examples of web applications and tools that allow or even facilitate collaboration among several people was created. These were then categorized into groups according to the framework by Grudin and Poltrock (n.d.) into *asynchronous information sharing* and *asynchronous coordination* and find the most prominent examples for these categories: *Document Repositories, Wikis, Team Workspaces, Workflow Management, Project Management*, and *Calendar Scheduling*. Then the meaning of each category was briefly outlined, and then the two to three most noteworthy representatives were introduced and the features that help collaboration were explained. On top of that, we actively looked for present functions that mapped to our awareness questions. As a result, we accumulated a substantial collection

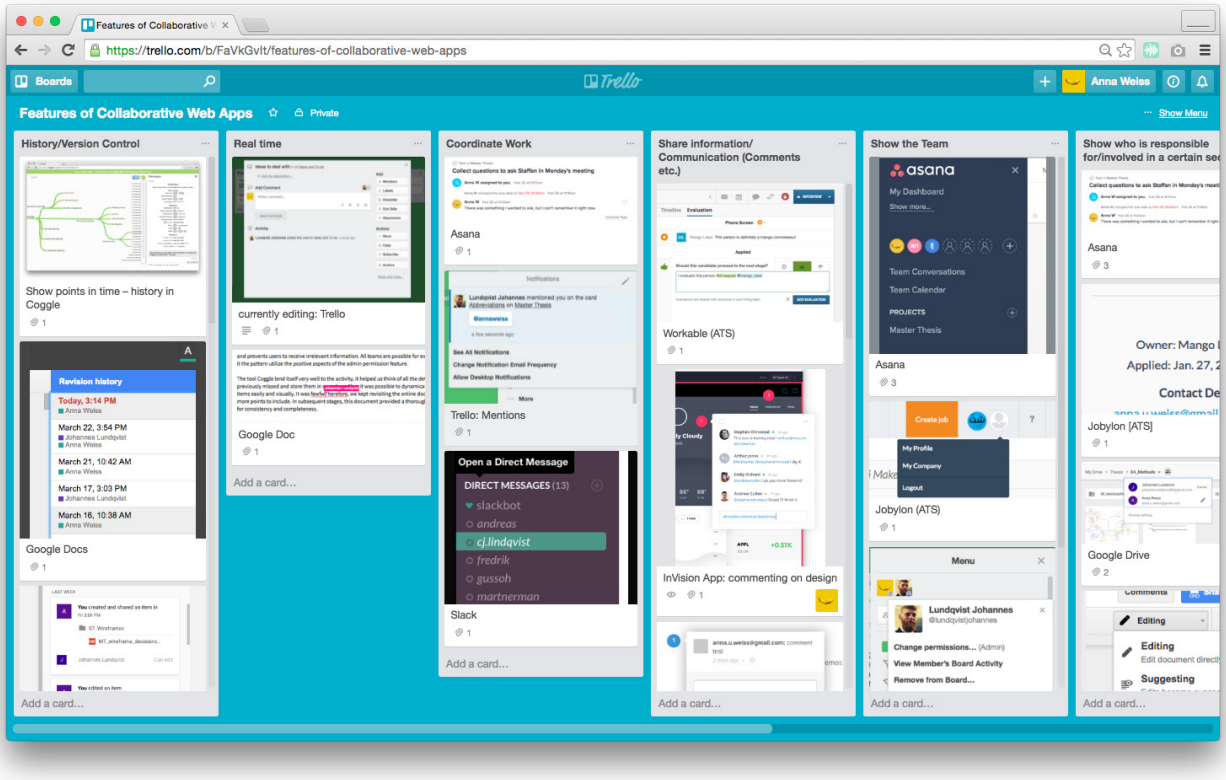


Figure 5.5 Screenshot of collected details from other collaborative web applications.

of different solutions for certain patterns. For example, history logs and version control, showing the involved team, coordinating work and so forth, were documented and cataloged in a separate Trello board, as a reference and inspiration for the pattern collection to be produced (see in figure 5.5).

5.3.3 Enter Stage: The "Patterns for Computer-Mediated Interaction" Book

During the literature review at the beginning of the project, we came upon a book titled "Patterns for Computer-Mediated Interaction" published in 2007. Procuring the title took a number of weeks and it was shortly before embarking on the implementation phase that we held it in our hands for the first time – the reason it is not introduced in the theory section where it would have otherwise naturally received its own dedicated sub-section. The tome penned by Till Schümmer and Stephan Lukosch proved to be influential on the subsequent course of the result finding process. The authors address three different potential audiences: the software developers creating groupware applications, the users of such systems, and finally researchers in the field (2007, p.2). This motive is evident in some places of the pattern explanations, which are clearly aimed at the technical implementation and the aspects one must be mindful of from a programming architecture perspective. While this is entirely valid and considering computer science is the domainal departure point for both contributors, the pieces of advice could be optimized for interaction designers and their demands to a pattern library. Seeing as IXD and UX were not as much a thing then as they are now, and the landscape of web applications has

witnessed a developmental leap, it makes our contribution of patterns all the more valid. Different aspects of considerations in regard to groupware are provided to the reader: Firstly, the advocation of making the user group explicit and their respective goals and *end-user requirements*. Secondly, the *interaction between people* should play a central role in the design of the CMI software at hand. And finally, technology should not be at the center, but the humans using it. This philosophy is strengthened by explaining how social interaction impacts "issues like trust and privacy" (Schümmer and Lukosch 2007, p. 4) – concerns we had already identified in the previous research and interview stage.

Schümmer and Lukosch propound a comprehensive collection of patterns sorted into two main sections. One congregates patterns under the umbrella of *Community Support*, pertaining to building online or intranet-based communities, which includes social media platforms, for instance. The patterns contained here are at a high abstraction level, answering conceptual questions like "How to arrive in the community?", "How to find out what's interesting in the community?", "How to protect users?" (Schümmer and Lukosch 2007, pp. 69, 86, 157), and the authors no doubt have undertaken an extensive amount of research into the realms of human behavior in communities, communal structures, and self-awareness and self-esteem, and their respective reciprocities. The second main section is titled *Group Support*, which musters paradigms concerning the platforms built for productivity among a group of people, rather than for the sake of creating a community itself. Here, patterns for collaborative applications are described and questions like "How to modify shared material together?", "How to organize textual communication?", and "How to become aware of other user's actions?" are thus explored (Schümmer and Lukosch 2007, pp. 191, 268, 315).

With the outcome of the benchmark analysis of collaborative web applications and other previous methods at hand, some patterns described in the book corresponded to aspects we had already identified. We took our already existing list of patterns and evaluated them with the closest relative in the book. When our design was complete and we went through the prototype looking for more patterns, we also used Schümmer and Lukosch as a reference. This is symptomatic of our aforementioned approach of developing a pattern collection and a system prototype in parallel, but the exact list of preliminary patterns and their respective origins will be described towards the end of the process chapter, under the Pattern Evolvement heading. For now, the stage is handed over to the more practical side of things: Designing the prototype.

5.4 Design of the Prototype Part 1

The fourth iteration was the most time intensive of all the cycles, as we had foreseen in our plan. It was initiated by an ideative *rapid prototyping* session, followed by implementational methods, like *wireframing* and creating a *click-prototype*, and concluding with evaluation in the form of *usability testing*. The last two methods were cycled through several

times, improving the prototype according to the findings of the testing sessions, and subsequently testing those improvements.

5.4.1 Paper Prototyping

In order to be able to discuss and share our opinions effectively, as well as getting started with the first implementational iteration, and translating requirements into features, we created paper prototypes (see figure 5.6). The intention was to recognize which sort of layout could solve and hold all the elements needed per view in the system. We prototyped one solution each individually, and subsequently discussed the pros and cons with each other. In the next step, we transformed these quick sketches into three different solutions in a more refined prototype.

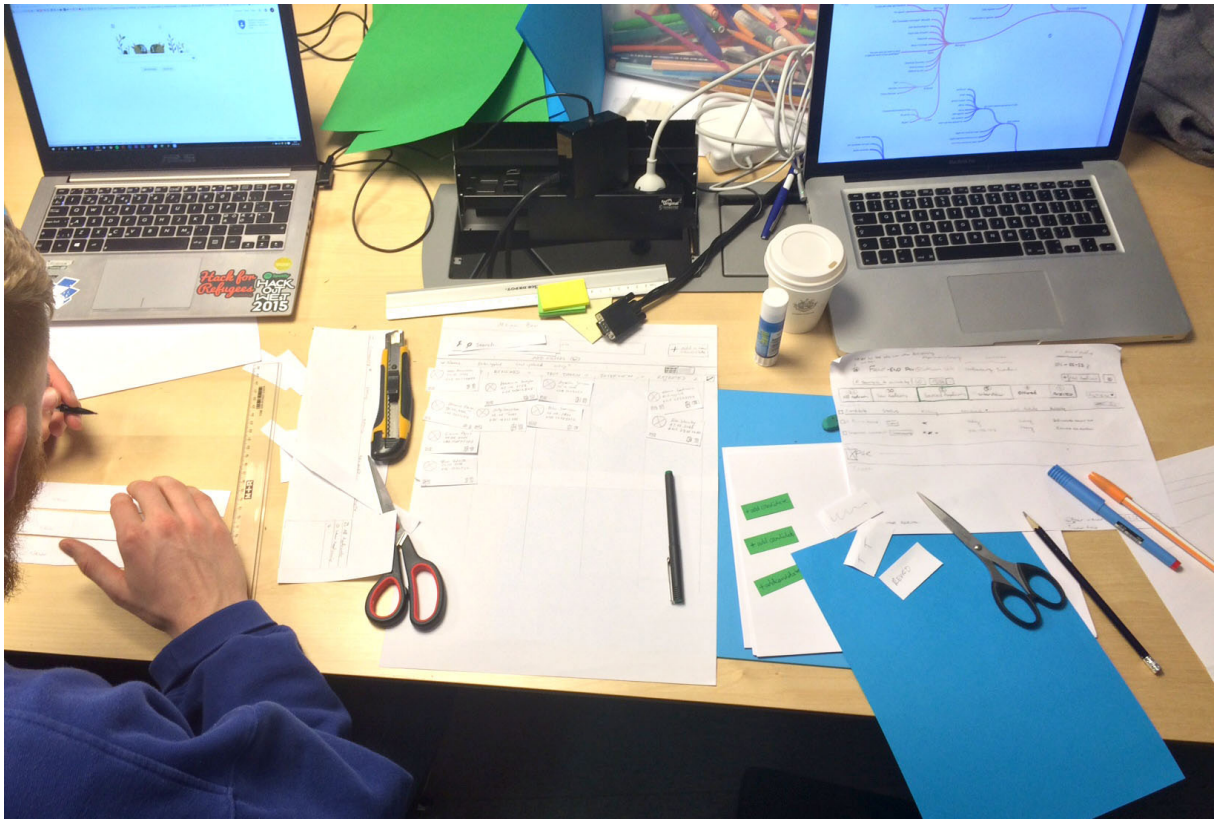


Figure 5.6: Ongoing paper prototyping.

The method was intended to be rapid and quick, and to generate ideas and spark a discussion, however it proved to be rather time consuming in relation to the results it yielded. Instead of resulting in a great and detailed design, the most fruitful takeaway came from our debate on how to divide the content and functionality into separate views for efficient workflow. Also, we realized that a limited set of design possibilities for the candidate overview – one of the central screens in the ATS – exists.

5.4.2 Paper Prototype Evaluation

From the previous method we landed in three different layouts each. The layouts represented the most vital parts of the system; the Candidate list, Jobs, and the detailed view of a Candidate. In order to gather more insights we conducted an informal evaluation method. The idea was to get feedback and more ideas that could strengthen our design and give us new angles to review our designs. One prerequisite was that the participants were unfamiliar with the recruiting process, so we could test whether our design was clear and logical enough for non-expert users to understand. To ensure that different professional backgrounds were represented we picked six people from our circle of friends.
As we showed the participants our designs we asked questions relevant to reveal their impression and understanding of the elements and functions. To keep the evaluation as unbiased as possible, we rotated the order in which we showed the views of the candidate list versions 1, 2, and 3. So for example, participant A was shown version 1, then 2, and then 3; participant B was shown version 2, then 1, then 3; participant C was shown version 3, then 2, then 1, and so on. This ensured that what the participants learned with each new version was randomized.

We got a number of useful feedback points on the candidate list view. It was helpful to hear about people's mental models of how the recruitment process should be shown, from individuals who had not seen a recruitment funnel or hiring process diagram before, because they had no previous biases from other systems. Feedback on the candidate single view was a bit scattered and did not exactly point towards one specific solution. By conducting this simple evaluation, we were guided in the decision of choosing which candidate list view and candidate single view layouts to pursue.

5.4.3 Wireframes

To take advantage of the momentum of findings and inspirations from the previous prototyping steps, we soon went into more detailed prototyping. Wireframing in Adobe Illustrator provided the opportunity to merge our feedback and ideas in a more realistic and concise size, different from the paper prototyping method. Another advantage of switching to digital prototypes was the simpler and faster implementation of small changes, which in turn resulted in a more flexible design process. It also made it much more efficient to actively consider and test alternative solutions to specific interaction and interface elements.

This was the first time we were getting down to the important details of the application. Before, most thoughts had been about the overall structure, and overall views. Now we were beginning with the main elements and figuring out of how to place those in relation to each other, and also looking out for continuity across the various views. When we began adding placeholders for buttons and labeling them, it marked the end of the experimentation and beginning of decision time (see figure 5.7).

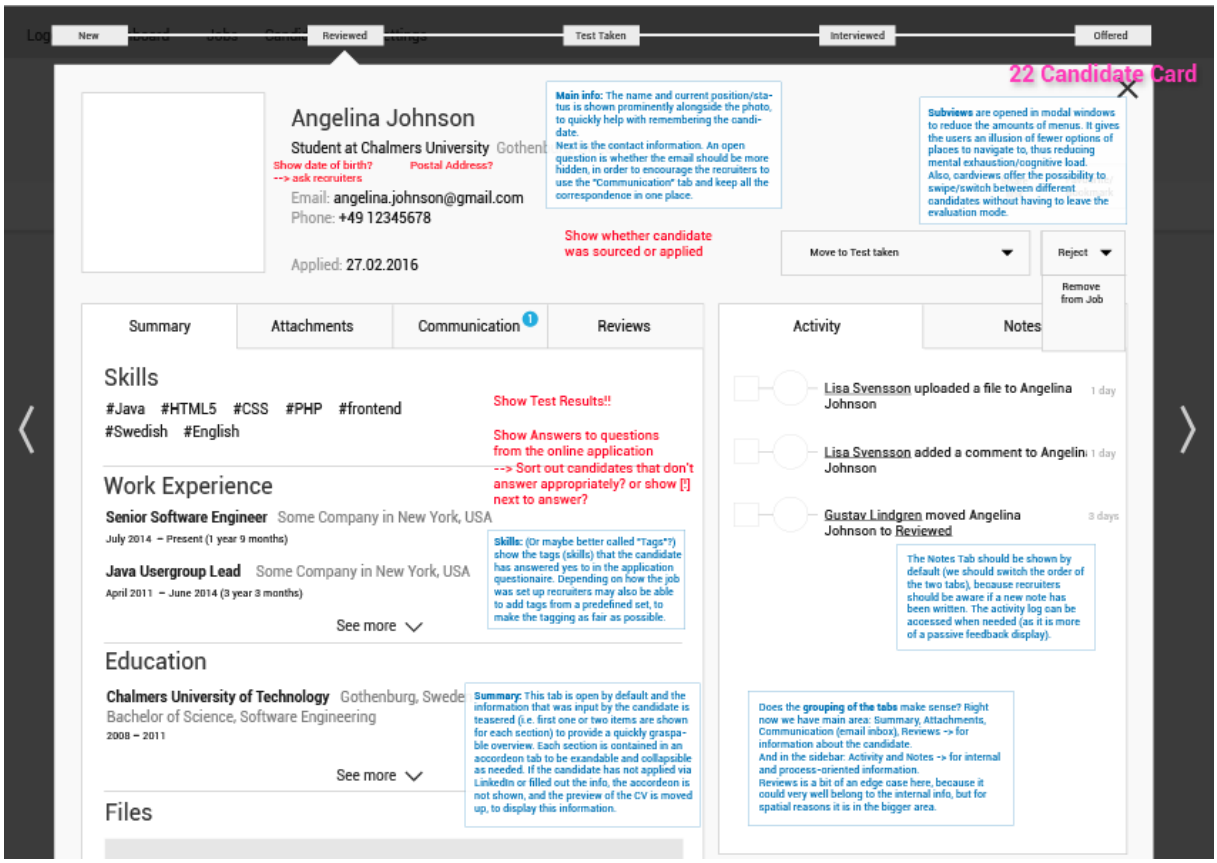


Figure 5.7: Example of a wireframe screen (Candidate Card Detail) with several iterations of comments.

We struggled most with screen "21 – Candidates in a Job". The main functionality here is to show and manage all the candidates that have applied for a given job position. Already in the paper prototyping stage we had difficulty settling on a solution that would satisfy both card view and list view. The combination of the two was based on stakeholder requests and the results of the interviews and contextual inquiry, where we had, among other things, examined the existing Trello solution. Here it was found that the drag and drop functionality, similar to Trello, as well a system that could accommodate larger numbers of applicants, would be worth exploring. The card view is not optimal for larger sets of applicants for several reasons; the amount of screen real-estate cards require, manual labor of moving cards as well as the large quantity of information users have to memorize due to the limited amount of information possible to present in a card. The list view was thus a way to make bulk actions possible but more importantly the possibility to present a lot of information, easily sorted and compared between the candidates.
More than trying to find a logical way of unifying the card view with the list view of each individual stage, we also put focus on determining which sorting and filter methods would make sense for each and how the search results would be presented. Due to requirements regarding system openness that favor flat organizations, we focused more on system behavior rather than solely finding layout and functionality solutions to the main screens. For example, we revisited our team functionality, found in the establishment of requirements and proposed ways to display the teams and connected sub-tasks. For instance,

how one would request to be in a team, how to invite a member, the consequence of rejecting an invitation *et cetera*. We also decided that individual ratings of candidates should be shown upon request and not only the teams' collected opinion, corresponding to the *Differentiated Voting* pattern. Furthermore, the option of communicating via adding notes on a specific candidate card that we had identified during the interview stage developed further at this point. Notes are visible for the whole team, but only the people tagged in the message with the mention feature are notified.

The method worked very well for exploring system behavior such as system settings, system roles and permissions, and confidentiality, because we had a better grasp on what the final system might come to look and operate like at this stage. Also, it was a valuable artifact to provide to the group of bachelor thesis students. With this preliminary visual guide of all the screens and contained functionalities and corresponding interface elements, the programmers could really delve into setting up the code structure correctly.

5.4.4 Axure Click Prototype

The wireframes were designed to visualize all the screens and components to be included in the final prototype. However, this sort of static artifact is not suitable to communicate and test interactive behavior of an application. Therefore, we built an interactive prototype that could be used for the planned usability tests where possible problematic areas and edge cases that occur in actual use could be revealed. Such a prototype also allows to show visual hinting, like hover effects and action feedback, and other supporting behavior that helps users understand the system. The tool employed for this purpose is called Axure RP, a software program where the interface elements are assembled (like in other graphics applications) and augmented with interactive behavior (Axure n.d.). This means that mouseover and clicks can be detected and are assigned state changes in the interface, so that elements can be links to other views, for instance.

There were many issues that needed to be addressed in the concerned main screens, and we cycled through numerous alternatives for these. One was the view switching conundrum. We realized that toggle buttons were not efficient to communicate current state, when they also have to convey interactivity. A few days into working on this problem, we realized that there were actually five different hover states required of the tabs depending on which one was currently active. This was a particularly complex problem to implement correctly and coherently with the programming logics. The view toggle also affected components outside of the table, such as the filter and sorting options, which should adapt accordingly. Even if this was the most time consuming phase of the whole process, it proved to be among the most valuable as it constantly revealed problems and necessary features for the design. Setting up the behavior required an unexpected depth of programming logics, with a lot of nested what-if statements that affected many of the elements as the prototype grew.

As aforementioned, a lot of decisions regarding the interactive behavior were taken in this stage. Not only did the prototype allow us to experiment with rich visual modeless feedback and hover effects, but it also enabled us to simulate the drag and drop function of the candidate cards in the Candidates in a Job view. It also helped to visualize the flow of action better and highlighted where, for example, error messages or the disabling and enabling of certain functions were required, depending on the current task context of the user. Moreover, it was an effective discussion initiator and acted as catalyst to contemplate and determine a standard behavior of buttons, text fields, and menus.

5.4.5 Usability Tests

By means of the interactive prototype, we conducted three sets of tests, both with recruiters – the main target user group – but also with other people who were not involved with recruiting, to collect more diverse feedback. The main goal of the tests was to see whether users understood the core functionalities, such as the Candidates in a Job screen, where the drag and drop card view and list view are available together. For the usability tests, the focus was on the main parts of the system, where the interactive elements were not common in other web applications. It was more time effective to not put so much emphasis on testing for already established and commonly known interactive concepts, and the time to develop a fully fledged prototype that contained all parts of the wireframe would have been outside the scope of time.

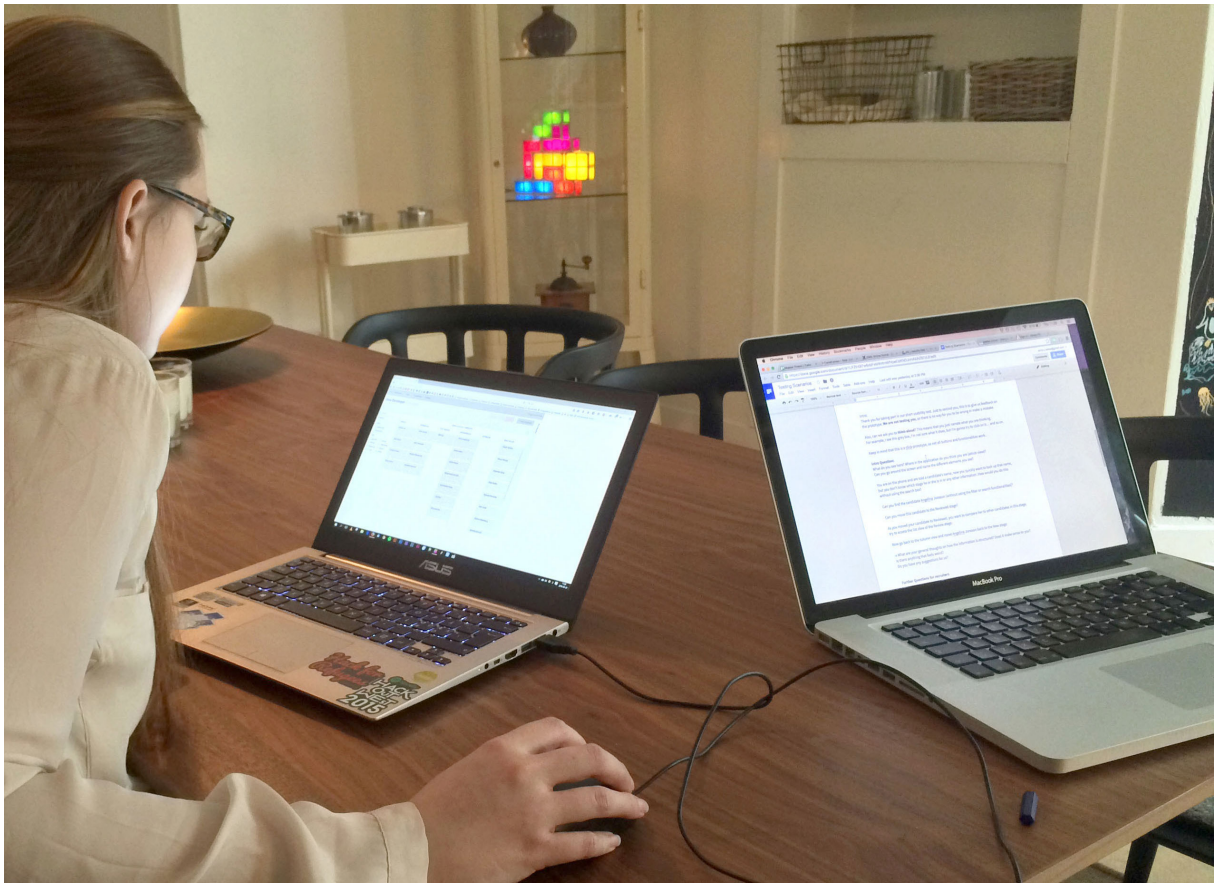


Figure 5.8: Usability testing with a recruiter.

The screens that were interactive and accessible for the tests were the Candidates in a Job, and the three contained views that could be toggled (drag and drop card view, table view of a specific stage, and table view of all candidates in a job), and three exemplary candidate detail cards. It was possible to move these three candidates to different stages and view the result in the different modes.

To conduct the test sessions, we conceived a scenario where tasks covering all of these elements for the test participants were embedded. As the test participants were interacting with the prototype we asked questions and subsequently gave new instructions. In the introduction of the test we also asked the participants to think aloud so we could note down opinions, instances of hesitation and getting stuck, and questions. We also made it a priority to make the participants feel comfortable and at ease by explaining that we did not test them or their ability, and instead it was their task to try and break the prototype.

We conducted three rounds of tests with five to seven participants in each, analyzed the feedback and improved and changed the prototype as well as the script, before the next round of tests. This iterative approach offered the possibility to focus on new areas of the prototype. The downside was that five to seven participants per round was not enough to yield decisive results on some points. In these cases we proceeded thus and considered the occurrences where two participants struggled at the same spot, to be an indicator to reconsider the layout or presentation of the given interactive element.

The changes of the prototype were minute between the first two rounds of tests, and affected mostly the hover states to signify the view toggle options in the tabs. Also, the controls of moving a candidate to a new stage was optimized after each round. During the initial round of tests, only one candidate realized that the timeline design (see figure 5.9) was an interactive surface. A vast majority – eleven out of the fourteen participants in the first two rounds – registered the element, but thought it was merely an indicator.

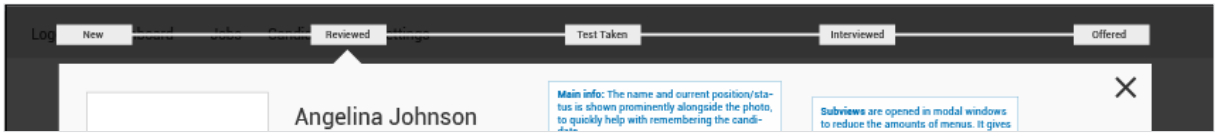


Figure 5.9: Version 1 of the assigning a candidate to a stage functionality in the wireframes.

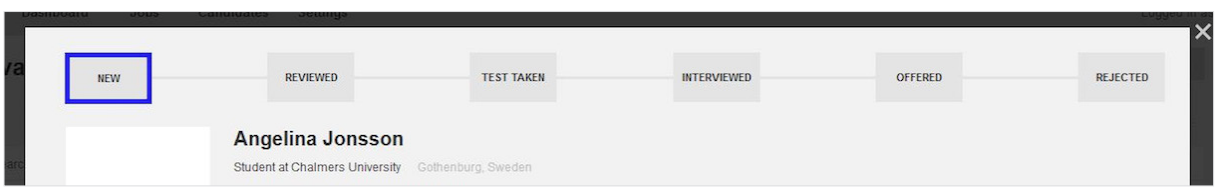


Figure 5.10: Version 2 of the assigning a candidate to a stage functionality in the Axure click prototype.

Therefore, the design was adapted to the next round, for the click-surfaces to be bigger, and the current active stage marked with a blue border (see figure 5.10). This increased the test participants attention to the element, but the results were still mixed, as the missing feedback to the executed action was insufficient.

The last round of tests was conducted with the four recruiters working at Software Skills, and Henrik, the CEO. It was striking how much the tools they currently use influenced their expectations of interactions in this new prototype. While the previous test participants were clean slates when it came to recruiting and had never used such a system, their expectations were only based on other web and desktop applications they had encountered before. For the usability tests with the recruiters, this meant that for the assigning a candidate to a stage functionality three out of five tried to click and drag the frame to the next field. That marked the point when we finally opted to abandon that feature and instead go for a more conventional button design (see figure 5.11).

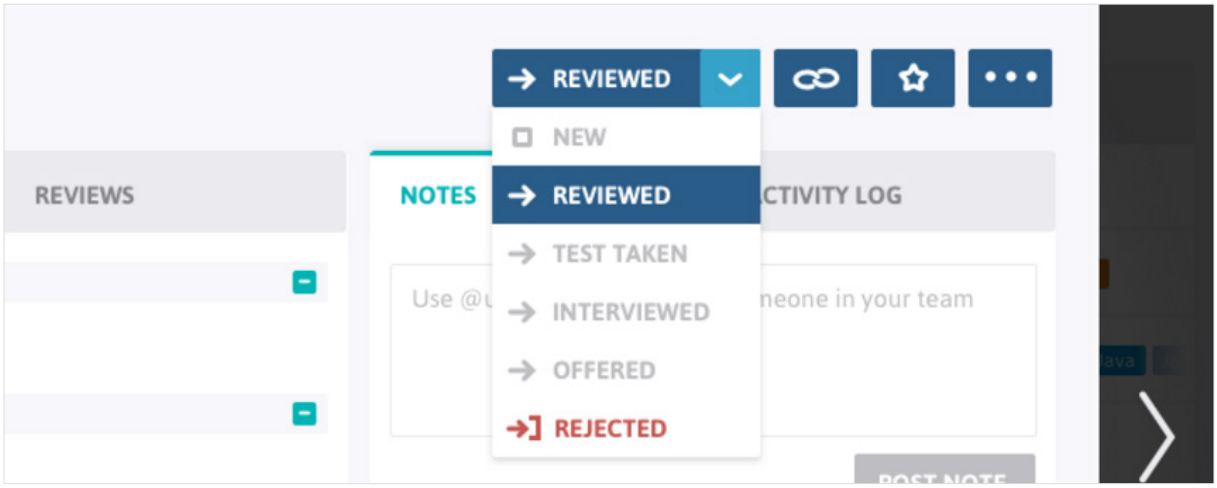


Figure 5.11: Version 4 of the assigning a candidate to a stage functionality in the first design iteration.

5.4.6 Requirements Check

To ensure that all previously discovered aspects were actually addressed in the prototype, we performed a requirements check, which entailed comparing the stipulated requirements with the prototype. This comparison warranted checking that functionalities, parameters, elements, and other content were included in the prototype. Where inconsistencies or gaps were discovered, we appended descriptive texts of what to add and remedy in the next design iteration. Since we had developed the prototype in regard to the requirements list most of the functionalities were present, at this point it was more about details. For example, making sure that the filter options actually corresponded perfectly to the parameters visible in that particular screen, as there are several very similar screens, that are only distinguished by those few details, and things had been copied over and not adapted properly for each view. On top of that, the method also revealed questions to parts of the design that needed to be answered by Henrik Enström and the recruiters.

5.5 Design of the Prototype Part 2

The fifth iteration cycle was a continuation of the design implementation of Honeycomb ATS. We initiated it with a card sort, applied as an inspiration type of method here. It had not been scheduled in the project plan at this particular time and had been intended to be used as an evaluation method, but it was a suitable way to kickstart the iteration by getting some fresh insights. The findings were subsequently implemented in the interface and the thus updated wireframes were subjected to an validation scenario evaluation. Subsequently, the final stage of the design development was initiated by creating style tiles, followed by the design implementation.

5.5.1 Card sort

The card sort method was used to inform which parameters the recruiters find most important to overview and compare when taking decisions regarding a candidate in the evaluative stage of the recruitment stage. We let Henrik and four recruiters rank, sort, remove, and add different pieces of information according to what they deemed most or least important for certain circumstances. The information was written on post-its that the participants were told to arrange on a piece of paper with a number scale.

Figure 5.12: Card sort results of a recruiter on the left, CEO's card sort results on the right.

The purpose was to determine the pieces of information displayed on the cards in the card overview and the table rows in the list overview. Because we were aware from our previous experience and methodology research that it would be difficult to ask about these two things specifically and get useful answers, we used the card sorting to retrieve

this information indirectly. This involved attentive listening, and asking follow-up questions on specific statements, to find out the reasoning that the participants based their decisions on. So this worked rather as a discussion tool about their work process, especially how they search for candidates, and the different scenarios surrounding these activities.

The method turned out to function as a collaborative brainstorming session, too, because a lot of the parameters written on the cards were not available in the current system. This led the recruiters to imagine how these pieces of information would influence their workflow. During one session, we were introduced to the importance of indicating if new content has been reviewed: One interviewee stated that "sometimes I open candidate cards several times in the New list, because I forgot I looked at them already." This was the impetus from which the *Aliveness Indicators* were developed later on.

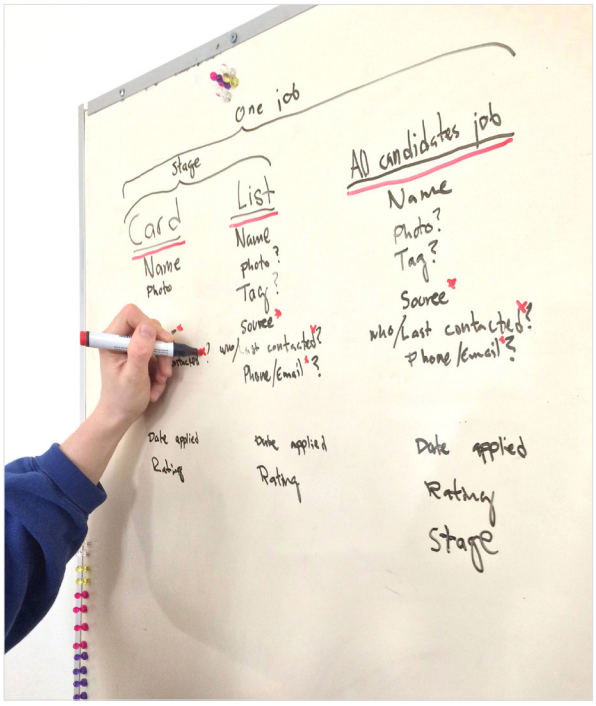


Figure 5.13: Reevaluating candidate parameters to be shown on different screens.

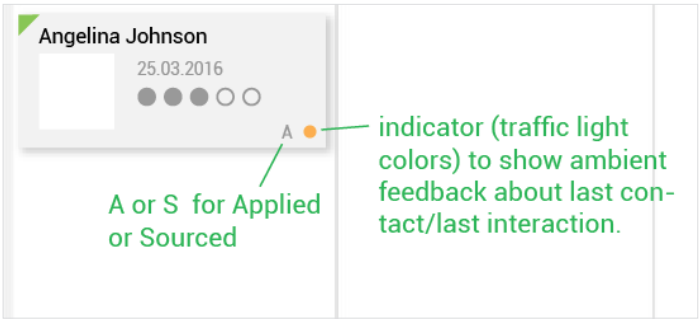


Figure 5.14: Card View, with pieces of information determined.

A number of useful findings and discussions sprung out of this method and led us to re-evaluate some previously taken design decisions. For instance, the hierarchical importance of information was reconsidered and also which tab should be visible by default in the open candidate card. Based on the results, we analyzed and worked out a new plan for which data to prioritize. Figure 5.13 shows the activity of ranking of parameters for each screen, and figures 5.14, 5.15, 5.16, and 5.17 are examples of the corresponding manifestations in the interface.

5.5.2 Validation Scenarios

The last method employed before creating the final prototype design, was validation scenarios as described by Cooper et al (2014). By constructing seldomly occuring but possible

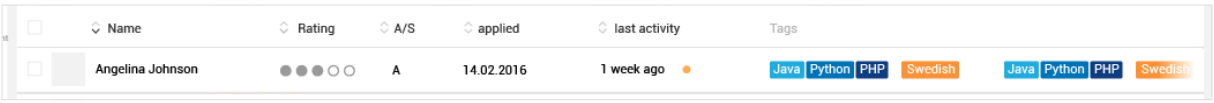


Figure 5.15: Data for the List View, specific stage.

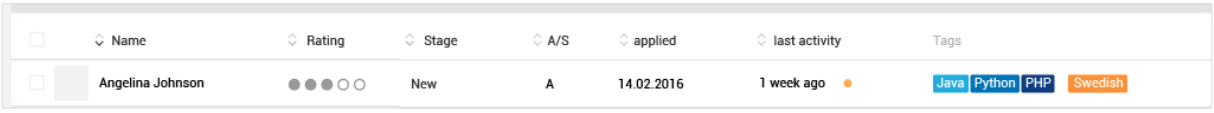


Figure 5.16: Data for the List View, all candidates in a job.

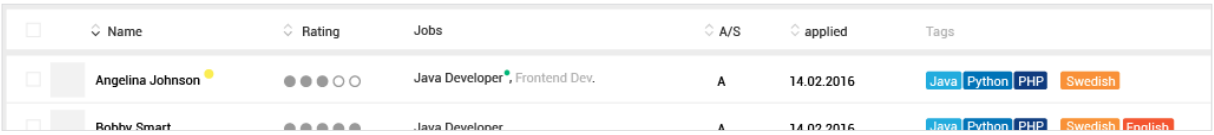


Figure 5.17: Data for the All Candidates screen.

use cases, conflicts where workflow or functionality of the application were either redundant or insufficient were to be revealed. Furthermore, the goal was to question the current status of design and our choices to see if there were any superior and/or alternative options. The validation scenarios were short to prioritize quantity over quality, because the main point of this activity was to initiate discussions. Here is an example of a scenario that we constructed inspired by the interviews and contextual inquiry:

> *The recruiter Hanna works alone today since her co-worker Frida is on vacation. Frida has left a message for Hannah to find a candidate with rare combination of skills such as Python, Java and UX design experience. Hannah believes she recalls a candidate she interviewed a few years back who had a similar skill set, but she does not recall the candidate's look or name.*

This method proved especially important for our search and filter options. We realized that there were many situations that could make it difficult to find odd information with little to none prior data knowledge of the information desired. Hence, it led to adding advanced search available on request. The various filter controls were adjusted to satisfy the needs identified by the scenarios. Furthermore, the importance of being able to edit the information on the candidate cards directly if new information emerged or turned outdated was highlighted. A result of this design decision was to deal with the issue of having multiple users editing the same content at the same time. The pattern *Floor Control* was introduced as remedial measure, informing users in real-time if someone else is editing the content one is currently viewing – an issue first discovered during the awareness mind map. This is aims to prevent disorientation caused by sudden state changes and conflicting editing commits.

Applying this method demonstrated that content provided by applicants could be overridden when new attachments are uploaded. This could also confuse recruiters since they might not recognize the candidate's CV or cover letter if it had been exchanged. Therefore, all submitted information is now sorted in different versions and attached to the candidate's system ID. Additionally, the method revealed situations benefiting from a different type of layout. For example, the sliding panel containing the activity log was adapted to fit into the layout better, where it had previously covered bits of information that could potentially be important to see simultaneously with the activity log. The validation scenarios therefore informed changes in the wireframes, given they did not affect any other more likely scenarios, and made the subsequent design implementation more effective.

5.5.3 Style Tiles

In order to determine the look and feel of Honeycomb we conducted Style Tiles. The first stage of this method was undertaken individually (see figure 5.18), i.e. we did not explicitly share our thoughts beforehand, in order to get two subjective views, and then subsequently be able to merge them. The motivation for doing it individually was to democratize the design process, instead of doing it jointly from the beginning, so that each of us had the opportunity to explore our own understanding of the product.

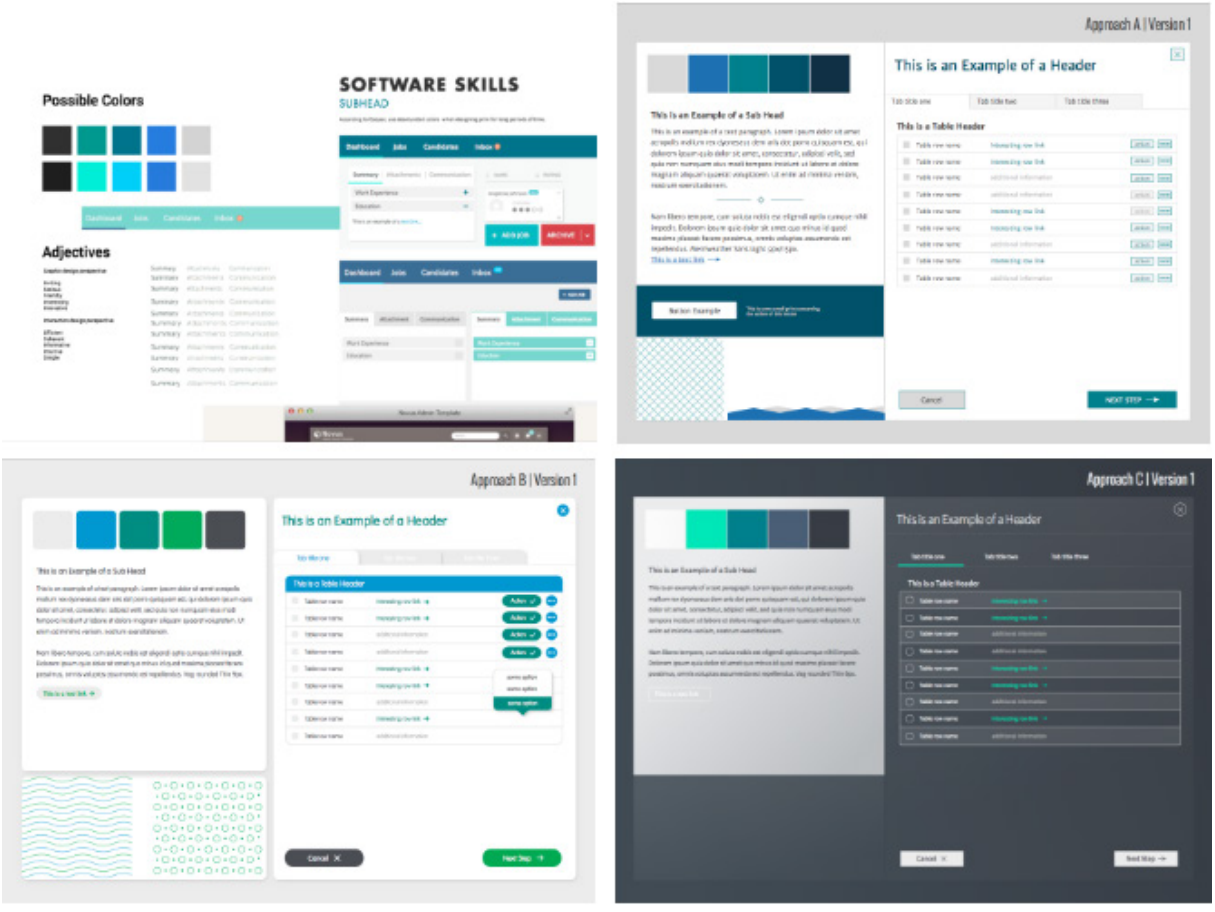


Figure 5.18: Four different Style tiles, according to the following adjective clusters: 1: friendly, innovative, efficient; 2: professional, powerful, robust, trustworthy; 3: easy, friendly, efficient, accessible; 4: forward-thinking, potent, cutting-edge, dynamic.

Both of us thought of sets of adjectives that we felt – according to our previous research of ATSs and knowledge of the stakeholders – should be visually and experientially communicated through the interface of Honeycomb. Particularly the answers regarding the SMEs' *life and experience goals* according to the GDD were used as a basis for these deliberations. Then we translated these adjectives into visual representations. For example, the adjectives "trustworthy" and "friendly" may be related, but they may also be interpreted differently when combined with further adjectives, like so: "trustworthy" and "serious" versus "friendly" and "fun". The first combination may then end up looking more corporate, with darker colors and an efficient layout. The latter combination in contrast, may be implemented with brighter colors and more playful details.

The second step was to get together and discuss our thoughts about why we chose those adjectives and how we had translated them into visual elements. Subsequently, we selected a final set of adjectives: *Trustworthy, friendly, efficient*; and their connected design attributes. Perhaps we should have specified the interface elements we used to illustrate the design beforehand, so that both of us had the same, and so that we could compare more easily, but with the adjectives it was clear which direction to pursue. Stipulating these descriptive adjectives meant to make more explicit of what kind of an image we had in mind for the ATS.

5.5.4 Design Implementation

The design approach determined in the style tiles method was applied first to a test screen in Sketch. The reason for switching from Adobe Illustrator to this program was twofold: Firstly, it pairs well with InVision App, because it recognizes Sketch layers as separate elements, which can then be linked and animations added to produce click prototypes, as a way of showcasing the design and functionality of the whole system. Secondly, Sketch itself has some good features, for example, to set up a general grid, its way of handling global colors, or exporting assets for different resolutions very easily.

In the first test screen, aspects like exact element sizes (of e.g. Buttons and table rows), text sizes, colors (tested for contrast and color blindness) and exact styles (border radius, general icon style etc.) were evaluated on whether they fit well together in terms of hierarchy and space. The grid had to be adapted several times to accommodate the sections of various sizes on different screens and stay consistent throughout. Also, the color codes of primary colors, secondary colors, feedback colors, and background colors were adapted in several iterations according to which elements they had to be applied for and how the contrast worked in each case. When this was warranted, hue, saturation, and brightness were slightly tweaked to achieve a pleasant overall color scheme that was in accordance with the previously determined brand value adjectives. Once all of these visual parameters were finalized, they were applied screen by screen, to the most central screens first. This was partly due to the Bachelor thesis group waiting for the screens, so they could code the final design into the backend that they had been building up to this point. At this stage, we

were in frequent contact with the bachelor group students and sent them each screen as it was completed. They in turn had a number of specific questions on how certain elements behaved dynamically, and made us aware of some inconsistencies in terms of the interactive components, which we would then improve. In this way, the cooperation with the programmers was very helpful and their fresh set of eyes created small-scale evaluation–implementation cycles. Generally it can be said that the more finalized and deliberate the respective wireframe was, the faster the design implementation occurred. Also, towards the later screens the implementation was faster, because by then many elements were already designed and just needed to be assembled and adapted accordingly.

It was also during this final stage of the prototype coming together, that we conceived a name for it. Relating to an existing Software Skills product called *Honeypot*, we brainstormed the name *Honeycomb*. Its structure relates to how the candidates are stacked in a specific order in the ATS and that it should be a safe place for its contents – a tip of the hat to data security and privacy.

5.6 Bringing together Patterns and Prototype

This marks the final iteration of the design process and reaching the final results for both the pattern collection and the design prototype of Honeycomb. It also is the stage in our process that deviates most from the project plan, as we did not formalize how exactly we would conclude the pattern collection, as we could not anticipate the type of result exactly or its level of completeness, and therefore the methods that would be appropriate at this point. As a last undertaking we conducted a final review of the patterns in conjunction with the designed prototype. Once the final pattern collection had taken shape, we mapped its components against the parameters specified in our research question at the beginning, which brought about an unplanned, additional result: the ACE (awareness, collaboration, efficiency) model which is presented further in section 6.2. The final section briefly recounts how we documented each pattern's evolvement. As this was a continuous process throughout the project, all the connections to the undertaken inspiration, ideation, implementation, and evaluation activities that have been previously mentioned are made explicit here.

5.6.1 Heuristic Review

As discussed in the methodology section, during an heuristic review a given result is cross-checked with a pre-stipulated set of requirements and parameters, to eliminate any lingering inconsistencies. In our case, this meant meticulously combing through each screen and mapping its components against the requirements Coggle map. Based on this, we adapted the last changes in the interface, especially streamlining which button options should be on each screen. Apart from looking at the ten points laid out in the method, we also undertook an additional review, matching functionalities found in the interface with

our preliminary patterns list. This revealed further concepts that we decided to include, for example, *Shareable Item Link*, and through discussion eliminated others as irrelevant, for example, *Active Neighbors*. When we had completed the thorough review, we turned our attention to complete the patterns still on the list in several iterations. First, we finalized the form to be: problem, context, solution, consequences, danger spots, why, example, and related patterns, borrowing from the previously studied pattern collections according to what suited our findings best. Most specifically, the headings "context" and "danger spots" were inspired by Schümmer and Lukosch (2007), "consequences" by Gamma et al. (1994), and the providing a reasoning for "why" we found helpful in Vora's "Web Application Design Patterns" (2009). Next, we polished the text in several iterations to be consistent across all patterns. Here it was really helpful to standardize some of the wording, for instance, all of the context descriptions begin with "Apply this pattern when…". Finally, we matched all the relationships among the individual patterns. For the latter activity, we wrote each pattern name on a post-it and placed them in relation to each other. Subsequently, we drew three kinds of arrows between them, based on the *containing, complementing,* or *contradicting* relationship types based on van Welie et al. (2003), described in section 2.3.1.

Inspired by this accomplishment and by seeing the fruits of our labors fall into place and interlock with each other in this manner, we elected to perform yet another evaluative endeavor. First, we considered a hierarchical pattern structure by the likes of van Welie et al. (2003) or Kruschitz et al. (2010) as introduced in 2.3 Pattern Languages. However, because our collection presents very tangible patterns that are each tied to specific interface embodiments, it was conceded this would not yield a stimulating arrangement. Returning to the research question posed for this project, we used the keywords *awareness, collaboration*, and *efficiency* (ACE) contained therein and tried to map the patterns to them in different ways. The next attempt at a mapping was to create a matrix of two ranges of parameters. The x-axis represented "low efficiency" on the very left and "high efficiency" on the right side. The y-axis was labeled "low-awareness" at the bottom and "high awareness" at the top. Once we started, we quickly realized that this plotting method was not especially exciting, since we had already eliminated all patterns located on the two left quadrants – because they caused *inefficiency*. The next attempt endeavored to utilize the three keywords to buttress a triangle structure. Based on the just suffered failures and consequential learnings, we anticipated that three axes, instead of two, would provide stronger indications of how to arrange the post-its.

Figure 5.19 shows the final arrangement of the triangle model. The distribution is relatively even, which is an indicator that we managed to address all three issues to at least a certain degree. We are aware that the positioning is only relative and no pattern has definitive coordinates, but the model provides a good overview of the patterns and their respective affinities. This activity also confirmed the level to which we had managed to answer the research question by this point.

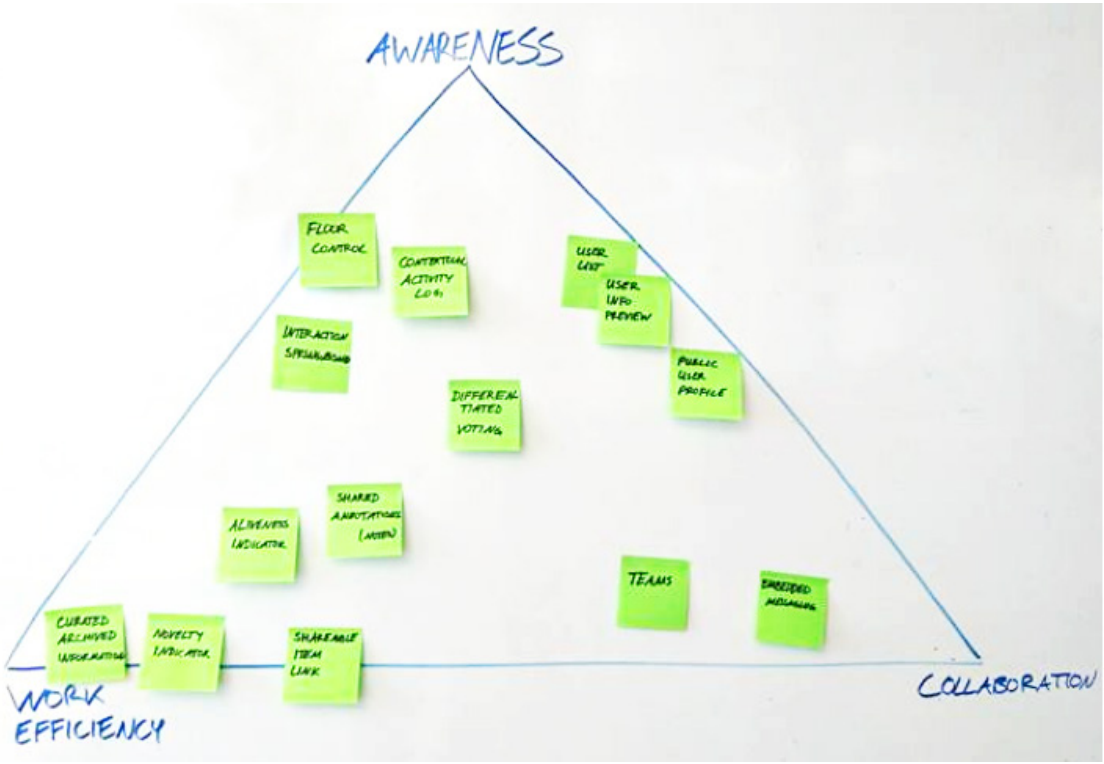


Figure 5.19: After much trial and error: The result of the pattern mapping.

Apart from the pattern model, this activity had yet another result: Wielding the patterns in this manner and consequently considering them both individually and as a holistic construct from different perspectives (in terms of relationships, hierarchy, awareness, collaboration, and efficiency) meant we noticed yet another constellation. The patterns could also be grouped according to their functional purpose – rather than the conceptual purpose of *ACE* – and two major categories emerged: Concepts related to people in the system, that we called *User Management*, and concepts related to *Information Sharing*. Some of the patterns can belong to both of these, while most are either or. This grouping with emphasis on direct functionality may help potential users make sense of the patterns more easily and is illustrated in the results section with figure 6.14.

5.6.2 Pattern Evolvement

As we studied the previously described "Patterns of Computer-Mediated Interaction" book (Schümmer and Lukosch 2007), we began connecting our already diagnosed examples with the patterns of the book. Continuing on the design of Honeycomb and reviewing old method results further contributed to the growth of the preliminary selection. The final evaluations portrayed above led to the collection as it is presented in the results section. To further corroborate the validity of each pattern, we added the heading "Evolvement", where the conception and journey of each pattern, that has been scattered throughout the process, is summarized and referenced. Most of the patterns were already identified by the time we got hold of the information from Schümmer and Lukosch. *Floor Control* was the only one by them that we added afterwards that made it into our final collection, which was not found through our process. For numerous other patterns the Schümmer

and Lukosch collection helped concretize our descriptions. This further scientific source also validated the patterns in terms of awareness. There were also several other patterns, excluded because they did not match the requirements and structure of the Honeycomb ATS. We had evaluated further concepts included in Schümmer and Lukosch's collection, on their applicability: *Active Neighbors, Remote Selection, Timeline, Periodic Report*, and *Flag* (2007). They may be relevant to other cases of designing an ATS or similar system, and are therefore included in appendix 4. The subsequent results section presents the prototype design, and the pattern model containing the fifteen patterns.

6

Results

As aforementioned, the deliverables of this thesis are twofold: The pattern collection for applicant tracking systems (find detailed descriptions in section 6.3), and the thus designed Honeycomb ATS. Both artifacts matured simultaneously and informed each other throughout the development process. Additionally, the pattern collection has been constructed into a model, which will be outlined before the patterns in section 6.2.

6.1 Honeycomb ATS

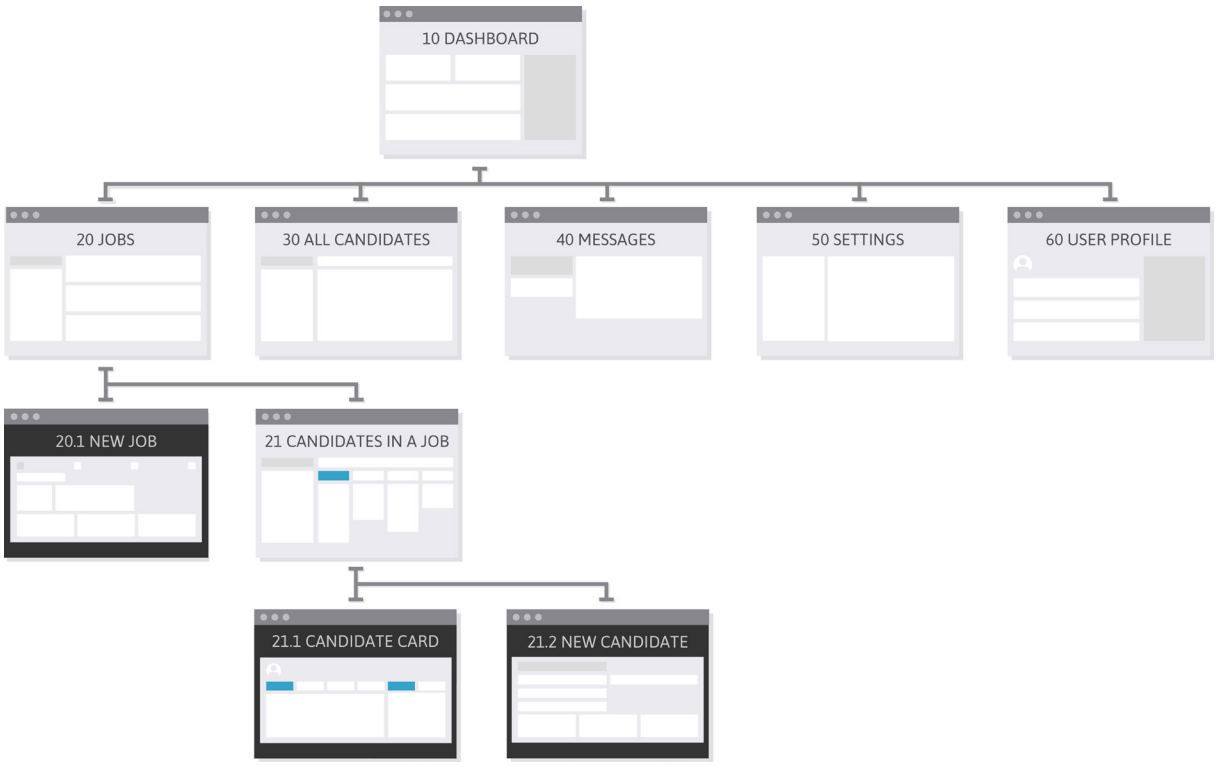


Figure 6.1: Sitemap of all main views and the most important modal views (tinted backgrounds).

The ATS flow is informed by the application and hiring process as outlined by Holm (2012) and Lee (2007) discussed in section 3.2, and on the stakeholder requirements. This includes everything from setting up the job posting with required prerequisites and description of the role; to receiving, sorting, and reviewing applications; to managing rejection and hiring of candidates. The system also deals with archiving and storing old information that can become relevant after a given hiring process has been completed. How the different parts are navigable and are connected to each other has been carefully thought through, resulting in the majority of the views being reachable from the main menu to avoid menu diving. This gives users the impression of an easily-graspable system and makes it simple to create a mental model of the application's structure.

6.1.1: 10 – Dashboard

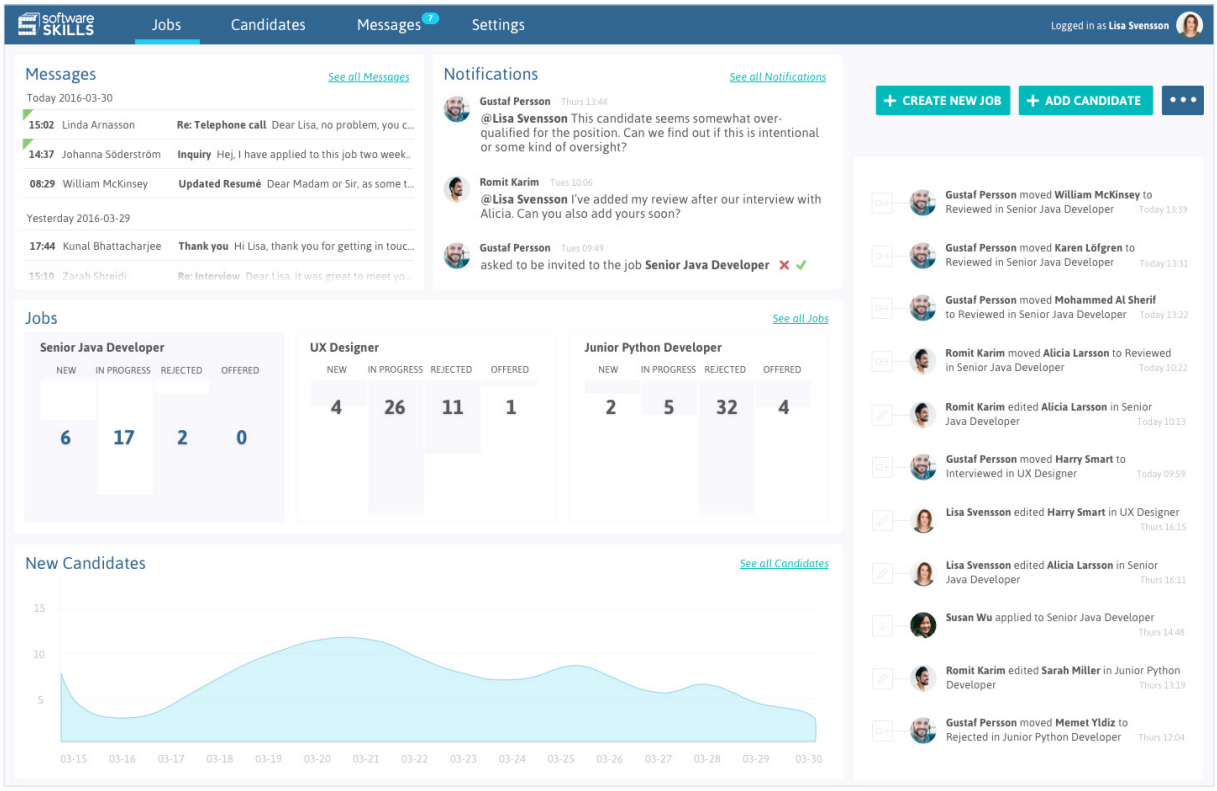


Figure 6.2: 10 – Dashboard is the start view of the application.

The dashboard view was determined to be the start screen, because it displays information about the present state of various relevant processes, and previous and upcoming activities. It is intended to make conveying information about the current workload and to-dos for the various users more efficient. On top of that, it serves as a curated display of entry points into various parts of the system and thus corresponds to the *Interaction Springboard* pattern. *Messages* and *Notifications* contain information of other team members' activities relevant to a given user. By incorporating these features, the recruiter is always up to date with the latest information and third party solutions are not needed. The idea of this feature is to support the user's recollection and information intake, thus preventing cognitive overload.

Notifications are comprised of new mentions (users may mention one another to point towards a note) and team invitation matters and offer direct access to the listed items. It is an alternative to communicating with fellow users in the system: It is faster than e-mailing and messaging, since it points directly to the concerned location or artifact.
The field where *20 – Jobs* are represented, highlights a selection of the active and most urgent hiring processes. Each process is illustrated with dynamic bar graphs, which is a visualization metaphor referencing the columns in the main screen of *21 – Candidates in a Job.* The height of the bar is proportional to the number of candidates in a given column – representing a hiring stage in the job – and therefore is visually reminiscent of the number of candidate cards stacked in the columns in the *21 – Candidates in a Job* view. Clicking on the job's title will transport the user to the specific job page (21 – Candidates in a Job), whereas the "see all Jobs" links to *20 – Jobs* – an example of how the dashboard conveniently provides information paired with contextual, quick access to specific places.
Below this section, a graph presents data about the progress of all ongoing processes, namely how many candidates have been applying over time. This is an indicator for how well the job postings are perceived and whether more marketing measures are needed to engage more people.

The part furthest to the right is a *Contextual Activity Log*, accessible in all process related views of the application. Information in the log is tailored to the particular view users are in – therefore contextual. It filters information so that only relevant activities for the current view are displayed, thereby easing the cognitive load for users. The log also makes the users feel more comfortable with exploring, and being more daring with the risk of making mistakes, since it is possible to backtrack changes and thus reverse possible errors. This also addresses the stakeholder requirement of catering to flat organizations and teams. Overall, the *Contextual Activity Log* offers more shortcuts into the system, as well as the obvious awareness of others' activity. Actions are consistently displayed in the top right corner throughout the application for discoverability. It displays the possible actions for every part of the application. For "first time users" showing available functionality communicates the purpose and possibilities of the application. On top of that it acts as a point of reference, to complement the top bar, by informing users of where they are in the application.

6.1.2: 20 – Jobs

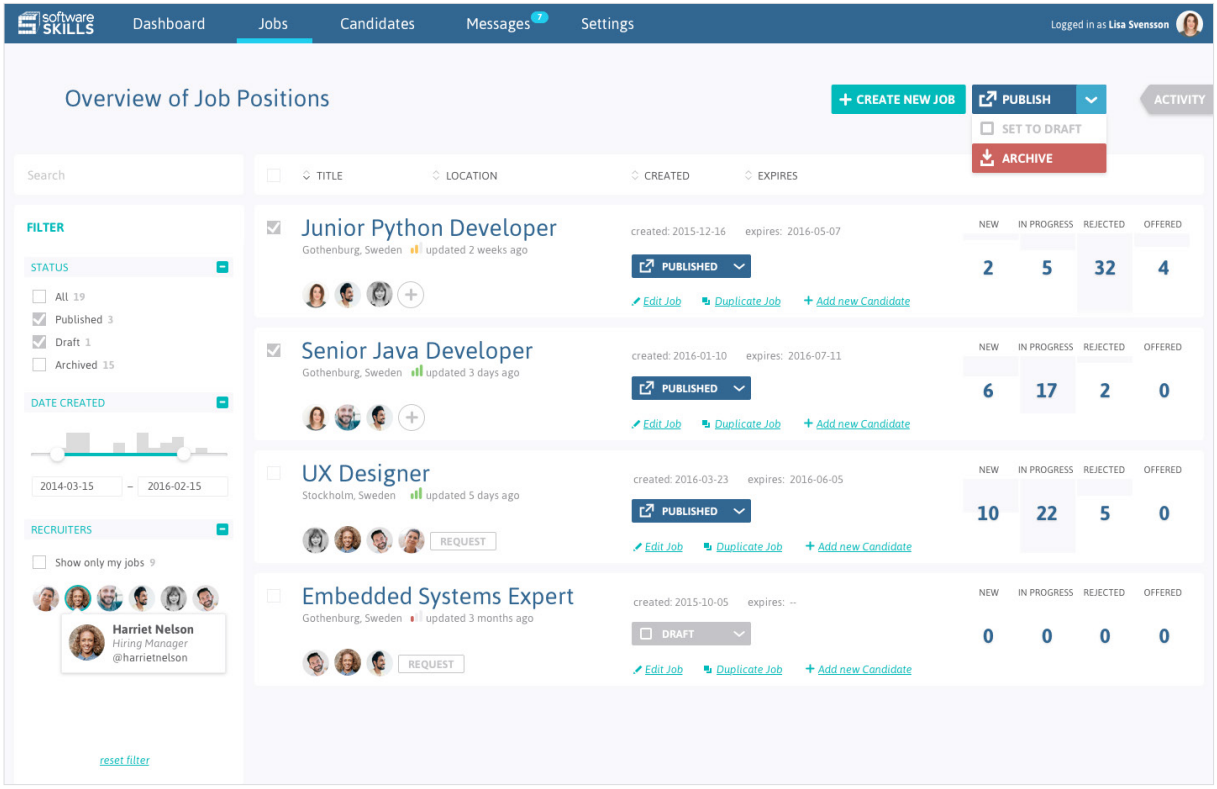


Figure 6.3: The 20 – Jobs view displays all hiring processes, upcoming, active and expired.

The *20 – Jobs* screen presents all hiring processes that were set up in the application. This is where users create new adverts for jobs, find active, expired, archived jobs or hiring processes that are saved as drafts. The page is equipped with sort, filter, and search features that support the findability of jobs according to various parameters. These have been determined through various methods like the previously described interviews, card sort, and usability testing. Each job in the list displays the set of recruiters involved in the process. Since the system promotes flat organization structures, which entails trust and individual responsibility the users can – if part of a group (see the Team pattern) – invite other users without the need of permission from higher instances. If a user wants to join a team it is possible to request access, and the concerned group then has the option to accept or deny the request.

More than offering quick accessible actions in the overview, the list displays visual feedback on each process. The bar chart presents how far the applicants have progressed in the process, and at which stage most of them are currently sitting. The information and possible actions are also available in the details of a job. However, this overview offers the possibility to not only compare the various processes, but to perform bulk actions that make it faster to handle a large number of job openings. The general layout of elements has been adopted from the most common *21 – Candidates in a Job* screen for coherency and familiarity.

6.1.3: 20.1 – New Job

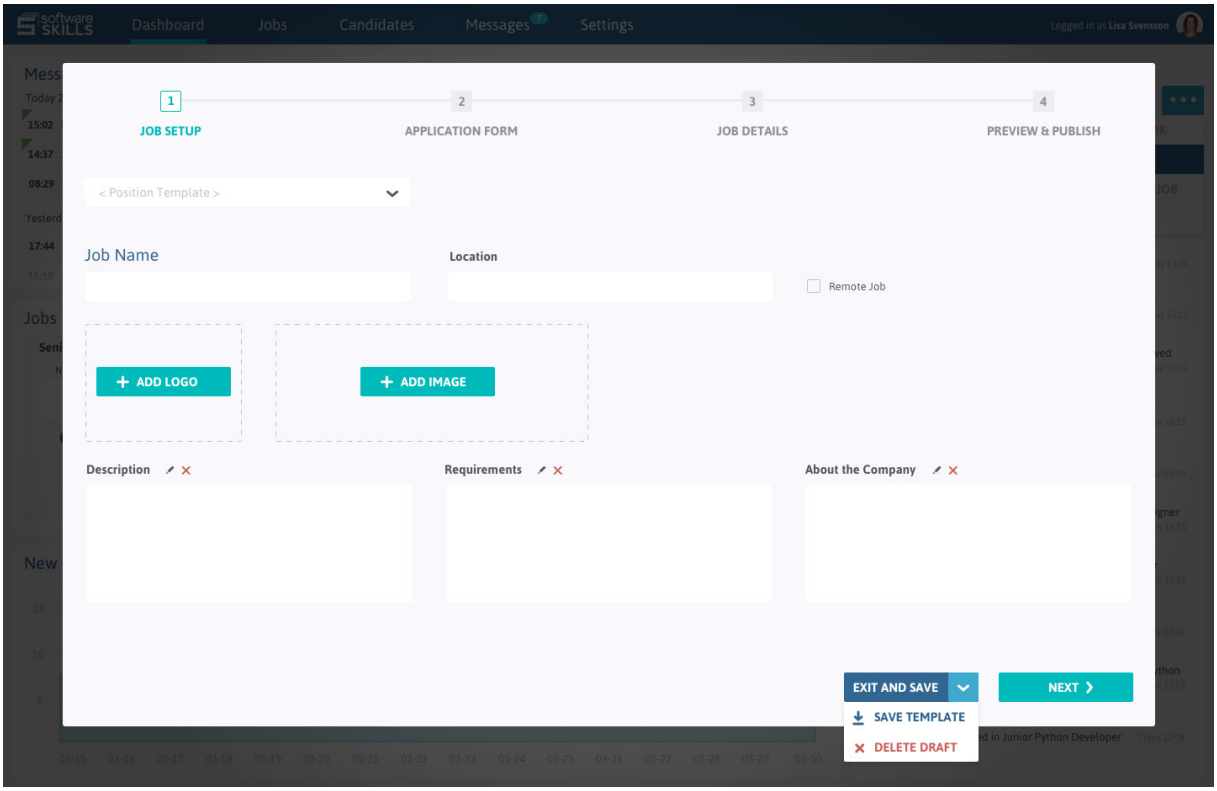


Figure 6.4: The 20.1 – New Job (1) view provides input fields to create a new job opening.

In the *20. 1 – New Job* modal view, a new job opening can be created, which will then be displayed publicly for applicants. This set up phase also manages how the internal process will be structured and carried out. The first step corresponds to the initial information that is shown as the job advert. With the dropdown control it is possible to select from a list of templates saved from previous job postings – a time saver when the job parameters are similar. Below, various dedicated input fields can be used to streamline the information, or can be removed if not needed. At any point, the user can save the current state as a draft to continue working on at a later stage, or save as a template, or delete the draft entirely.

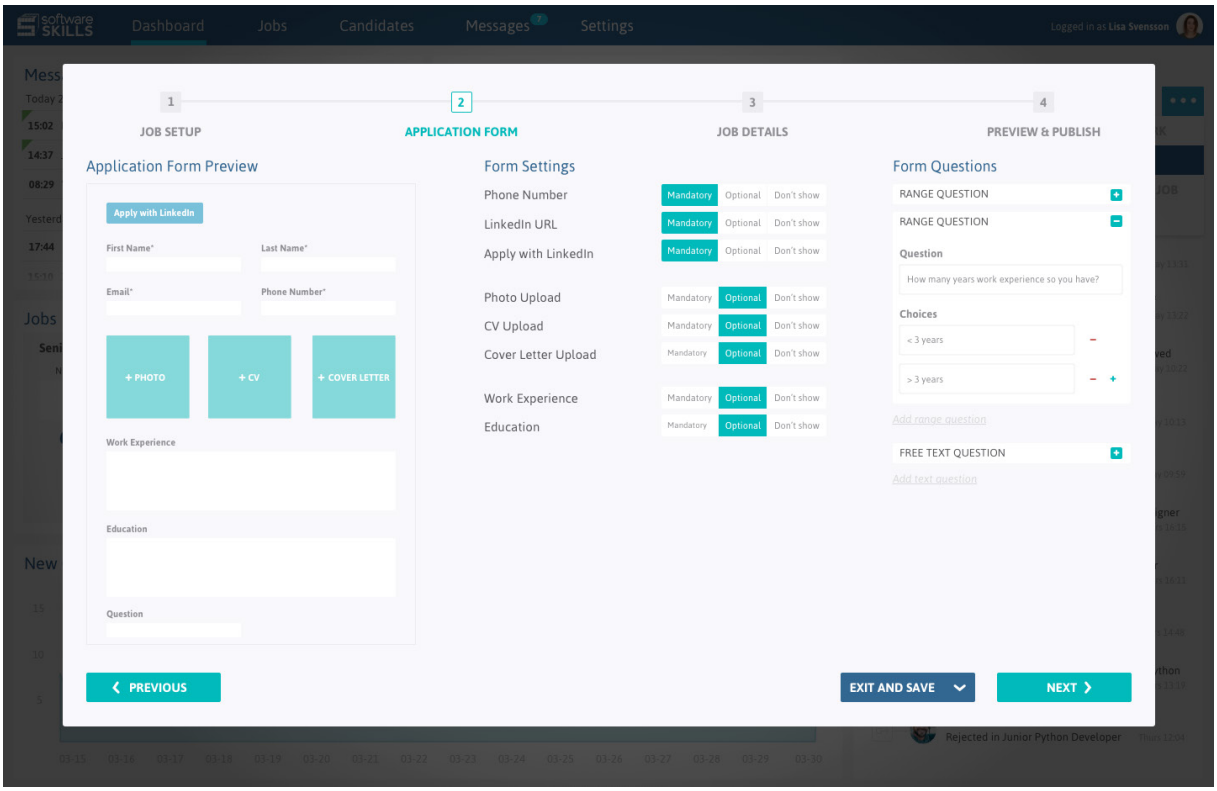


Figure 6.5: The 20.1 – New Job (2) view is for customizing the application form.

The second stage of the setup process corresponds to the information that is required from the candidates. Here, the recruiters can customize the input fields candidates are required fill in and all fields can be designated as mandatory, optional, or be removed completely. Recruiters can also add questions that are interesting for specific positions, these can be defined to be answered as free text, as range question with a slider input, or as multiple choice inputs. The left side displays a live preview of how the application form will look once it has been published.

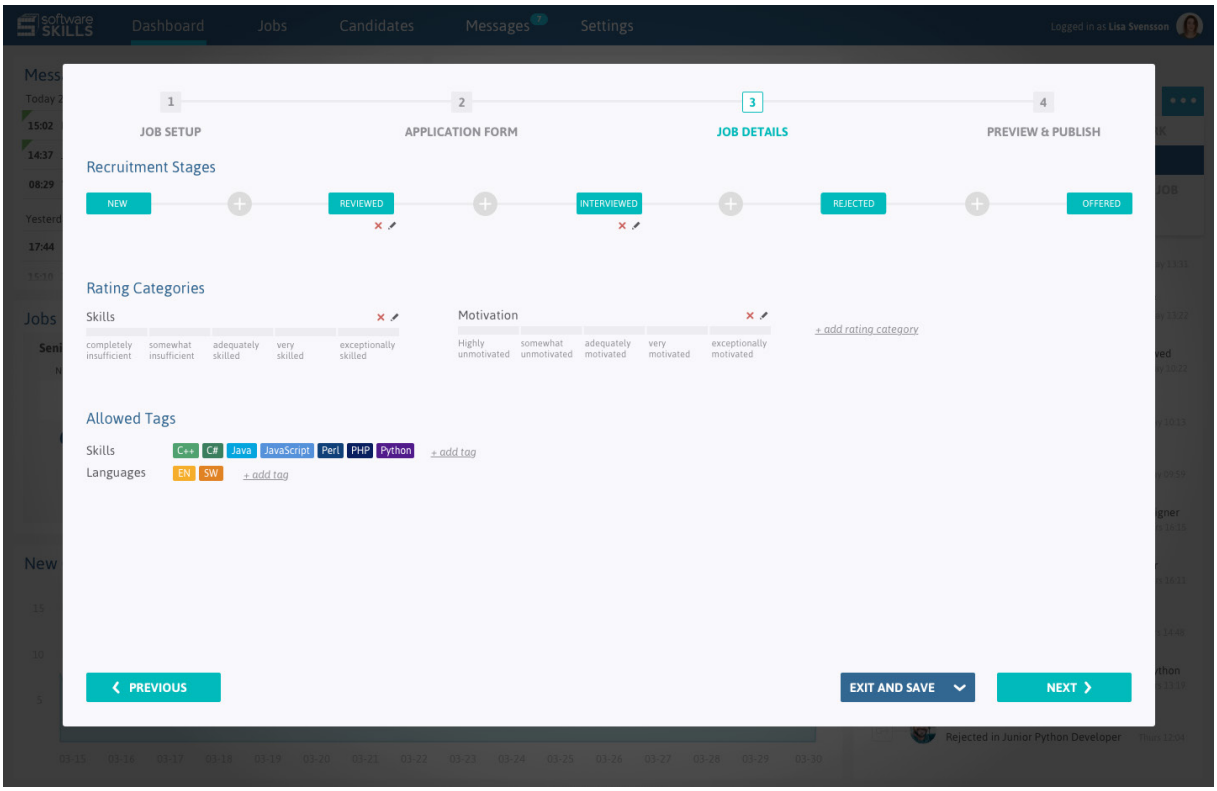


Figure 6.6: The 20.1 – New Job (3) view presents options to customize the recruiting funnel and settings.

The third part consists of setting up the internal work process. All options can be changed in a later stage, but a default structure provides the recruiting team with an understanding of what is important for this particular job opening. First, the user establishes the various hiring stages: whether one or two interview stages are needed, or rather an initial phone interview followed by a test phase, is deemed the most effective way to determine the best applicants. Second, the categories that are appropriate to rate the candidates by are settled and augmented with a corresponding scale, which tell the recruiting team what qualities to look for when the review and interview candidates. In the third and last step, the user is to decide on the specific skills that are desirable for a potential hiree to have, which are added as tags. These are then linked to the candidates and shown on their profiles to simplify locating relevant applicants in the system at a later stage.

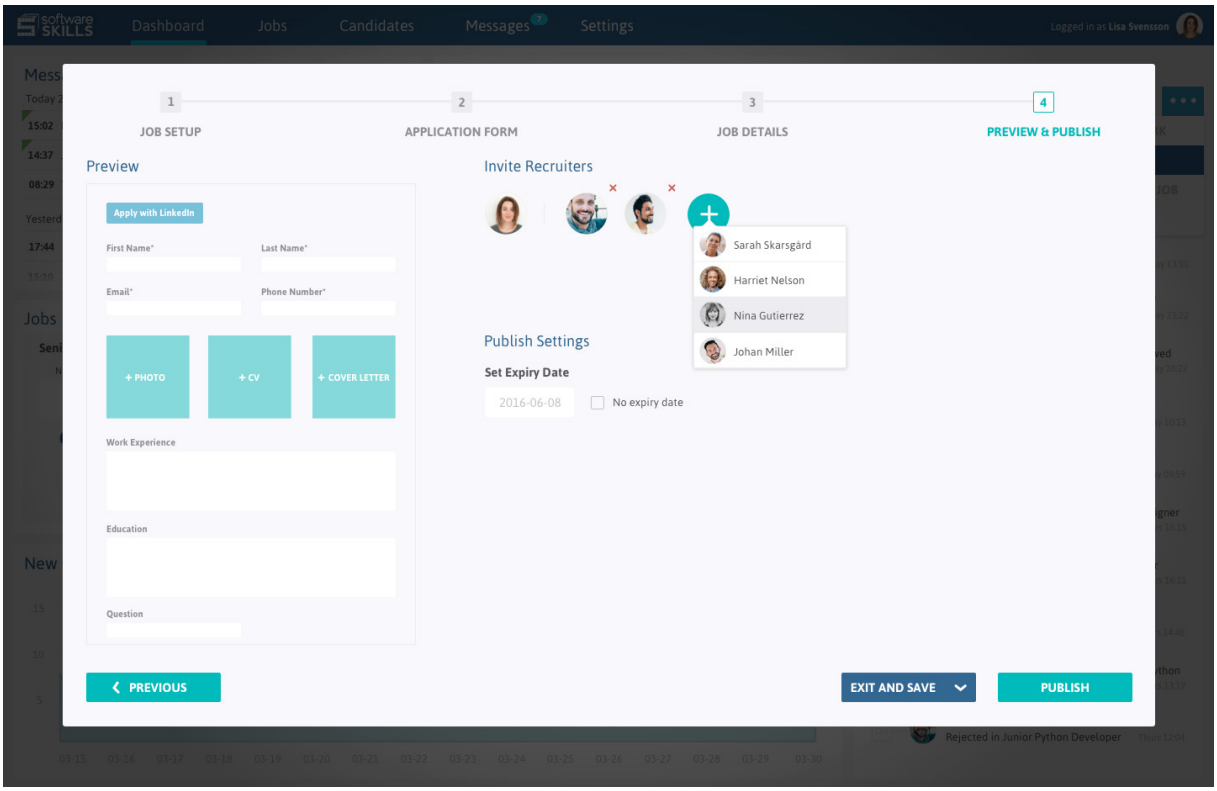


Figure 6.7: The 20.1 – New Job (4) view is the final step of creating a new job, and provides a preview of the application form and allows to assemble the recruiting team.

The final screen, before publishing the position online, is for reviewing the previously specified criteria and inviting other recruiters for the job process. The latter is connected to the *Team* pattern and also makes use of the *User List*. Finally, an expiry date can be added that communicates how urgent it is to work on the process.

6.1.4: 21 – Candidates in a Job

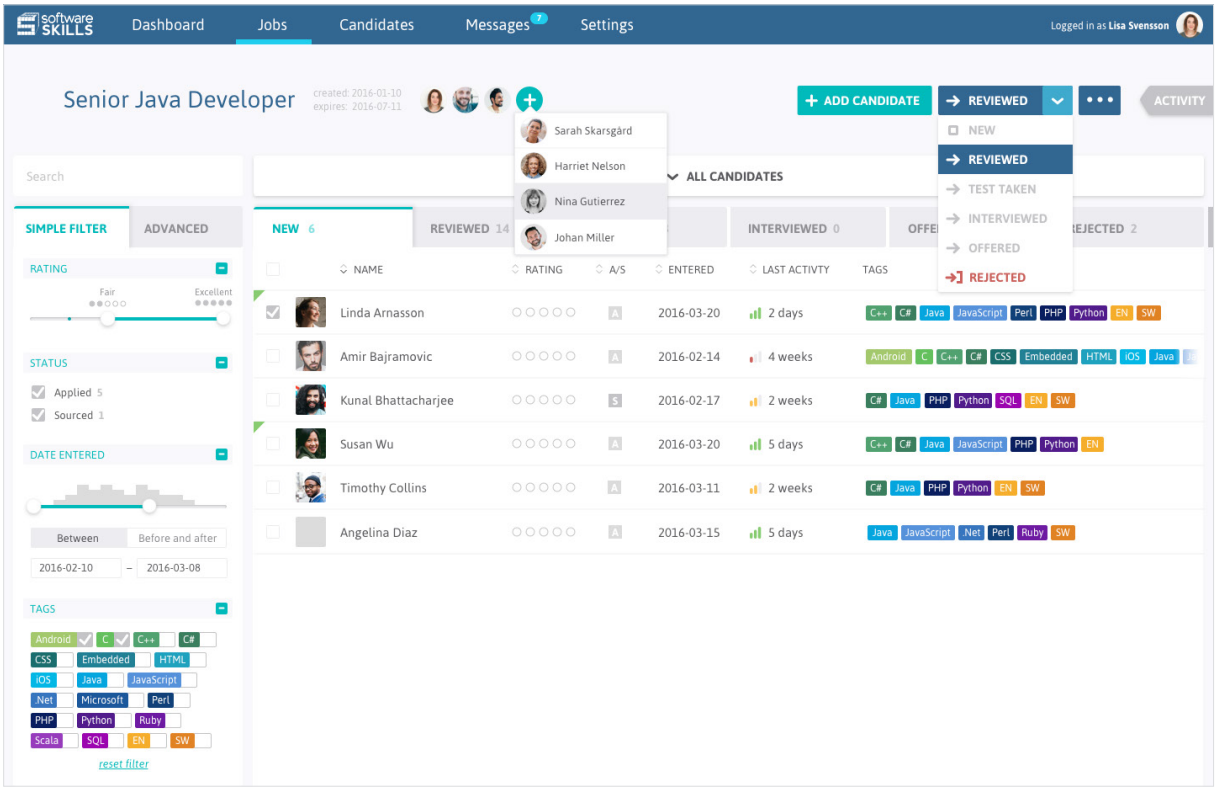


Figure 6.8: Displays the tabbed list view layout of 21 – Candidates in a Job.

The difference between the two views in the *21 – Candidates in a Job* is the overall motivation for using one or the other: One view supports a more emotional or personal connection to the candidates, because the recruiters can manually sort candidates with drag and drop and the information is presented according to what people remember about each other. For instance, the candidate profile picture is bigger, the aggregate rating is shown and icons conveying whether the candidate has been sourced or applied are included. Also, showing the relation of how many cards are stacked in each stage provides intuitive and implicit visual feedback of how the overall process is going. On the other hand, the list view allows more automatic sorting, more quick finding of specific information, which is less personal. The list view works more objectively with parameter-driven actions and access to bulk actions. This difference creates two varying experiences for the users and – depending on both situation or preference – can better cater to the respective needs of that situation.

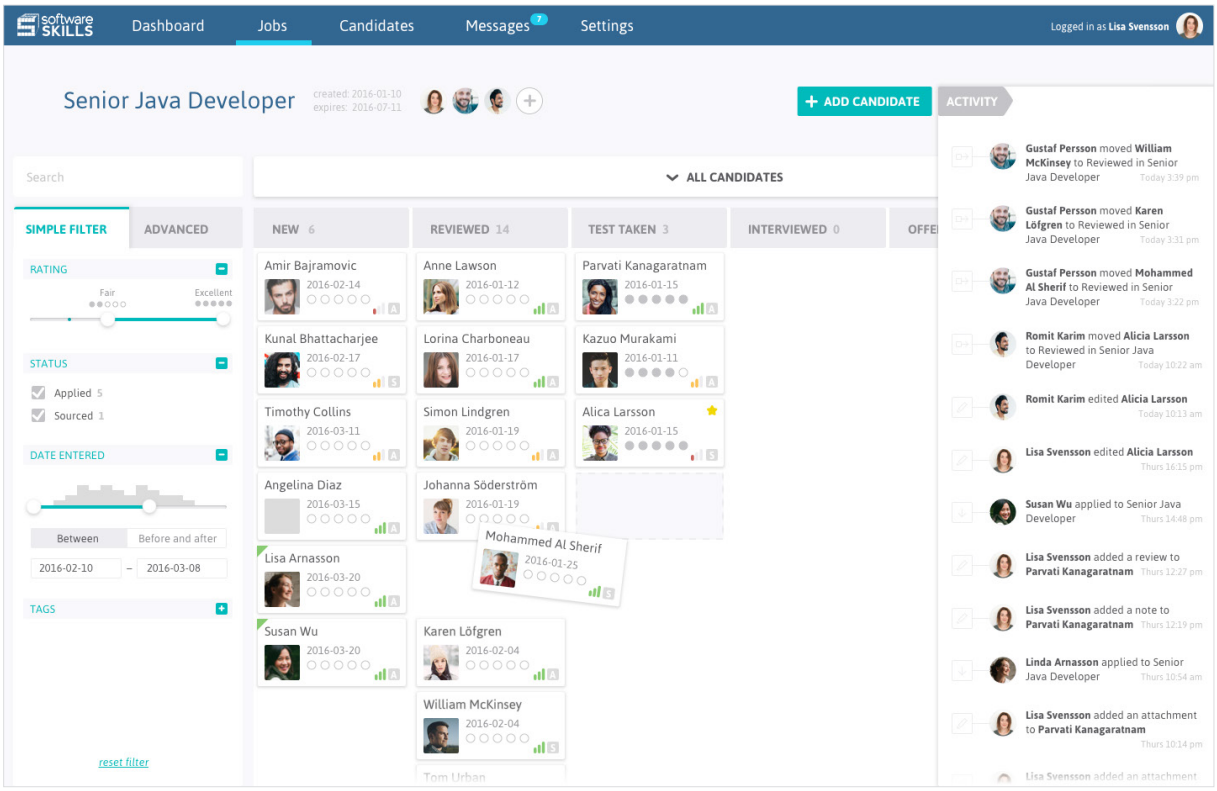


Figure 6.9: The card view layout of 21 – Candidates in a Job.

The candidate view offers an overview to support the recruiters to navigate the content of all applicants of a job in a comprehensive and intuitive manner. The candidates are on display under different tabs for the list view and in columns for the card view depending on what stage they've reached in the hiring process. The idea is that all new applicants end up in the New category automatically. Thereafter recruiters review, conduct interviews, tests and rate the candidates in order to determine whether the candidate suits for the position. As the process continues, the recruiter moves the candidate from the *New* category to either *Reviewed, Test Taken, Interviewed, Offered, Rejected*, or any other section the recruiters have chosen in the job setup phase.

The card view supports the mental model of sorting artifacts, which proved to be very popular in user tests. However, for large job openings with many applicants it is not optimal since it requires a lot of manual work and does not offer the possibility to make bulk actions. Hence, an alternative list view is provided for greater process overview. Instead of having columns with all candidates for that job visible at all time, the candidates are sorted under tabs, which is an idiom that follows users' mental model of how things are normally stored, according to Cooper et al. (2014). Displaying the candidates in a list saves a lot of screen-real-estate and makes it possible to easier sort and compare the different candidates. On top of the list, there is a tab for displaying all candidates in the job at once without the division of different categories. This enables comparisons, searching, and filtering of all candidates in the job at once.

Another handy functionality visible here are tags. Recruiters can set up questions for the application process to evaluate whether a candidate has certain skills that are required for a given job. Tags are added automatically to the application according to how an applicant answers the questions on the corresponding expertise, during the online application. This prevents candidates from being discriminated in a hiring process, due to a recruiter failing to add the correct tags. Being able to quickly filter candidates according to skills speeds up the finding of relevant candidates perceivably and is something the recruiters often do, as we have found in our contextual inquiry. The controls for filtering and advanced search are placed on the left side of the overview, because it corresponds to the western reading direction where information uptake and action initiation takes place in the upper left area. The functionality serves to find certain combinations of skills, experiences, current occupation and other relevant criteria.

6.1.5: 21 – Candidate Card

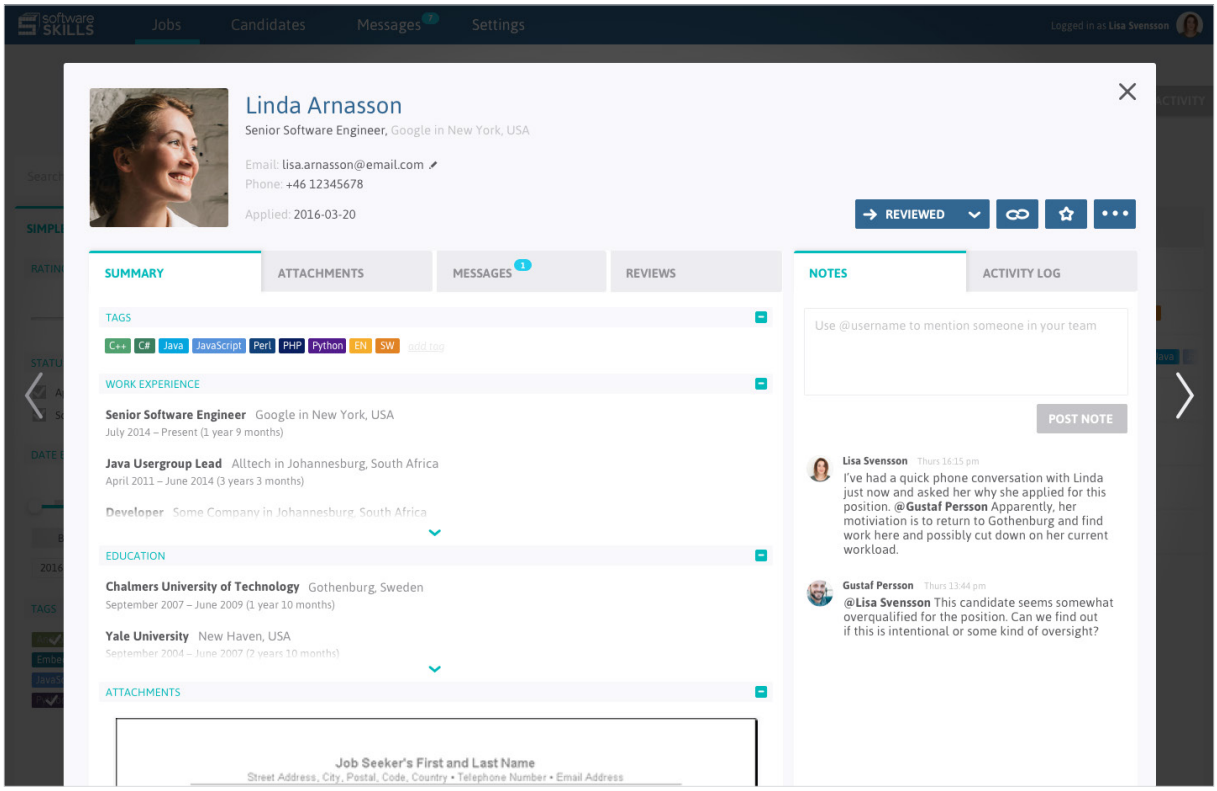


Figure 6.10: Displays 21 – Candidate Card where all information related to a candidate's application is stored.

Subviews like the *21 – Candidate Card* are opened in modal windows to reduce the navigation destinations in the menus. It gives the users an illusion of fewer options to navigate to, thus reducing mental excise. Also, displaying the candidate in a modal view offers the possibility to switch between candidates in a focused manner. The cards contain all information the applicant has submitted, notes (corresponding to the *Shared Annotations* pattern), and reviews from recruiters (corresponding to the *Differentiated Voting* pattern) as well as controls for actions that can be applied to a candidate. There are also arrows on either side, permitting to browse all candidates in a hiring stage without having to exit the detail view.

The information and features are positioned in order of importance and relevance for the recruiters determined through interviews and ethnography. The modal view is divided into three sections. The data displayed in a card can have two different kinds of destinies. This means that after a recruitment process is over, the candidate and some information is still stored within the system. Information not relevant for the future is archived in old versions or removed completely.

The top section contains information that is more static over time, hence always stored with the candidates ID in the database. The bottom-left area contains information that is updated over time as the applicant submit and applies for new positions. The summary tab is always active when opening the card since this information is of most relevance when reviewing a candidate. The tab displays the most relevant piece of information that is parsed from their attached CVs, Cover Letters or LinkedIn. The full information is offered upon desire under the Attachment tab not to overwhelm the user with information.
The communications tab displays all email correspondence with the candidate which helps the team of recruiters to keep up to date. The Reviews tab contains the *Differentiated Voting* pattern thus offers the possibility for the recruiters to rate different aspects of the candidate after an interview. This supports efficient decision making in the recruitment process. Reviews and attachments are kept with the candidates ID but stored in different versions if new updated information is added in another hiring process.

The bottom-right section contains information solely added by or automatically because of recruiters. The notes or *Shared Annotations* pattern is added to support efficient decision making and sharing of knowledge. The notes and the activity log or *Contextual Activity Log* pattern are set to invisible after the process has been archived which corresponds to the *Curated Archived Information* pattern that filters out information irrelevant over time.

6.1.6: 30 – All Candidates

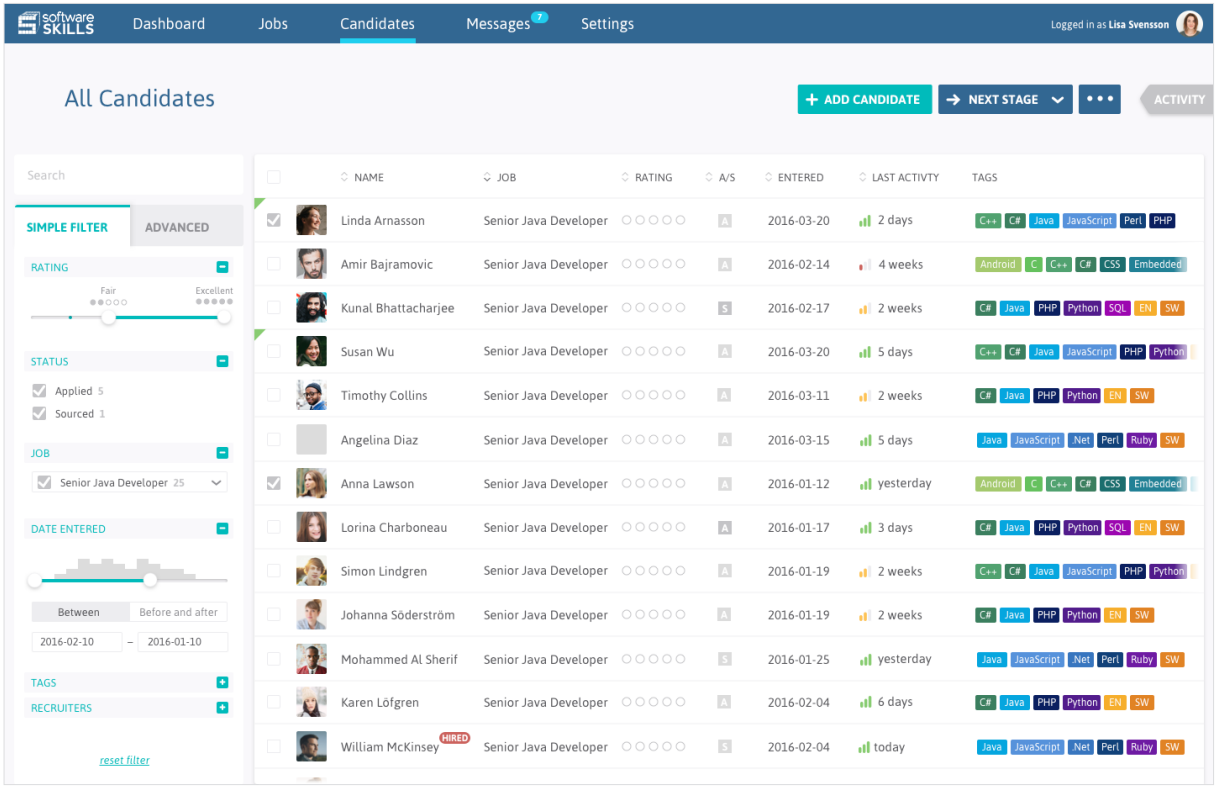


Figure 6.11: Displays 30 – All Candidates that ever been entered into the system.

The *30 – All Candidates* screen is the heart of the application. Here all candidates, active or not, are stored. This database is used by the recruiters to find candidates suitable for upcoming jobs. Often recruiters find candidates that meet a lot of the required criteria, but for varying reasons do not sign a contract with one of the clients. Hence, the candidates are stored and if possible matched to suitable career opportunities at a later point in time. Candidates who get signed are marked with an icon to prevent approaching candidates that already got signed, because doing so could affect the relationship to their clients negatively.

The screen is very similar to the list view of *21 – Candidates in a Job*, with the exception of the exact parameter that are shown in the list, and the corresponding filter options. For instance, the overview displays the current hiring process instead the hiring stage. As aforementioned, candidate data is archived, modified, or removed after an active hiring process is concluded. Therefore, the presented candidate information may differ when accessed through here than instead through a specific job opening.

6.1.7: 60 – User Profile

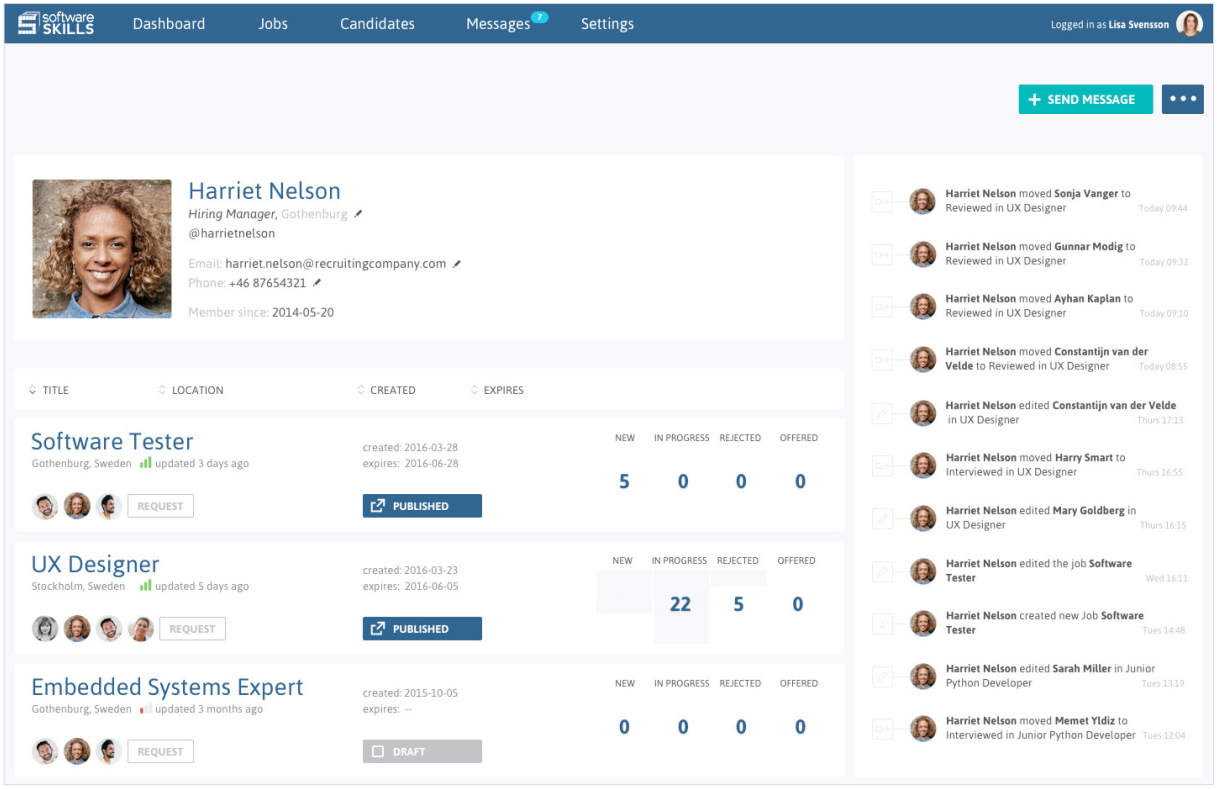


Figure 6.12: Displays 60 – Profile, where a system user's information and activity is shown.

The different users' *60 – Profile* pages are accessible for any user of the system and helps them to understand who is present in the system. The *User Profile* pattern in the results section explains this in further detail. The view displays contact information, the job processes the user is tied to, and the user's action history. The view also offers visitors the possibility to directly compose a message and thereby engage faster. By presenting a user's active job processes recruiters can request access to help out with a process if one of their colleagues is struggling with their workload, for instance. Every user has the option to display their own activities in a log on their public profile page. This makes it easier for other users to coordinate their work when supporting a recruiter's work.

6.2 The ACE Pattern Model

As mentioned in section 2.3 pattern languages, it is of great importance to coherently structure and frame the collection. Since the patterns are an interconnected, holistic concept it can appear confusing and overwhelming for people coming into contact with the pattern language for the first time. To facilitate understanding, a model that categorizes the patterns was developed. It may guide other potential users of patterns to the appropriate type of solution, provides a visual reference, and may also help them to grasp the problem space as a whole.

The model was conceived to triangulate the items of final pattern collection between the three aspects that were most relevant to our selection process: *Awareness, efficiency,* and *collaboration (ACE)*. For example, the *Embedded Messaging* pattern is intended to address the possible lack of communication within an application. It does not automatically offer information about other users' work, and is neither entirely aimed at speeding up the current workflow nor at increasing its efficiency. Instead, the solution provides an asynchronous means of communicating between users. Since this pattern facilitates collaboration more than any of the other patterns, it is positioned the closest to the *collaboration* corner.

A given pattern can address one or several of the three aspects and the positions in the model are only approximations. The placement is strongly influenced by what kind of role the pattern plays in our specific design prototype. Adding a further pattern that corresponds to one of the parameters more extremely than the currently contained ones, would shift all other patterns to a slightly more moderate position. Also, how strongly a pattern is affected by one or several of the three keywords can also change according to how the pattern is implemented in a given system, and how that particular usage changes the system in terms of awareness, collaboration, and efficiency. Yet the fact that the distribution is relatively spread out across the triangle, shows that the collection covers all of the three main aspects of the desired attributes to a certain degree.

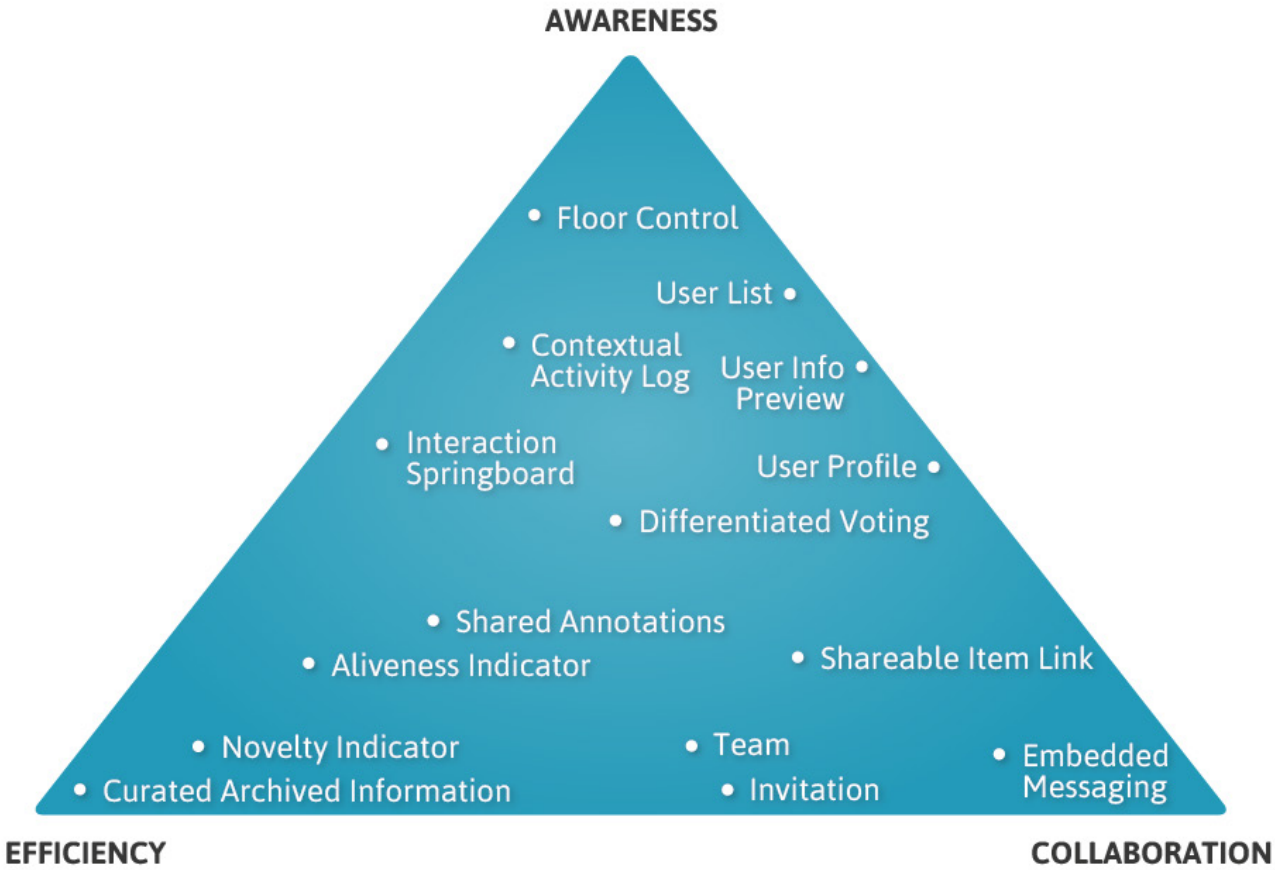


Figure 6.13: Triangulation model of the ACE attributes, containing all patterns for applicant tracking systems, according to the conceptual purpose.

6.3 Patterns for ACE in Applicant Tracking Systems

As described in section 5.6.1 Heuristic Review, the final result to come out of our process was the emergence of two groups among the patterns. They connote the respective functional purposes of the patterns – *Information Sharing* and *User Management* – and because some of them satisfy both purposes, a venn diagram is a suitable way to visualize the groups (see figure 6.14). The pattern collection is subsequently presented according to *Information Sharing* first, then the overlap of both categories, and finally the patterns that belong exclusively into the *User Management* circle.

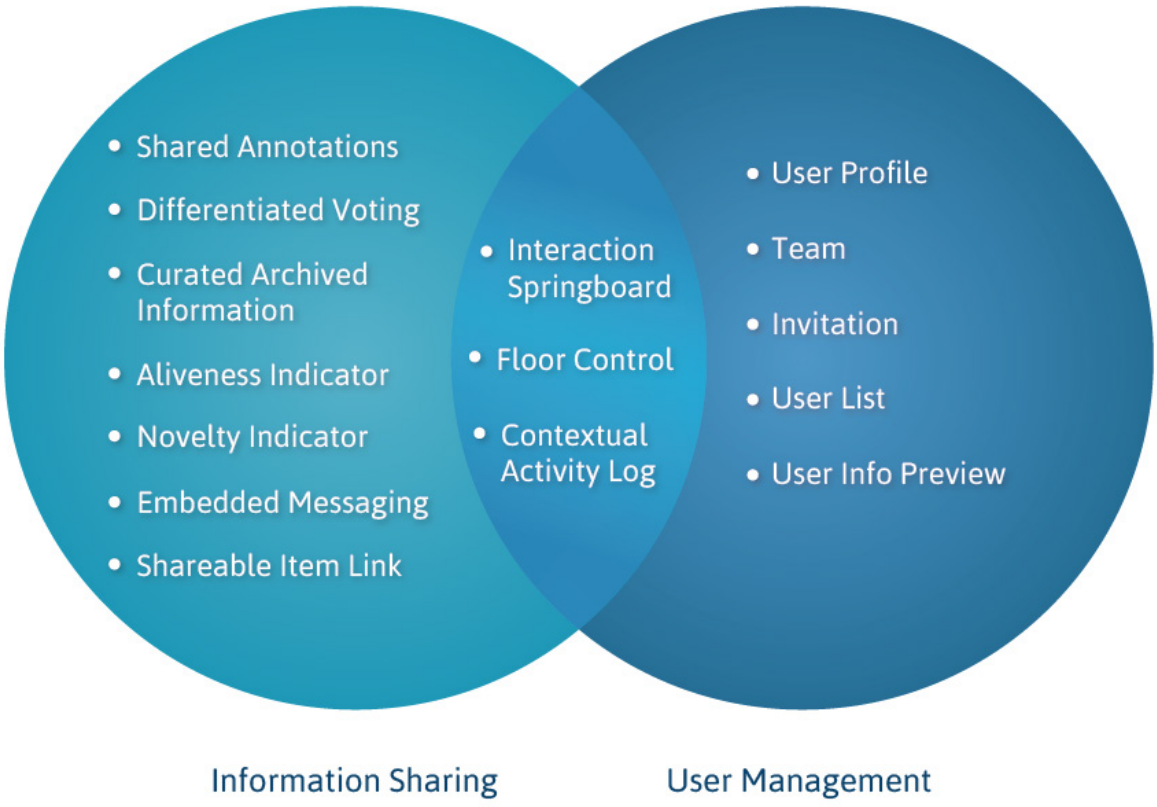


Figure 6.14: The two pattern groups representing the functional purpose of the patterns

6.3.1 User Profile

Problem

Users may not know each other personally and it is difficult to understand each other's roles in the system.

Context

Apply this pattern when the community grows and changes, and not all users may have personally met each other, especially when users are located in different geographical places.

Solution

Provide a personal profile page for all users, where they are encouraged to upload a picture, write a short descriptive text about themselves, and display their contact information. Also, their tasks and activities are shown so other users can get a sense of what the person works on and what their area of expertise is.

Consequences

- This creates a better understanding of who is part of the community/user group, and what everyone is doing in the system
- Creates a sense of familiarity of other users

Danger Spots

- If one or more people refuse to add information about themselves it could start a negative trend in the system, since users might feel less inclined to provide more information about themselves than others
- Provide wrong or false information if the system requires too much or too private information
- Information can become outdated, therefore it can be a good idea to ask users to update their information on a regular basis

Why

The solution raises a sense of team spirit and familiarity among (unacquainted) users.

Evolvement

The collaborativeness of an application is inextricably based on accommodating users. For said users to manage their account details and representation in the system, a view often called user profile or profile page is provided. Since this is such a common occurrence,

this feature had already been part of the first version requirements list, very early in our process. However, it turned out to be one of the last screens to be wireframed and designed, because it is not a central feature of the core functionality of the Honeycomb ATS. The makeup of the user profile as it is represented in Honeycomb is therefore heavily influenced by the other parts in the system and is optimized for awareness. Therefore, it consists of a summary of the user's activity in the system, showing the personal activity log and the current tasks, apart from the standard user information. The equivalent we matched in Schümmer and Lukosch's book is called "Virtual Me" (2007, p. 97) and is concerned with the user profile as a tool to construct and cultivate one's online identity in a community. While interesting for a large online community, most issues they raise here are not directly related to a productivity tool used in the internal company-context, where all the users have similar goals for representing themselves in a professional manner.

Examples

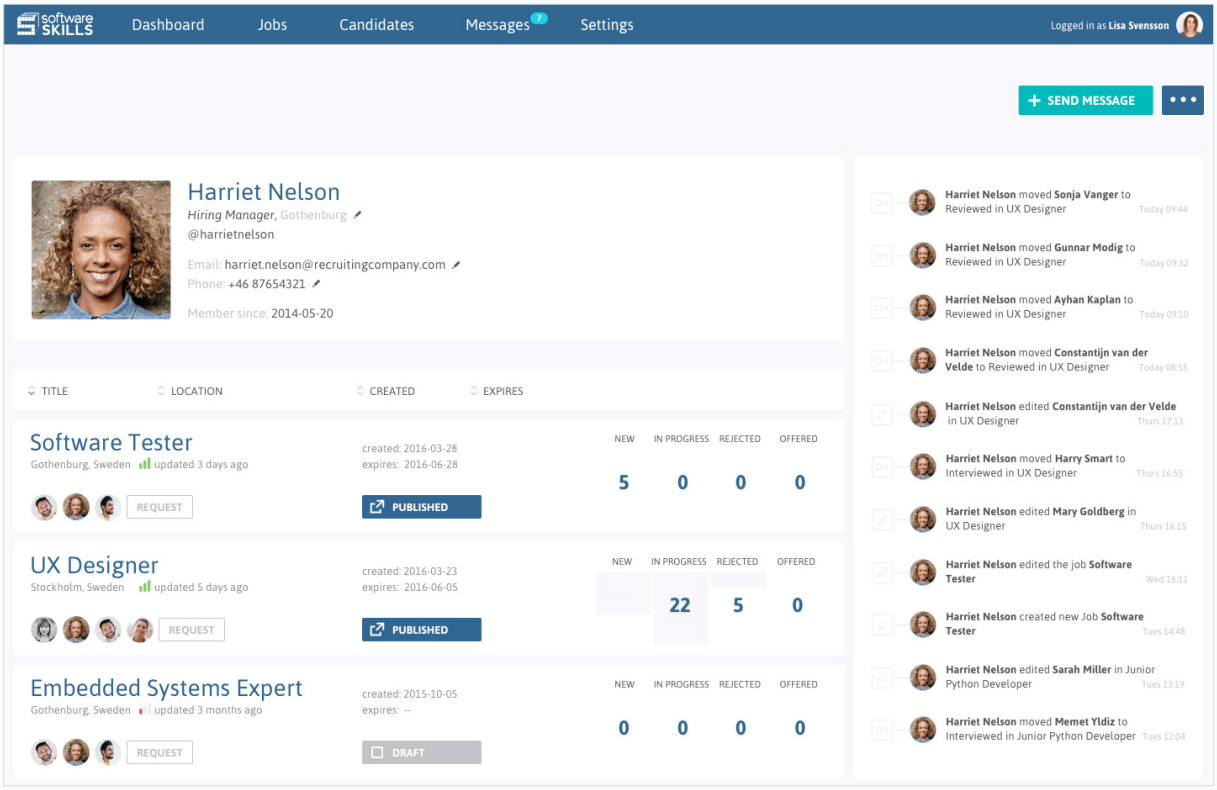


Figure 6.15: User Profile with all information related to a user.

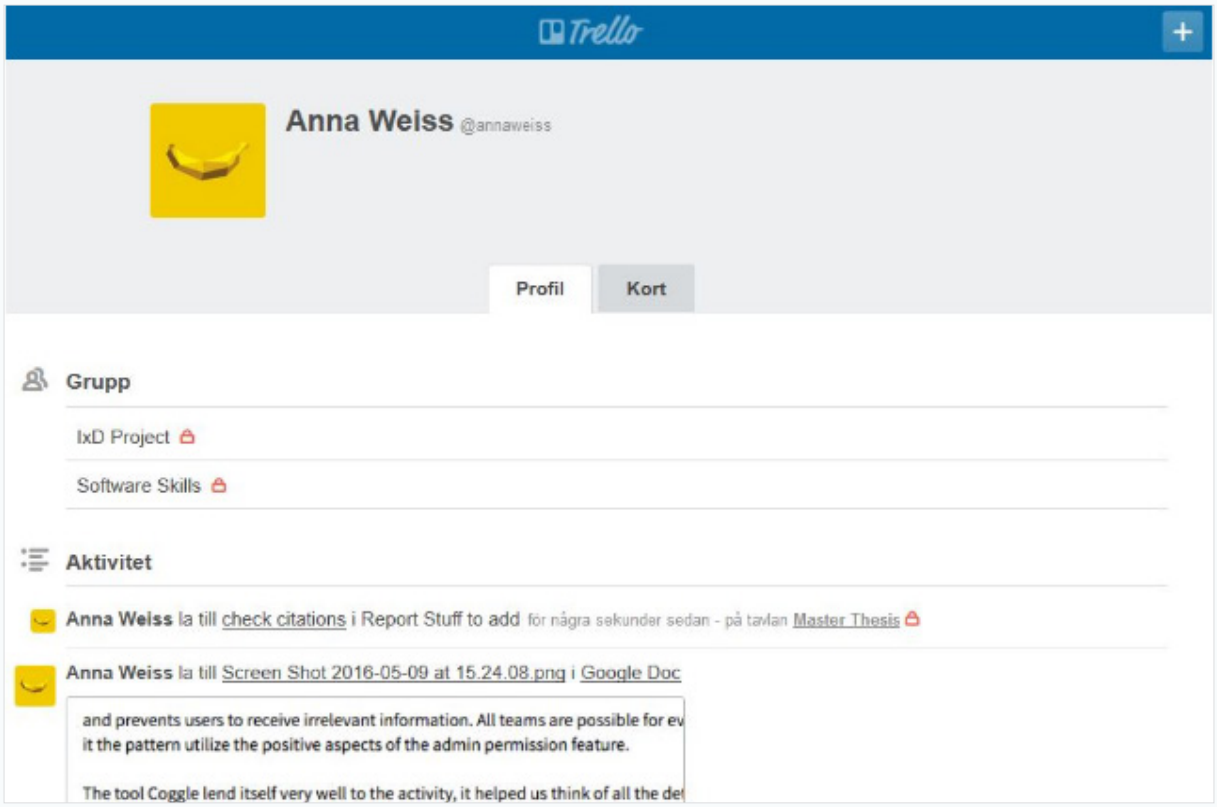


Figure 6.16: Trello user profile.

Related Patterns

- User List: The users in the *User List* are represented through information they have uploaded to their *User Profile*.
- User Info Preview: *User Info Preview* links to the respective *User Profile*.

6.3.2 User List

Problem

Users do not know who is available in the system as a potential interaction partner.

Context

Apply this pattern when many users are members of a system.

Solution

Provide the list of users present in a system.

Consequences

- Users get a feeling of working with others in a group
- It is more convenient to initiate interactions with others or invite others to teams

Danger Spots

- When there are many users in the system, the list becomes overwhelming

Why

Conveying who users are sharing information within a certain space with is essential for reassurance and creates a safe and efficient working environment.

Evolvement

This functionality is a sub-pattern to *Teams*, as it is important to provide information about possible users to invite. It is a feature that emerged after the first design iteration. Upon reviewing the prototype for patterns not included in the collection yet, it was realized that it was closely related to awareness and collaboration, because not having this feature would make the interaction between users less effective. Subsequently, Schümmer and Lukosch's collection verified this feature as a pattern (2007, p.319). They describe it in a synchronous context, where users are invited to real-time collaborative sessions, with the help of a user list that shows who is currently online and available. Therefore, it does not overlap exactly with *User List*, however, many of the advised danger spots still apply. The name "User List" was adopted, which hitherto had been referred to as "User Overview". In conclusion, *User List* is a pattern, where the conceivement was mainly informed by the process of designing the Honeycomb ATS.

Examples

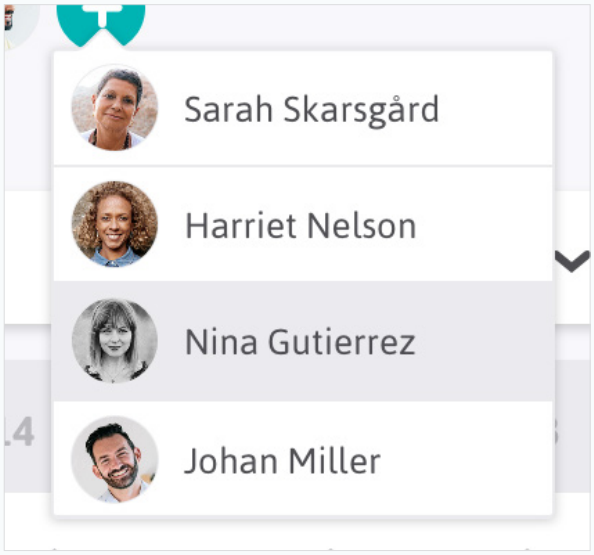


Figure 6.17: Example of a User List.

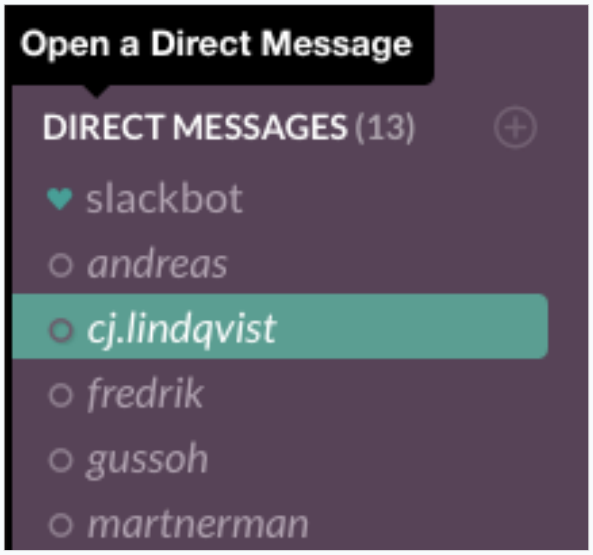


Figure 6.18: Example of a User List in the chat application Slack.

Related Patterns

- Embedded Chat: *Embedded Chat* should contain a *User List* to make it convenient to initiate interaction.
- Invitation: The *User List* is useful to select a user to initiate an *Invitation*.
- Teams: The *User List* is useful to choose members for a Team.
- User Profile: The users in the *User List* are represented through information they have uploaded to their *User Profile*.
- User Info Preview: *User Info Preview* augments the *User List* with information.

6.3.3 Team

Problem

There are many users and it is difficult to know who is working together.

Context

Apply this pattern when there are many users within the system working on different tasks.

Solution

Allow for the grouping of users and be able to handle formed groups the same way as an individual user. Various teams can receive access to different parts of the system to coordinate and divide areas of responsibility.

Consequences

- Saving time since users know the areas that apply for their work, instead of always accessing all areas of the system.

Danger Spots

- Can create a sense of alienation for users not in a team
- If this feature is misused it can lead to a hierarchical system structure
- Decreases awareness, however for users own benefit, as they are only informed of what is happening in the areas that pertain to them.

Why

Stipulating groups within a system makes it more efficient when coordinating work since it is not necessary to do so on an individual level.

Evolvement

The first initiation of this pattern occurred through discussions with the main stakeholder, Henrik Enström, who drew from his experiences as a manager at large software companies, in which capacity he also ran the recruiting efforts of the department, and resulting issues related to visibility of certain content and hierarchical structure. Instead of having rigid teams, the goal is to have a fluid system, permitting to easily add users and teams on a case-by-case basis, and this has been a fixed component of the Honeycomb ATS since the first requirements listing. Also, Asana the project management tool or Trello, as described in the collaborative web applications benchmark, have a similar solutions to that. Schümmer and Lukosch title a comparable pattern Groups, where they also specify

that this means that a group of users is treated the same way as an individual user (2007, p.194). This pattern requires the subtasks "Add users to a group – Remove users from a group – Create groups – Name groups – Remove groups" (2007, p. 195) in order to be complete.

Examples

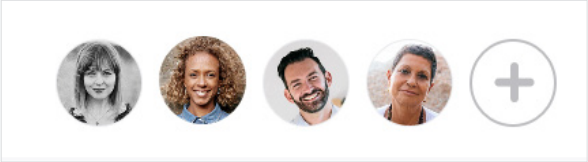


Figure 6.19: Current members of a team, with function to add new members.

Figure 6.20: Trello has different categories of team members and pending invitations.

Related Patterns

- Embedded Chat: *Embedded Chat* could make use of the *Teams* so it is more convenient to initiate group conversations.
- User List: The *User List* is useful to choose members for a *Team*.
- Invitation: *Invitation* is recommended to add users to a *Team*.

6.3.4 User Info Preview

Problem

With many different users present in the system, showing solely the avatars makes it difficult to recall people's user names. Also, it is inconvenient to initiate interaction with fellow users if one has to go to another place in the system first.

Context

Apply this pattern when there are many instances where users are shown to be involved, and communication among users is beneficial for the workflow.

Solution

Having a user info flyout that appears when a fellow user's avatar is clicked. This flyout shows the name, the position, and the user name, and offers the possibility to contact the user.

Consequences

- Makes it easy to keep track of fellow users.
- Makes it possible to provide little information on constant display to save screen real-estate.

Danger Spots

- None identified.

Why

To quickly remind or inform users of whom they are sharing the system with, and encourage communication.

Evolvement

This type of interactive flyout, revealing more information when hovered over a user's representation in the interface, like the user name or avatar, is a common feature in collaborative web apps and social media platforms, as the benchmark analysis revealed. Its adoption into the collection was closely tied to the Profile and *User List* patterns, because without those, there would be no place to instantiate the *User Info Preview*. Schümmer and Lukosch stress in their pattern collection that the flyout should be interactive – naming the pattern "Interactive User Info" (2007, p.337) – highlighting the feature of providing direct means of engaging with that user.

Examples

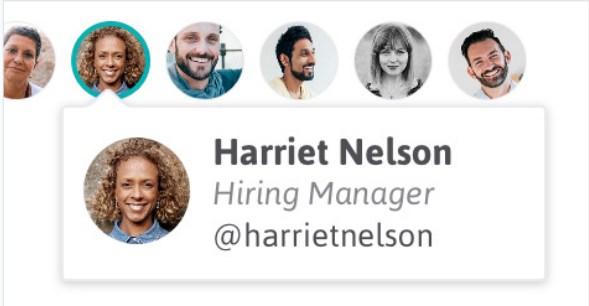


Figure 6.21: Example of an active User Info Preview, in the context of a user list.

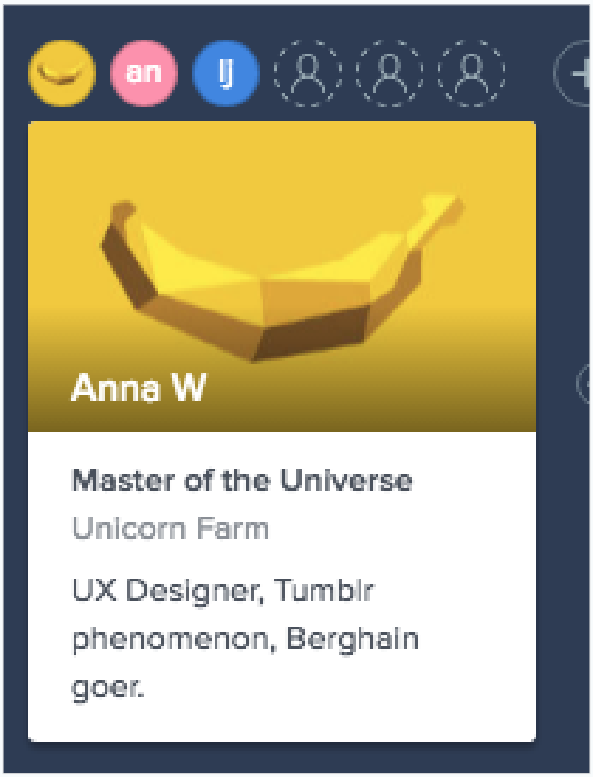


Figure 6.22: Example of User Info Preview in the project management web application Asana.

Related Patterns

- Activity Log: Both the *User Info Preview* and *Contextual Activity Log* can contain the *User Info Preview*, to make it easier to initiate communication with the users connected to a certain log item.
- User List: *User Info Preview* augments the *User List* with information.
- User Profile: *User Info Preview* links to the respective *User Profile.*

6.3.5 Invitation

Problem

Potential users or team members need to be notified that their participation is requested in the system.

Context

Apply this pattern when a team needs to be assembled politely and efficiently in an asynchronous setting.

Solution

Through an invitation, users can be notified to join a team. This invitation can then be accepted or rejected.

Consequences

- Users are in control of which teams to be a part of

Danger Spots

- Invitations may be ignored or missed
- Not having a team leader and allowing everyone to invite users can create uncontrolled and less coordinated teams
- Users may be hesitant to reject an invitation, because they worry it may be regarded as impolite

Why

An invitation rather than a direct assignment provides the individual users with more control over what they need to be directly involved in (and get notifications for), so they don't fall victim to overzealous colleagues wanting them to be a part of everything.

Evolvement

Invitation is a sub-pattern to *Teams*, as its existence is necessary for finding potential team members and request participation in a polite manner. It is a feature that was present in the prototype after the first design iteration. Upon reviewing the prototype for patterns not included in the collection yet, we realized that it was closely related to awareness and collaboration, because not having this feature would make the interaction between users less polite. Subsequently, we also consulted Schümmer and Lukosch's collection to verify this feature as a pattern. They describe it in a synchronous context, where users are invited to real-time collaborative sessions, with the help of a *user list* that shows who is currently

online and available (2007, p.255). Therefore, it does not overlap exactly with this corresponding pattern, however, many of the advised danger spots still apply. To sum up, *Invitation* is a pattern, which was largely inspired by the process of designing the Honeycomb ATS.

Examples

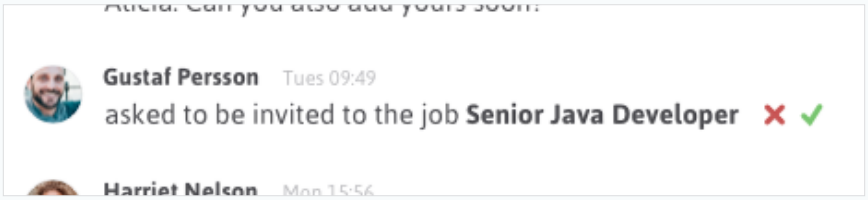


Figure 6.23: A pending invitation in Honeycomb.

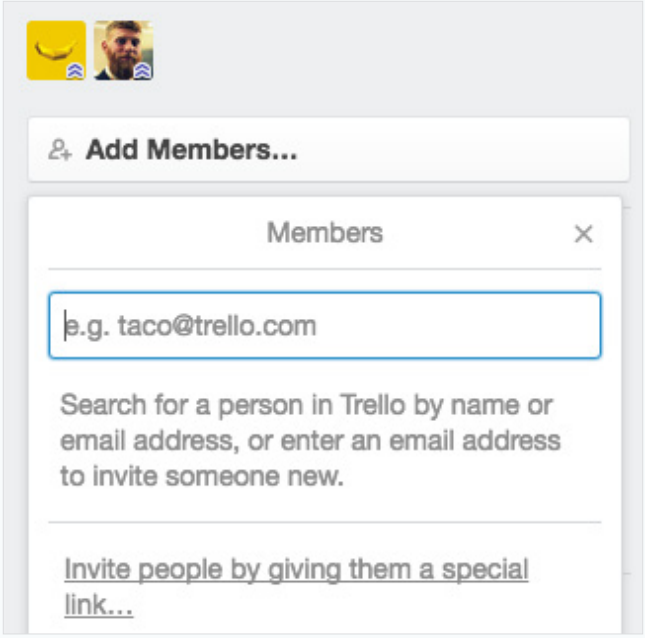


Figure 6.24: Inviting already registered members to a Trello board, inviting others to sign up.

Related Patterns

- User List: The *User List* is useful to select a user to initiate an *Invitation*.
- Sharable Item Link: Both the *Invitation* and *Sharable Item Link* are initiations for collaboration, the only difference being that the *Sharable Item Link* extends this invitation to external contributors.
- Team: *Invitations* are recommended to add users to a *Team*.

6.3.6 Shared Annotations

Problem

Users have varying knowledge and opinions about artifacts in the system. It is difficult to collect these different insights and share them effectively.

Context

Apply this pattern when opinions on a specific artifact in the system need to be shared among users.

Solution

Collecting all thoughts and information about the artifact and attaching them clearly to it and making it accessible for future reference by all privy users. When it comes to answering questions and resolving issues connected to these artifacts, it is possible to address a specific user by adding a direct mention in the comment.

Consequences

- Easy to overview both the status of the artifact and the collective opinion about the information
- Facilitates coordinating future actions among users

Danger Spots

- Can turn into chat-like conversation (with irrelevant content), which makes it more difficult to find relevant information
- If notes are public and stored, the discussions are likely to be more censored than verbal conversations
- If the annotations can be edited by their authors, it is possible that information is taken away that is referred to in a subsequent comment. This may cause the annotations to not make sense when revisited later

Why

Helps with informed decision-making as the collective thoughts from the involved users are attached to the artifact.

Evolvement

Adding different kinds of information to a candidate card is a common practice within the previously existing Trello solution at Software Skills. This included attachments and images, but also comments that recruiters added throughout the hiring process. Keeping

all the relevant information in one place was found to be a logical method, and inspired a specific note taking feature, to encourage this practice further. ATSs like Jobylon and Workable also include this functionality. While initially called "Notes", the more descriptive name "Shared Annotations" by Schümmer and Lukosch was adopted later (2007, p. 291).

Examples

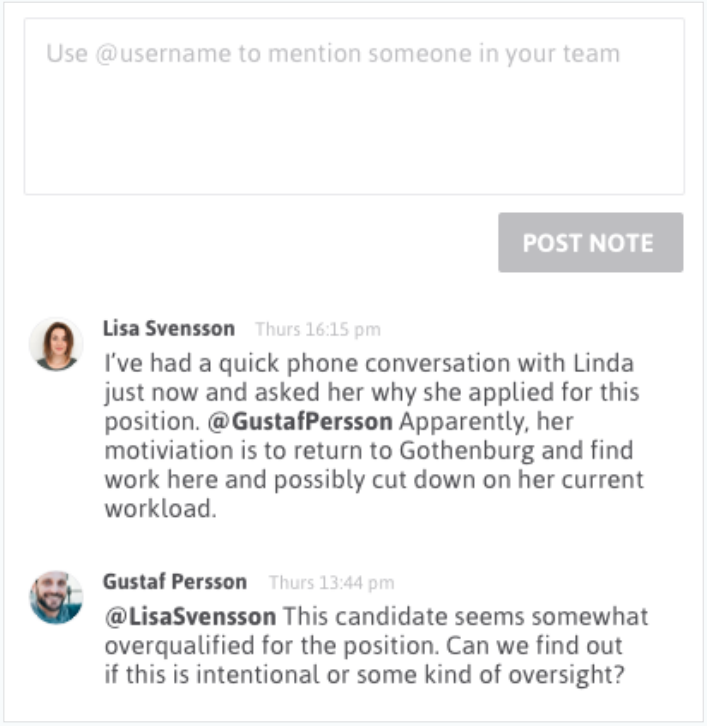


Figure 6.25: Shared Annotations in Honeycomb, with the possibility to tag other users (mentions feature).

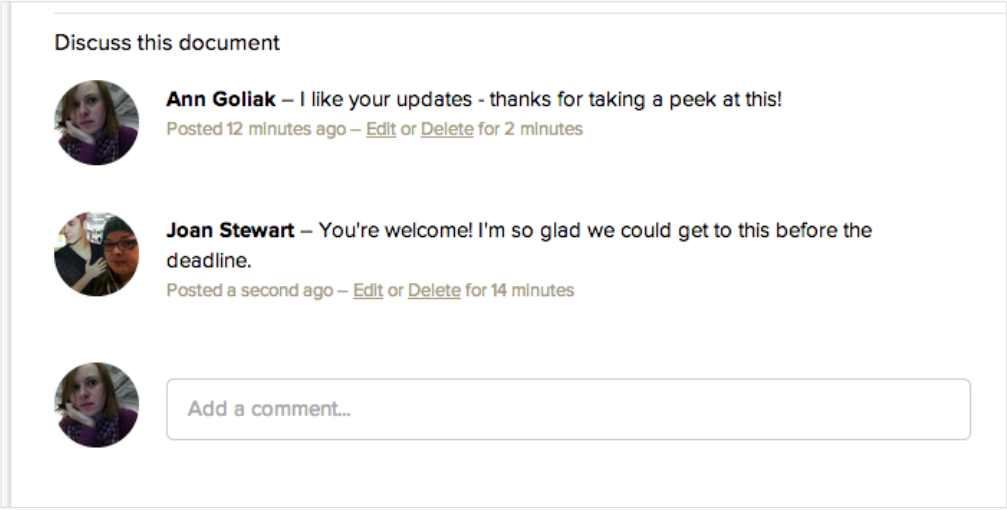


Figure 6.26: Basecamp allows creating notes attached directly to artifacts.[1]

[1] https://basecamp.com/help/assets/images/bcx/projects/textdoc-comment.png (accessed: 17.05.2016)

Related Patterns

- Embedded Messaging: Both *Shared Annotations* and *Embedded Messaging* allow to direct messages to specific users.
- Differentiated Voting: Both *Shared Annotations* and *Differentiated Voting* offer users the possibility to attach messages and opinions to a certain artifact visible to all privy users.
- Curated Archived Information: Curation removes the *Shared Annotations*.

6.3.7 Differentiated Voting

Problem

Users need a way to find a common agreement regarding a problem.

Context

Apply this pattern when a collective decision needs to be found and the process should be unbiased for each partaking individual. Good for high frequency of decision-making in non-crucial contexts.

Solution

Therefore, two aspects should be addressed: Firstly, the individual user needs to be able to input her decision without being influenced by the rest of the group. Secondly, all the submitted ratings/votes need to be aggregated by the system so that it is easily visible what the general consensus is. It is also recommended that the voting options are very clear, and for example scales have a clear labeling so that everyone has an equal understanding. Furthermore, all the available options should be the same for one instance of voting, so that the input data remains the same throughout one voting process.

Consequences

- Provides indication of what to do next
- Eliminates need for extensive discussions that lack expedience

Danger Spots

- Users can be more extreme or less thoughtful in their voting behavior if it is anonymous

Why

Everyone has their personal opinions, which need to be collected fairly to lead to a common consensus. It is also important to be able to trace votes to understand reasonings behind opinions, so that an informed decision can be made and appropriate follow-up actions can be taken.

Evolvement

Several of the benchmarked ATSs include options to rate candidates. Jobylon, for example, has a five star system that one may recognize from movie reviews or the iTunes music library. The problem with this is that everyone has a different concept of what, for example, a four-star candidate looks like. Workable tackles this issue by reducing the

number of options, and making clearer what each one means. Recruiters can evaluate a candidate by adding a thumbs down, thumbs up, or a star badge, and the number of votes on each is displayed. However, the arbitrarity of a simple rating system was discovered during the first *SME interview* when the recruiter was explaining the job interview note-taking template. The sheet included boxes to provide ratings from numbers zero to ten for various categories, such as personal drive, or work experience. The value of quantifying skills and personal traits was recognized and subsequently adjusted. Through many iterations, and researching different methods of collecting data, like the Likert scale, the exploration arrived at the *Differentiated Voting* solution. It connotes that each contributor's vote is counted towards an aggregate rating, which is displayed on the lists of candidates. The individual evaluation is further split up into predefined categories, whose scales are explicitly labeled. The main purpose to apply this pattern however is to support the team to make a decision, so the name evolved from *rating* to *voting*, which is also what Schümmer and Lukosch term it (2007, p.208), although without the emphasis on differentiation.

Examples

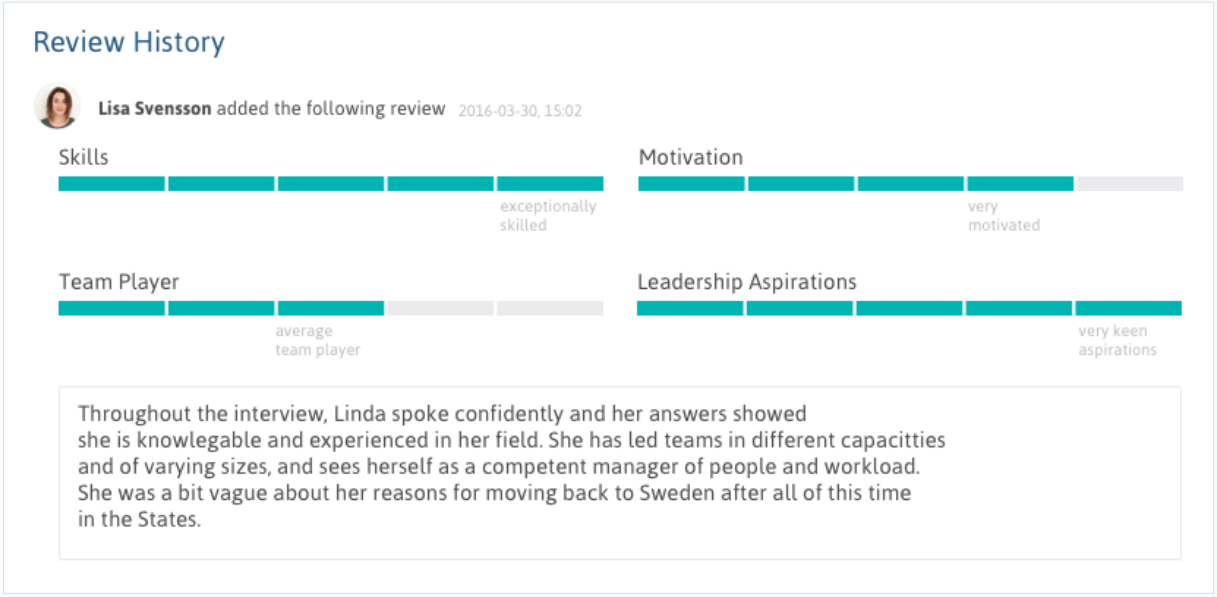


Figure 6.27: Example of a recruiter submitting her review of a candidate.

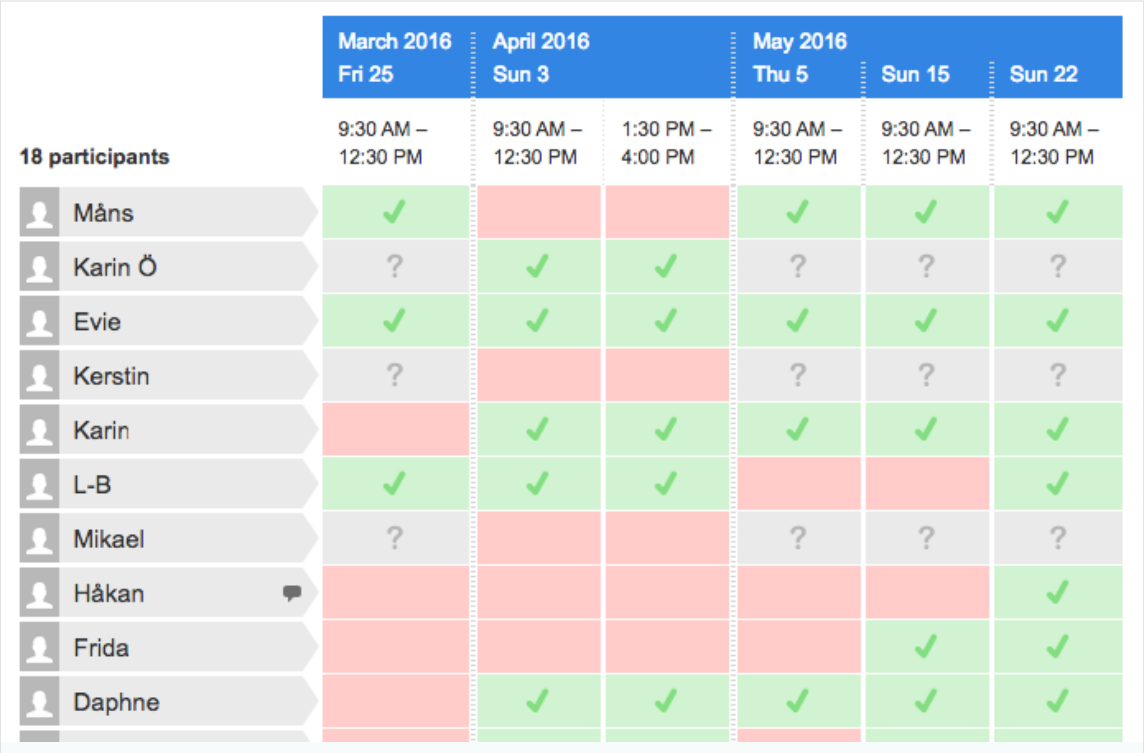


Figure 6.28: Doodle, a system for voting and agreeing on dates suitable for everyone involved.

Related Patterns

- Shared Annotations: Both *Differentiated Voting* and *Shared Annotations* helps to relay information and reach consensus, and is attached to a specific artifact, and is visible to all privy users.

6.3.8 Embedded Messaging

Problem

Lack of appropriate communication channel, because of remote and/or asynchronous work.

Context

Apply this pattern when work related discussions among users are needed.

Solution

An embedded communication system functioning as a chat, but designed as a email service. This means that long threaded conversations are supported and the design accommodates the composition of longer messages. Group conversations are supported as well as communication between different kinds of users i.e. recruiters and applicants. The chat is service is both asynchronous and synchronous for internal users (recruiters) and asynchronous for external users (applicants) since infrequent users of the system get email notifications to their regular email account when messages are received within the application.

Consequences

- It allows for users to work solely with the application and there is no need for third party email solutions
- Supports both asynchronous and synchronous remote work
- Its immediacy helps the synchronous workflow, especially when issues need to be resolved quickly
- Supports group coordination

Danger Spots

- Can be obtrusive for the workflow
- This solution requires extra steps for external users (i.e. applicants) as they have to use their private email as notification tool for the *Embedded Messaging* system and then log in to the system to respond. It is therefore recommended to enable responding to messages through one's email directly.
- In group conversations, it may be difficult to understand and backtrack sub-conversations, because the sorting is strictly chronological and no threading happens.
- Not providing time stamps on sent messages can confuse users of when a message is relevant

Why

This pattern is complementary to all other communication and awareness patterns. Since it is impossible to cover every possible scenario, the *Embedded Messaging* pattern solves this by offering a general communication channel.

Evolvement

The interviews revealed that the recruiters do a lot of communication with clients and prospective candidates via email, but that information often stays there and needs to be painstakingly gathered from different places when needed at a later point. Schümmer and Lukosch proved to be influential for the pattern formulation and design of this issue, as their explications of consequences and danger spots were highly applicable to the current scenario (2007, p.271).

Examples

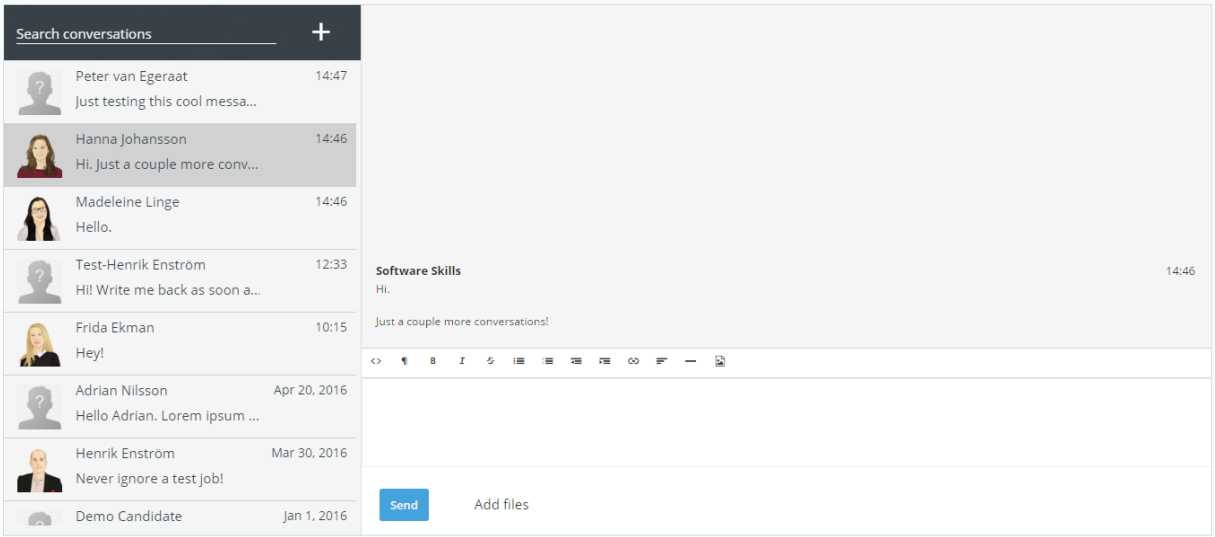


Figure 6.29: The Embedded Messaging in the system, with different conversations shown in the column on the left, and the current conversation in the main area.

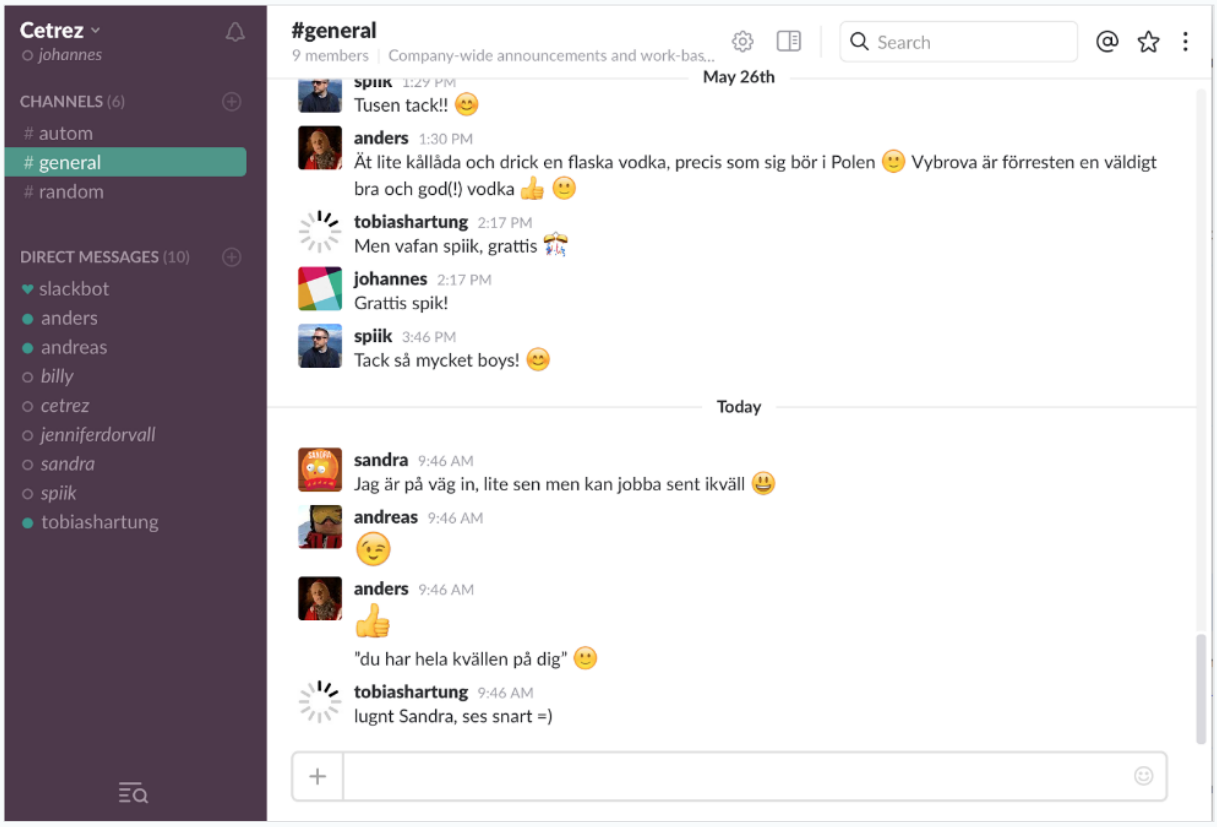


Figure 6.30: The chat system, the core feature of the team communication app Slack.

Related Patterns

- Embedded Messaging: Both *Shared Annotations* and *Embedded Messaging* allow to direct messages to specific users.
- User List: *Embedded Chat* should contain a *User List* to make it convenient to initiate interaction.
- Teams: *Embedded Chat* could make use of the *Teams* so it is more convenient to initiate group conversations.

6.3.9 Interaction Springboard

Problem

Facing a multitude of options and having to decide which is the most important to address.

Context

Apply this pattern when a user is given agency to decide for herself what is important to do within the system, without having to verbally coordinate with others.

Solution

Showcasing the current status of the collective progress and providing hints where to start or continue with work. Providing a hierarchical presentation of the most common actions in the same place. Also, a notification or update feature can brief of new developments that in turn can inform actions.

Consequences

- More efficient work, since upon opening the web application users are informed of what has been happening in their absence
- Provides the users with ample feedback of their progress
- The users receive the latest news automatically without having to use several services to get up to date

Danger Spots

- As this pattern provides many starting points and updates, it does not facilitate the feeling of closure
- If the information is not carefully presented and curated, the user will be overwhelmed with information and it will take a long time to process and decide

Why

It is a useful tool for transparency within an organization and gives individuals ample responsibility to work freely without needing constant supervision.

Evolvement

The justification for including this pattern is rooted in several places. It came up first during talks with Software Skills, who considered this feature a chance to improve their sales as such a screen could show performance indicators. The second hint was to find that many of the ATSs sported such a dashboard: Greenhorn, Teamtailor, or Jobvite, to name a few. A few weeks later, when brainstorming to mind map awareness, it was realized that

many features corresponding to awareness would fit in seamlessly in the starting view of the application, which could be integrated with the hypothetical dashboard. Inspiration was drawn from Schümmer and Lukosch's pattern, titled Interaction Directory, where they suggest to "list potential interaction contexts" to help users engage with the system's capabilities (2007, p. 248). This is a beautiful example of how the pattern evolved and was validated by all of the following factors: business goals, state-of-the-art industry examples, methodological results, and similar pattern collections, as well as making sense from an interaction design perspective.

Examples

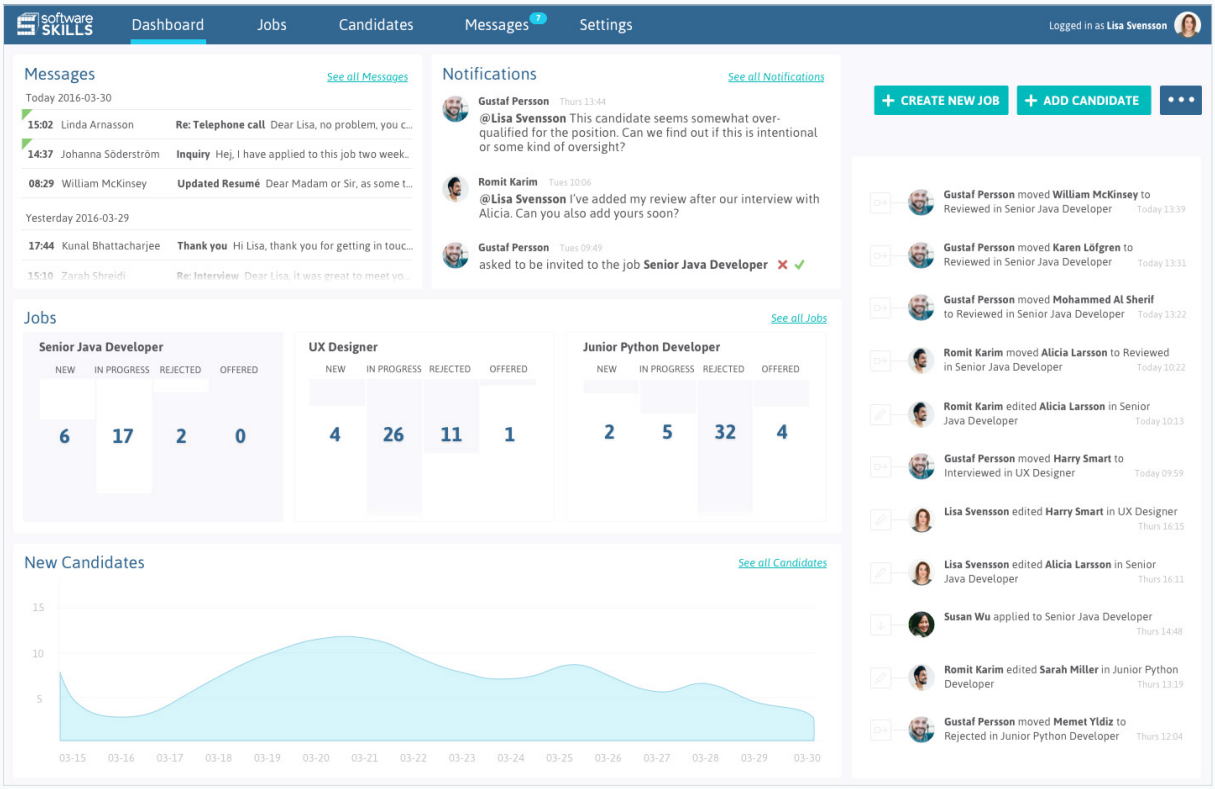


Figure 6.31: Interaction Springboard on the first screen of the system with numerous updates and suggested actions.

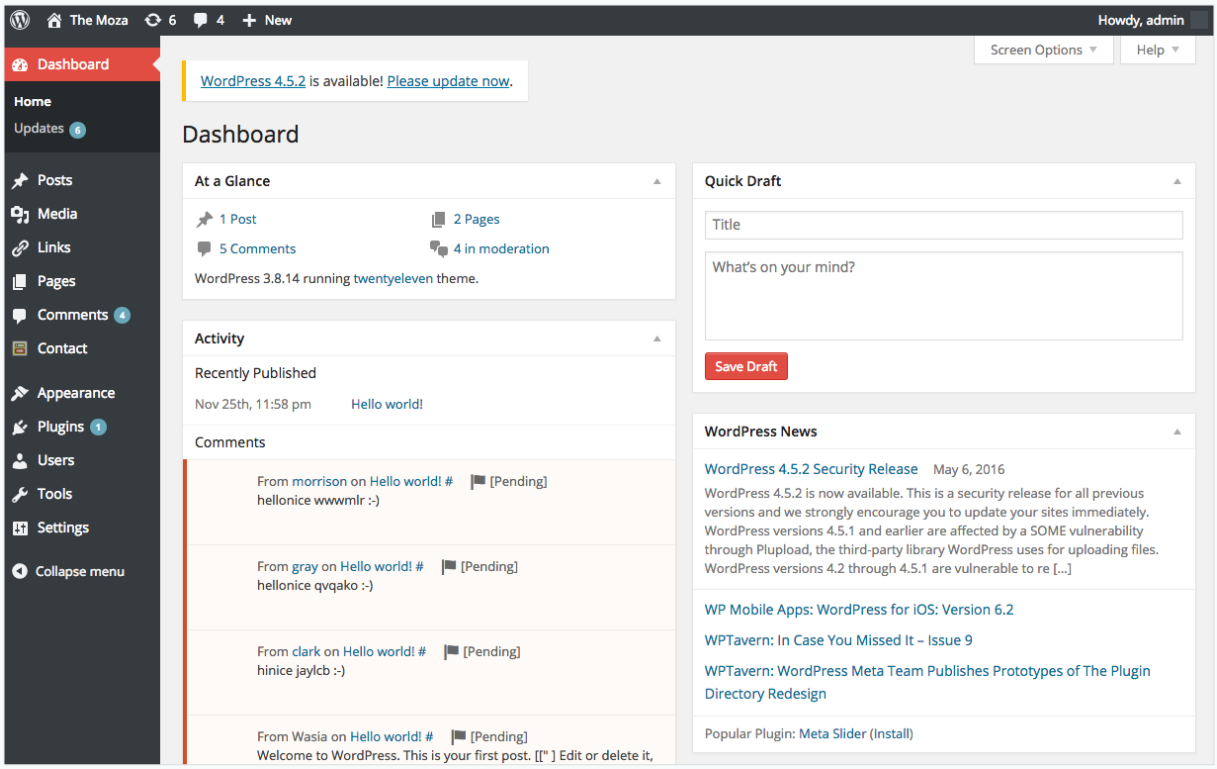


Figure 6.32: The CMS system Wordpress offers a customizable dashboard with news and quick access to tools.

Related Patterns

- Novelty Indicator: Both the *Interaction Springboard* and *Novelty Indicator* present the current state of the system and thus support subsequent action planning
- Aliveness indicator: Both the *Interaction Springboard* and *Aliveness Indicator* present the current state of the system and thus support subsequent action planning

6.3.10 Contextual Activity Log

Problem

The lack of understanding of fellow users' activities on shared artifacts in the system.

Context

Apply this pattern when information on the activity of other system users is crucial for one's own workflow.

Solution

Therefore, information of what happened where, when and in what order, and who executed the action should be shown. To limit information overload, the activity log is kept contextual, meaning that only the activity linked to the current view is shown in the log. It is recommended that the solution allows to access the mentioned part of the system (people, places) directly through links.

Consequences

- It allows for a more open system, because it facilitates reversing actions if need be, i.e. eliminates the need for complicated permission structure
- Can create a sense of purpose and team spirit, because it is visible that work progress is ongoing

Danger Spots

- Can create information overload if all items are logged in great detail
- Can create a sense of surveillance
- If the activity log updates dynamically in real-time, the information becomes a distraction. It is therefore recommended to only update the information upon page refresh

Why

Everyone makes mistakes or forgets both their own and others' activities. The activity log is a useful tool to support one's memory, share awareness about other's progress, and facilitates the reversal of mistakes since it is easy to review previously taken actions.

Evolvement

Activity logs are a common feature in collaborative web applications. Dropbox, Google Drive, Figma, Coggle, and many others all incorporate this effective method to show changes in a system and access the mentioned places and artifacts. It was one of the

earliest stakeholder suggestions in the process. The functionality and usefulness was evaluated and as a result it emerged that, while in many systems there is only one general activity log, it would be more conducive to show information that is relevant only the screen at hand. Thus, the *Contextual Activity Log* was introduced.

Examples

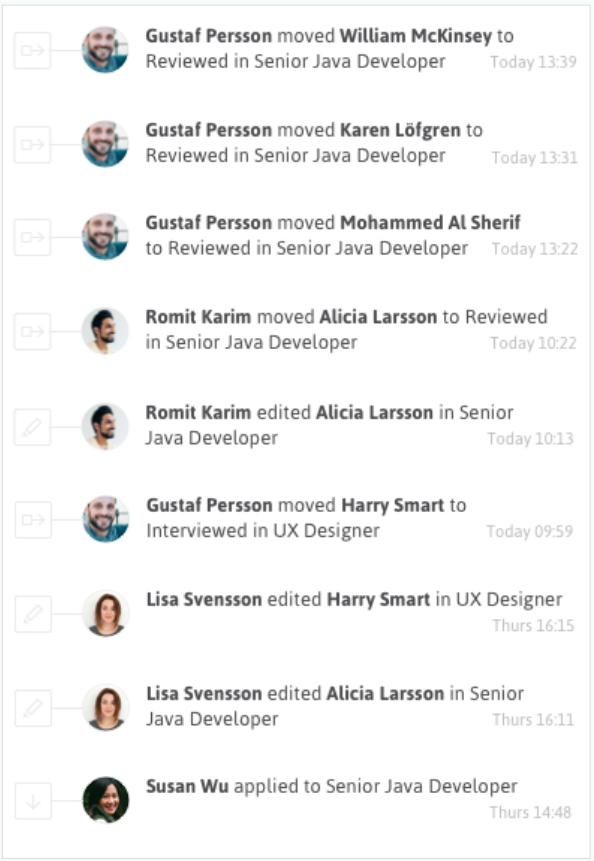


Figure 6.33: Example of a Contextual Activity Log on a higher level screen (showing the activities from all parts of the system).

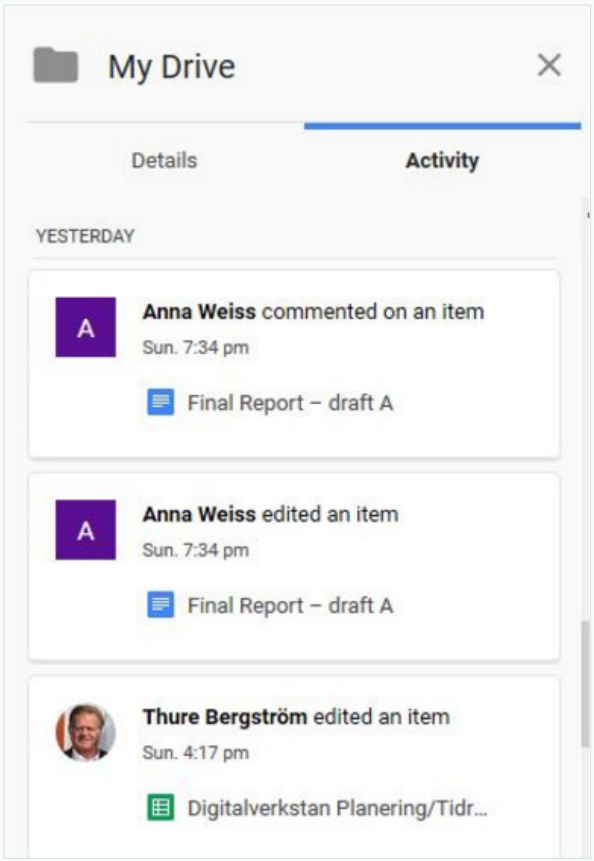


Figure 6.34: Google Drive displays activities relevant to the current folder.

Related Patterns

- Aliveness Indicator: Both the *Contextual Activity Log* and *Aliveness Indicator* have a similar purpose of helping users decide what to do, and understanding the state of the system.
- Novelty Indicator: Both the *Contextual Activity Log* and *Novelty Indicator* provide information about recent new additions in the system.
- User Info Preview: Both the *Contextual Activity Log* and *User Info Preview* can contain the *User Info Preview*, to make it easier to initiate communication with the users connected to a certain log item.

- Floor Control: Both *Floor Control* and *Contextual Activity Log*'s purpose is to display who is making changes to an artifact; *Floor Control* does this in real-time, and the *Activity Log* asynchronously.
- Curated Archived Information: *Curated Archived Information* contradicts the *Contextual Activity Log* on a short-term level, as the latter preserves all information. It is also more immediate, whereas the *Curated Archived Information* pattern concerns older information.

6.3.11 Aliveness Indicator

Problem

When some artifacts change frequently, it is difficult to know where updates have occurred.

Context

Apply this pattern when there are many artifacts in a system where information changes at unpredictable frequencies.

Solution

Artifacts have an indicator of how recently they have been interacted with.

Consequences

- This communicates which artifact is up to date and actively interacted with by users, and similarly conveys when an artifact has been neglected for a certain time period.

Danger Spots

- The right frequency of time for the indicator may differ from artifact to artifact, from system to system.
- Every artifact and system has different revision requirements

Why

To act as a reminder to users what needs to be dealt with in a system. With many artifacts in a system it is difficult for users to recall what exactly they have completed and to plan what to do next.

Evolvement

The conception of this pattern occurred during the card sort method. The main stakeholder, Henrik Enström, expressed interest in knowing "date of last contact", "date applied", and "date last updated" and why and when these would be applicable. He highlighted the importance of knowing how "warm" a candidate is, and of making sure that no one is forgotten over time. This also relates to building a fair and human system, where everyone is kept equally updated about the status of their application process. As part of the benchmark analysis during the first iteration it was identified that many ATSs show the parameter of "last activity" and then a date or timestamp. However, a visual timestamp solution was developed, which is faster to overview, and its three possible states make it easier for the user to categorize and take action accordingly. Schümmer

and Lukosch diagnose a similar problem for which they suggest an "Aliveness Indicator" (2007, p. 393). In their case however, it is intended for showing the activity or aliveness of community members, rather than system artifacts.

Examples

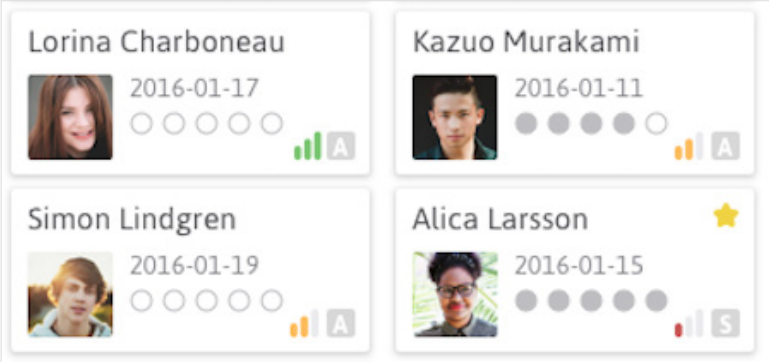


Figure 6.35: Example of Aliveness Indicator.

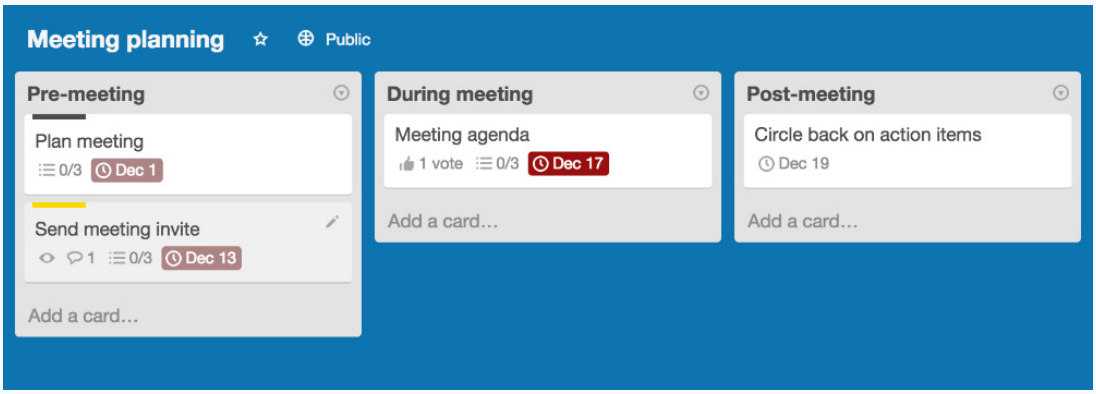


Figure 6.36: Trello utilizes aging cards for cards not interacted with for weeks.[2]

Related Patterns

- Contextual Activity Log: Both the *Aliveness Indicator* and *Contextual Activity Log* have a similar purpose of helping users decide what to do, and understanding the state of the system.
- Interaction Springboard: Both the *Aliveness Indicator* and *Interaction Springboard* present the current state of the system and thus support subsequent action planning.
- Novelty Indicator: Both the *Aliveness Indicator* and *Novelty Indicator* shows a changed state of a given artifact.

[2] http://help.trello.com/article/820-card-aging (accessed: 16.05.2016)

- Floor Control: Both *Floor Control* and *Contextual Activity Log*'s purpose is to display who is making changes to an artifact; *Floor Control* does this in real-time, and the *Activity Log* asynchronously.

6.3.12 Novelty Indicator

Problem

When new artifacts automatically appear in the system, it is difficult to distinguish the newly added from the previously existing artifacts.

Context

Apply this pattern when there are many artifacts in a system, and it is difficult to keep track of which ones are new.

Solution

Any artifact added newly into the system is marked as such, so that the users can review and process this new artifact accordingly.

Consequences

- Repeated (and unnecessary) reviewing of new items is avoided, because the artifacts requiring attention are clearly marked.

Danger Spots

- The indicator may behave different for every system: either the indicator is removed when one user has reviewed an item or when everyone involved has depending on what is important and more efficient for the users.

Why

To act as a reminder to users what is new to the system. With many artifacts in a system it is difficult for users to recall what was on display in the previous session.

Evolvement

This instance of rich visual modeless feedback communicates whether a new artifact appearing automatically in the system has been interacted with or not. While this may be a common element in email clients, for example, where it marks new messages, the idea for this pattern was developed relatively late in the process. The first impetus was given during the card sort method, where one of the recruiters admitted that she always clicks on applicants in the new-column several times, only to realize that she has already reviewed this person, and then – irked – closes the applicant card again. The *Novelty Indicator* was thus the result of eliminating this redundant operation. The appearance and behavior was tested, refined, and finalized in the later stages of the design.

Examples

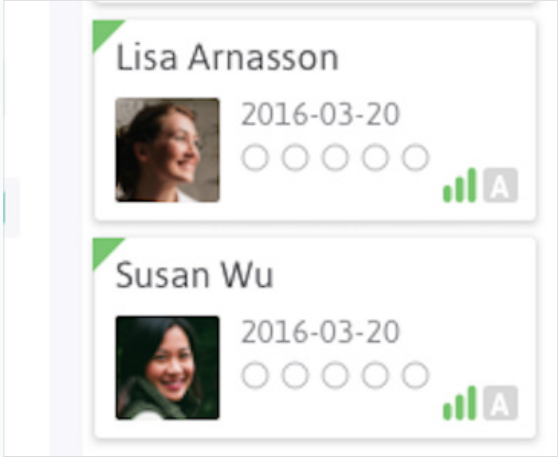


Figure 6.37: A green mark on the top left corner of the candidate card indicates that the artifact is new to the user.

Figure 6.38: The e-mail client Thunderbird uses bold text and green icons to show that an artifact has not been reviewed.

Related Patterns

- Activity Log: Both the *Novelty Indicator* and *Activity Log* provide information about recent new additions in the system
- Interaction Springboard: Both the *Novelty Indicator* and *Interaction Springboard* present the current state of the system and thus support subsequent action planning
- Aliveness Indicator: Both the *Novelty Indicator* and *Aliveness Indicator* shows a changed state of a given artifact.

6.3.13 Floor Control

Problem

Users may be editing the same artifact at the same time and unaware of each other's actions.

Context

Apply this pattern when users work in the system simultaneously.

Solution

Provide an indication that editing by another user is in process in shared collaboration spaces.

Consequences

- This indicator informs of ongoing editing, so other users are made aware of this. Once the edit has been committed, the change is dynamically updated, and the other users on the same view will see the updated version.

Danger Spots

- Users may avoid an item if it is currently being edited.
- Edit wars may emerge if users are disappointed with the outcome of the latest version.

Why

To prevent conflicting versions and loss of work.

Evolvement

The pattern emerged during the awareness mind map, where the questions "What are the others doing?", "live activity log", "show current location next to user avatar", "display real-time editing", and "show who is currently working on a job/candidate" was annotated. The latter two embody the birth of the *Floor Control* pattern. However, the functionality was not formalized until the validation scenarios revealed that it would be problematic if users were to edit information on a candidate card that was currently being edited by a fellow colleague and both parties were unaware of each other. This would lead to conflicting versions. Declaring the feature to be a pattern was finally consolidated by its presence in the collection by Schümmer and Lukosch, who also coined its congenial naming (2007, p. 226). The pattern is further validated by its usage in Trello, when a fellow user is editing the contents of a card momentarily.

Examples

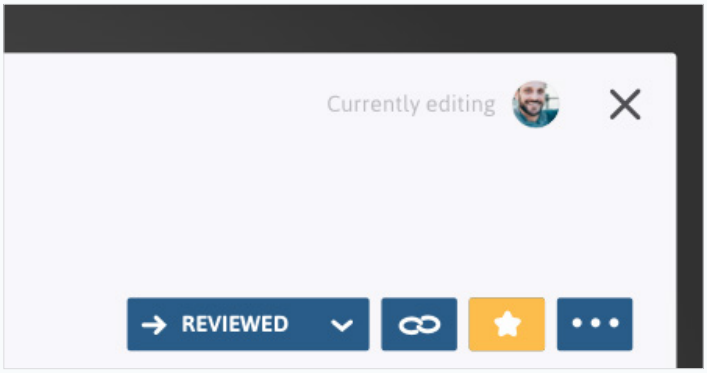


Figure 6.39: Floor Control in Honeycomb, showing the avatar of the currently editing user

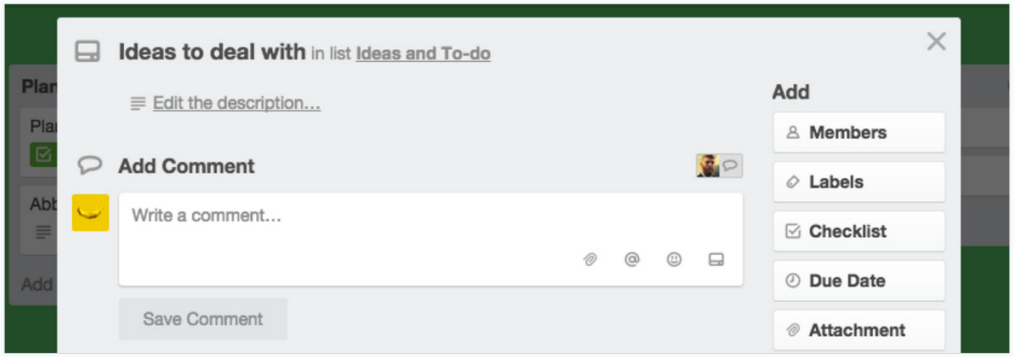


Figure 6.40: Trello displays a small profile image next to a speech bubble icon when other users are editing the currently active artifact.

Related Patterns

- Activity Log: Both *Activity Log* and *Floor Control*'s purpose is to display who is making changes to an artifact; *Floor Control* does this in real-time, and the *Contextual Activity Log* asynchronously.

6.3.14 Shareable Item Link

Problem

Only registered users can collaborate in the collaborative web app, which makes it difficult to have non-members contribute in any way.

Context

Apply this pattern when non-members should be able to be invited to collaborate.

Solution

Provide direct link to cordoned off, specified artifact or area to non-members and unlock certain functionalities for them to interact with the content.

Consequences

- One way to give partial access to confidential content instead of creating new accounts for short review of information
- More input and expertise that can be immediately connected to the content in question, without extra effort from existing members

Danger Spots

- Confidential information may be spread more carelessly and the share links may end up in unintended places
- Important to decide on what information is contained with the link, so not company or personal secrets gets distributed

Why

To extend access to people outside of the system, who can provide valuable input to a certain artifact.

Evolvement

The pattern emerged after reviewing the Honeycomb prototype. During this activity, the strong relation with collaboration, with the special characteristic of enabling participation of people, who are not registered users of the company, was found. The *Shareable Item Link* thereby extends the collaboration space beyond the limits of the system. This functionality is present in other ATSs that we have reviewed, for example Workable and Teamtailor, but also in other collaborative web applications this is a common feature; among them Trello, Google Drive, Dropbox, and Asana.

Examples

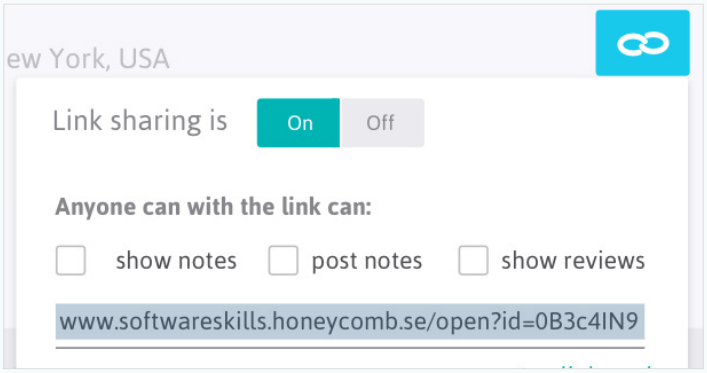


Figure 6.41: Honeycomb provides a shareable item link in a dialog.

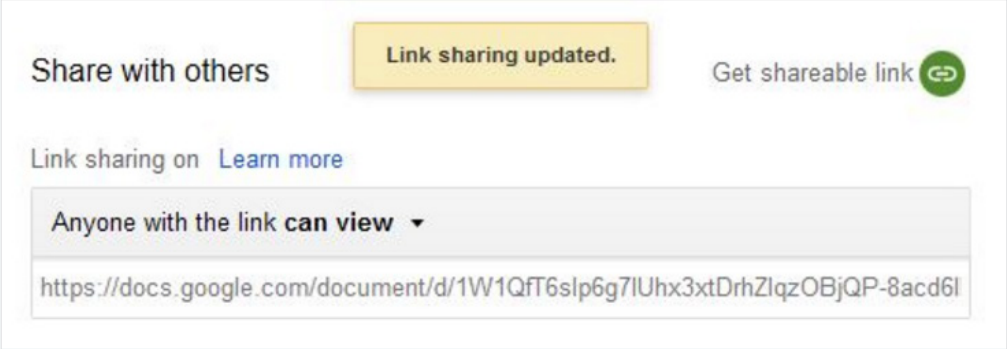


Figure 6.42: Google Docs allows sharing specific content with non-members via a dedicated link.

Related Patterns

- Invitation: Both *Sharable Item Link* and *Invitation* are initiations for collaboration, the only difference being that the *Sharable Item Link* extends this invitation to external contributors.

6.3.15 Curated Archived Information

Problem

Information input by users and related to the working process can become irrelevant over time.

Context

Apply this pattern when different kinds of information can confuse and slow down the decision making process.

Solution

Archive information that is not longer relevant for the decision making process. The archived data should only contain information that has relevance for future purposes. When artifacts are archived at the end of an active process, only the artifact information itself remains visible, and the information that was added by users is hidden.

Consequences

- Non-awareness pattern in the sense that it filters out information for users' convenience
- Information that may shed negative light on users is not permanently stored and thus avoids perceiving this information out of context
- Information is not lost, but more complicated to access which protects the user's integrity

Danger Spots

- Decide what to filter out carefully since it may be difficult to foresee which pieces of information will be relevant at a later point in time
- If the archive occurs automatically, consider the option to retrieve the full information

Why

To store only the information that has long-term relevance, and thus make the reviewing of archived content more efficient and less biased.

Evolvement

The treatment of data connected to a concluded process had been a recurring discussion point since the awareness mind map, where it was first discovered. Many issues regarding user privacy and the integrity of candidate information were explored henceforth, and this

pattern is one of the manifestations of the conclusions thereof. Declaring this functionality as a pattern was as a result of the validation scenarios.

Examples

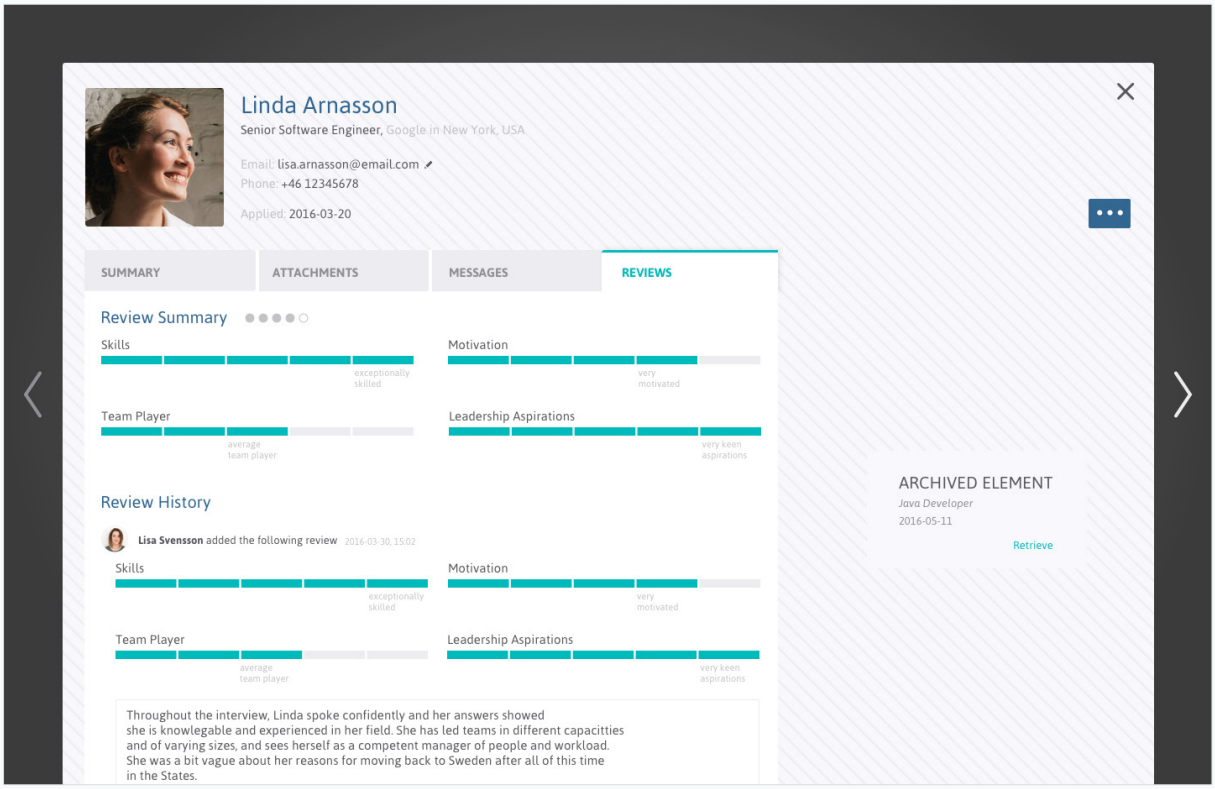


Figure 6.43: Displays the difference between an active and an archived candidate card. The latter hides Shared Annotations and possible inputs.

Related Patterns

- Contextual Activity Log: *Curated Archived Information* contradicts the *Contextual Activity Log* on a short-term level, as the latter preserves all information. It is also more immediate, whereas the *Curated Archived Information* pattern concerns older information.
- Shared Annotations: Curation removes the *Shared Annotations*.

7
Discussion

What follows is a discussion on all aspects of the thesis. This involves contemplations on the methods employed, analyses of the results produced, future work to be done, and the implicated ethical questions. Throughout this project, we have continuously gathered theoretical and empirical knowledge in pursuit of answering the question of how ATSs can be designed to facilitate awareness, and efficient collaboration for teams working together through CSCW. The differentiation between real-time and asynchronous collaboration put forward by Grudin and Poltrock (see figure 2.3) have made a significant impact on our deliberations on the patterns but also on our understanding of awareness in general. Our Honeycomb ATS is a system largely supporting asynchronous collaboration. Being aware of this early on helped to evaluate findings and consequently plan suitable focus areas in the subsequently conducted activities. As the three keywords *awareness, collaboration,* and *efficiency* emerged as recurring concepts in the intermittent results of our process, we dubbed the trio ACE. This also helped us to keep track of those three principles, and measure the findings against. Another influential source of inspiration were Schümmer and Lukosch, who, in their pattern collection, manage to wrap many of the subtleties and required sensitivity when dealing with computer-mediated interaction. For example, when pointing out danger spots connected to *User List*, they write that "an important acceptance of awareness patterns is trust and privacy. User monitoring only works if it is mutually accepted by all participants. Otherwise, the effect will soon be that users complain about being monitored or try to find ways to evade the awareness features" (Schümmer and Lukosch 2007, p. 321). This illustrates the delicate proverbial towing of the line between providing awareness and opportunities for communication and collaboration on the one hand, while maintaining productivity, efficiency, and privacy on the other hand, to achieve just the right amount of awareness, which we constantly engaged in throughout our assessment of patterns and how to best recommend them.

The pattern collection presented in section 6.3 is deliberately tailored to ACE, focusing on the cross section of this specific area for the particular application within the ATS context. Most of the HCI pattern collections introduced in section 2.2.2, like by Tidwell or Vora are

broader in their capacity, but they have grown over years of work and have had the benefit of iterative contributions. In its limited scope of time and resources, the creation process of our ACE pattern collection was more focused on the quality of the patterns, rather than quantity

7.1 Reflection on the Process

Overall, the thorough explication of suitable methods, their precise sequencing into iterative cycles, and the consecutive planning proved to be an excellent preparation for the project's work. Especially, the recurrent evaluative methods ensured that we were using the results from the precedent activities and channeling them into the subsequent work. Also, setting up a process diary, where each time the departure status, the conducted procedure, and the findings were recorded, helped to keep track of intermediate learnings for later reference.

Overall, we kept to the project plan. The strong influence from the HCD process – with its emphasis on iterations – on our planning efforts meant that we were constantly using and improving the results gathered from preceding methods. We were easily able to adhere to the number of iterations and the respective focus areas stipulated in our plan, which we take to indicate that we already had a good grasp of the project scope at that time. As for the influence of GDD on our process, it was less structural and more evident in specific artifacts. For example, the notable focus on the requirements that we kept coming back to, and also the employment of methods such as validation scenarios. Within the cycles, some switching around of methods occurred, when we needed more input in order to carry out a certain method, for instance using the card sort method as an inspiration method, instead for evaluation. An impromptu divergence from the plan occurred after working on the wireframes for several weeks, where we realized that we had taken so many micro-decisions that we needed to make a point to chronicle them. So we set out to annotate every element in the Illustrator wireframe, recording the status quo at the time and the reasons why. This in turn influenced the subsequent work, as having the information available in context – on the canvas where the designing was taking place – made it easy to reconstruct why an element was a certain way. Based on this we could challenge and question our previous thinking as and when new findings emerged. Whenever we realized that something was inconsistent, we could improve the design and add a new bit to the annotations. The thus created history logs are a testimonial to the thorough scrutiny, based on different factors, the elements and views have undergone, and that each has their raison d'être. We will likely take this method of contextual interface annotations along to future projects of similar scale, to be able to record the development process effectively. The flexibility with the project plan at that lower level (not affecting the overall iteration structure) demonstrated here, is evidence of us asking ourselves frequently what it was that we needed to learn and create next, in order to have a purposeful process, instead of willfully keeping

to the plan simply for the sake of it. It allowed us to work efficiently and to react to and process new findings from the performed activities.

From some of the methods conducted in the first and second iteration cycles, we did not obtain exactly the tangible and prescriptive results we had anticipated, but they opened up more questions and were a way to spark in-depth discussions. In that way, they proved to be catalysts that propelled us forward each time, even though sometimes it might not have felt so productive. On hindsight, we realize that we had considered many different aspects from diverse angles, which enabled us to build the comprehensive and well-grounded product that we did.

There are two aspects of the project's circumstances that were influential over the course of events in practical terms. For one, being located for the most part at the offices of Software Skills meant that we were immersed in the recruiters' everyday tasks. We heard snippets of phone conversations, caught glimpses of nervous applicants waiting in front of the interview room, learned about how Henrik's sales meetings went and how the clients commented on the products and services. It also meant that we got input on a rolling basis, and were able to ask questions directly as issues emerged. The second aspect was the group of five bachelor students, who was arranged to implement parts of the system we built at the same time as our project was underway. On occasion this involved preponing a practical decision, where otherwise we might have taken more time to consider. But that in turn also created a more authentic way of working and the agile process was fruitful as it connoted a further source of feedback for our product. The close communication was a strong influence on especially the permission handling and the embedded messaging function, and generally the consistency of data objects throughout the system. That we were not working towards a hypothetical realisation in the vague future, but saw the immediate impact our process had on the technical implementation, was inspiring. Actually, both these aspects served as constant reminder of the realness of the stakes, keeping us motivated and consistently engaged with the project.

Our personal sore spot, leaving us somewhat discontented with the results, is the unsatisfactory level to which we were able to evaluate them due to the time constraints. We had planned for another round of usability testing to validate the prototype after the first design iteration, however, as our focus has shifted more towards awareness and collaboration we would be thrilled if we could test the prototype and with it the patterns represented therein with an actual team. But setting up the real-time tool required for that would go far beyond the temporal and technical mold of the project. How this could be achieved is discussed further below in the next section.

7.2 Reflection on the Results

The resulting prototype would not have been satisfactory in answering the research question on its own. However, the effort of the design process helped create the understanding necessary for developing a well-grounded pattern collection. Conceptualizing and building the prototype demanded a larger share of attention throughout the process, but without the first-hand experience of what it means to design a web application of this kind, the understanding of why certain design choices had to be made would be lacking. The ATS might not be the most compelling *research* result, but it was an integral part of the process that served as the foundation to the pattern collection.

The initial research and analysis of data for the prototype informed not only the layout and necessary functions for efficient workflow, but also provided direction in how to find an equilibrium of awareness and integrity, which was a constant balancing act. Without the development, testing, and resulting feedback it would have been impossible to determine which functionality to preserve, nurture, and to transform into design patterns for collaborative applications. The patterns derived from this process are prescriptive: They identify common problems arising in groupware, present corresponding solutions, and explain their validity.

During our opening talks with the stakeholders, specific features were mentioned or even requested. However, we vowed in the beginning of the project to base all of the system's features on the research and test results. The goal was always to be advocates for the prospective users and focus on building the best possible solution based on their needs. Also, it was vital to keep sight of the research question and explore requirements relevant to it, instead of paying heed to each and every new suggestion that the stakeholders at the company came up with. As we were hired as interaction design experts, we knew to rely on the methods to provide us with the raison d'être for features, instead of whims or assumptions. Therefore, we made an effort to extract the problems that those suggestions were triggered by, and explore and solve them throughout the project – perhaps with other solutions, perhaps by making use of the suggestions, but that was for the method results to show. This way, the product is also applicable to a wider context and satisfies other future users of the system, whose requirements do not exactly overlap with the primary stakeholders here.

The format of the presented patterns is intentionally concise. Since patterns are to be understood and readily applied by other designers we devised a format that focuses on the problem, context of the problem, the proposed solution, and the justification for the solution. Attempting to address specific problems, the pattern collection is thus prescriptive, rather than descriptive. To further support designers, we organized our patterns in a model and provided information of how the various patterns connect, by either containing, complementing, or contradicting each other. The intention of designing the patterns this way was to give designers enough instruction and support for independent use. Whether

these connections and the model actually enhance third party understanding is something we would like to investigate further, as discussed in future work.

We did evaluate the patterns by creating the Honeycomb ATS and implementing each concept therein, however, we do not have definitive, measurable indicators of how much the patterns contribute to more awareness, collaboration, and efficiency. To obtain accurate data, this kind of evaluation would require building to fully working prototypes, test and observe over time and conduct a thorough evaluation. To isolate our patterns in a design and thereafter measure an abstract concept such as awareness would require a lot of additional work. The Wizard of Oz method described in the methodology section could be a possible solution. This kind of test would have been time-intensive to set up, and was therefore not possible in the timeframe available to us. Therefore, we had to rely on our research and the stakeholders' previous experiences of similar or identical functionalities in other collaborative systems.

The *ACE pattern model* represents the third type of result. It was not essential for our progress of discovering and inventing new patterns, but its construction served as an evaluation and discussion tool. By mapping out the patterns in the model, we could see whether a pattern actually answered or covered our research question and if the focus was too strong in one aspect and thus opposed another goal. The model inspired us to merge and fork some of the collected patterns to simplify and optimize for generalizing a given pattern. It also served as an evaluative artifact, because it was a way to confirm that the three aspects awareness, collaboration, and efficiency had each been addressed fairly, and thus bestowed a certain level of completeness to the collection.

7.3 Validity

The patterns were carefully selected and their exact formulations improved and tweaked to work for – not exclusively Honeycomb – any collaborative web application, even though our emphasis was on finding suitable design patterns for ATSs. The final pattern collection is based on our particular circumstances and prerequisites, however more potentially relevant for wider use patterns can be found in the appendix. The here presented paradigms have to be regarded as suggestions for other designers and further evaluation for other application scenarios is encouraged.

Even though the pattern collection evolved as a result of this individual project, our previous interaction and interface design knowledge, and experience of developing applications also went into the process. Throughout the whole design process, we consciously opted to use other web applications, such as Google Drive, Coggle, Trello, and Asana, which are geared towards collaboration and shared awareness. This gave us more practical experience with features that augmented our workflow and informed us of each other's progress. Immersing ourselves in a computer-mediated work process contributed to our general

understanding of awareness, collaboration, and efficiency in groupware. But it was mainly the research and development of the prototype that generated the necessary knowledge for patterns, which in turn strengthened the design implementation, and which exemplified how the patterns are connected to each other.

To a large part, the rigorous process that the project adhered to and the multiple sources of input can attest to the degree of validity that the presented pattern collection has. As recounted in the process chapter, the primary and secondary research methods in conjunction with evaluations of results, repeated over six iteration cycles, strengthen the relevance of the patterns. The specific journey of each pattern, and how it came to be based on multiple sources, is summarized in section 6.4.2 pattern evolvement. The validity of each pattern is further strengthened by not only showing occurrences of it in the Honeycomb ATS, but each is linked to an example present in other ATSs or collaborative web applications.

7.4 Generalization

The entire design process has been concerned with finding patterns suitable for awareness and collaboration, which are both broad concepts in their own right. Hence, we continuously called to mind that these concepts may operate differently or exhibit varying effects in systems that are not Honeycomb. Throughout the project's development, the interpretation of awareness and efficient collaboration has been investigated from a multitude of different angles. Our collective experience of the process influenced how the patterns are conveyed through the structure. The wording has been deliberated carefully to help outside users of the collection to recognize the characteristics that are shared with other types of applications. On top of displaying how the patterns are embodied in our ATS, examples from other collaborative web applications are provided, collected from the two benchmarking activities. This is a further indicator for the generalizability of the collection outside of the e-recruiting context.

Another point supporting the claim for wider applicability is contained in the process: Consulting written sources and other pattern collections has been an important source of inspiration. Especially Schümmer and Lukosch's book "Patterns for Computer-Mediated Interaction" has been a key authority, providing reference to the level of completeness we had reached and highlighted the aspects we had been neglecting. Having these reference points to contextualize the pattern collection introduced here, further increases our confidence in its validity and generalizability. However, further verification is necessary in order to conclude for which applications and situations the collection holds true and where

it begins to lose relevance. How this can be achieved is discussed further in the following section.

7.5 Future Work

The next step for this project would be to evaluate the effects of the patterns in the existing collection. However, iterating and developing the prototype further will most likely result in new findings, which in turn could yield additional patterns. The time constraints limited us to a certain number of features to be incorporated in the first release of the product, but there is a list of components that would be desirable for future versions. It includes items such as a scheduling feature to set up and manage meetings in-app, an embedded task coordinator that lets users assign tasks to themselves and others to more effectively coordinate work, and a detailed reporting tool that provides the team with visually represented performance indicators. These are expected to result in further sub-features and exhibit aspects related to awareness and collaborative design patterns. An investigation into, for instance, which data to display in the dashboard view or how to design feedback indicators that will improve the workflow and coordination of tasks, could lead to further minor improvements.

Adapting the application to be viewed in the browser of smartphones by making it responsive and researching what features to strip away to make it powerful yet productive on the go, may not generate a wide array of new additions to the pattern collection. However, it would supply insights into how to adapt the already detected patterns to the new conditions such a transformation would cause.

As for testing the pattern collection, there are several steps to be taken to collect more conclusions on the validity. Setting up and conducting the previously described Wizard of Oz method would be the most viable at this point. This method uses a prototype that, while not functioning fully, the testers can create feedback behind-the-scenes by faking the system behavior without the participant's' knowledge. Conceiving and technically setting up a scenario, where certain actions by the test participant would result in predefined events and state change, would still be a time-consuming feat, but receiving more user input at this development stage is undoubtedly called for. The thus adapted patterns should be reflected in the prototype as well, and after its adoption it would be an appropriate time to initiate the beta launch. Here, more user feedback can be collected, more informally through surveys, or to evaluate specific features by conducting AB tests, as described in the methodology section. As a more long-term commitment, finding another team of designers who would be ready to make use of the here proposed pattern collection for an applicable project, like the development of an ATS or other collaborative application would be invaluable. Receiving feedback from such a study would produce greatly beneficial input for the subsequent enhancement of individual patterns and the collection overall.

As previously mentioned in section 7.3, producing and evaluating a second ATS, where all of the proposed patterns and corresponding features are intentionally stripped away, would be a compelling experiment to conduct and would provide more appreciation of the pattern's impact.

Furthermore – as this project is rooted in the practice of interaction design – it would be functional to study the results of these proposed evaluations with a business-oriented mindset and add another layer of legitimacy to the design. Such an assessment, through key performance indicators for instance, should reveal the monetary benefit (as opposed to other ATSs or Software Skills' previous solution) and through this quantifiable juxtaposition validate the raison d'être.

7.6 Ethical Issues

Ethical issues that may arise pertain mostly to the data security of personal information of the applicants in the database. The fact that many ATSs are created in the United States or at least stored on US-based servers, where different laws are in place, makes European customers dubious about who exactly is privy to all of this personal information. By creating a European-based system that is subject to the local rules and regulations, this issue is rectified to some extent. However, it does require awareness of the legislation, such as the "European Directive on Data Protection", where, for example, it is stipulated that users have to be notified and their consent be sought of how their data is going to be used (Laudon 2016).

Apart from storing data securely according to law, what is most relevant for conceiving the visual and interaction aspects, is perhaps which pieces of information are shown when and to whom. This occurs mostly in large companies, where employees may apply for other departments internally, but their current superiors could be vexed when inadvertently learning of the reorientation desires of their underling. An ATS has the power to address such social faux-pas by limiting the access to certain information for different roles. The trade-off here is that the system receives yet another layer of complexity. ATSs have been criticized heavily for the merciless screening process that candidate resumés are put through and eliminated before a human eye gets to look at them. While in theory this is supposed to help recruiters to efficiently narrow down the applications to the most feasible ones, in practice many good candidates have already been weeded out at this stage, simply because they lack certain keywords. As one recruiter puts it: "Fail to meet one the [pre-set criteria], and your application gets tossed, even if a good HR director might have spotted your potential" (Hansen 2013). This shows that some systems are designed too rigidly, too unforgiving for human input that is inherently unpredictable. These should be tools that make the hiring process easier, instead of hindering their very own purpose. For this reason we decided to not include this functionality for our ATS

prototype and focus instead on optimizing the human effort required in the hiring process by designing a supportive digital tool.

An important factor that pertains to looking for new employees is adhering to the laws against discrimination. Conducting a fair hiring process means to be unbiased towards candidates, regardless of their "sex, race, disability, age, sexual orientation, gender reassignment, marriage and civil partnership, pregnancy and maternity, and religion or belief" (Acas). It could therefore be considered the designers' ethical responsibility to devise an ATS that discourages or even actively prevents the filtering of these parameters. In our case, we therefore do not have a dedicated field to display a candidate's age for example, and also do not allow any filters according to sex or gender.

Another issue was the fact that this particular ATS had to integrate at certain points with the existing product of Software Skills; For example, the job ad feature, where jobs are posted and managed. This could have led to having to sacrifice some of the conceptual patterns for the seamless integration with the present product. However, thanks to the flexibility of our primary stakeholder, who was very open to our suggested improvements, they rather adapted the system the results of our research, than us having to adapt our results to fit into the existing technical structure.

8

Conclusion

This master thesis project has been undertaken to examine the landscape of collaborative web applications, and applicant tracking systems in particular, as well as understanding aspects related to e-recruiting. The posed research question was:

> *How can applicant tracking systems be designed to facilitate awareness and efficient collaborative work across a team?*

In pursuit of answering this question, a diverse range of research has been undertaken. It looked into recruiting, CSCW and awareness, and pattern collections for HCI, and on the practical side, assessed existing ATS and collaborative web applications. Overall adhering to the HCD process and drawing from the gathered inspiration material, the ideation methods led to numerous intermediate implementation artifacts, on which evaluations were administered. This resulted in the holistic concept and design of an comprehensive applicant tracking system, named *Honeycomb* that is detailed in section 6.1. It consists of the views shown in the sitemap (see figure 6.1), where the all job openings, applicants and their submitted information, and the corresponding hiring workflow can be managed. Furthermore, the design result also integrates all necessary functionality connected to users, and incorporates all of the proposed ACE patterns.

The other result is connected more explicitly to the research question: To support teams working with and through groupware, *awareness* and *collaboration* are two key factors that need to be respected, and that can be mapped against *efficiency* to measure how well this is achieved. We have proposed a collection of patterns that addresses each of these three aspects, and that provide guidance for designing such computer-mediated applications. For easier overview, we have grouped the fifteen patterns into *Information Sharing*, *User Management*, and an overlapping area of both according to their functional purpose (see figure 6.14 for an illustration of this):

Information Sharing:

- *Shared Annotations*
- *Differentiated Voting*

- *Curated Archived Information*
- *Aliveness Indicator*
- *Novelty Indicator*
- *Embedded Messaging*

Intersection of both Information Sharing and User Management:
- *Interaction Springboard*
- *Floor Control*
- *Contextual Activity Log*

User Management:
- *User Profile*
- *Team*
- *Invitation*
- *User List*
- *User Info Preview*

Each pattern is presented in the form of *problem, context, solution, consequences, danger spots, why, examples, evolvement*, and *related patterns*, where the examples illustrate how the concept is implemented in Honeycomb, and where it can be found in other collaborative web applications. The *related patterns* section is also significant to understand the occurring interdependencies.

Evaluating the patterns according to the parameters (ACE) of the research question with a triangular layout led to an unplanned but welcome result. The patterns mapped as per *awareness, collaboration,* and *efficiency* emerged to answer their inherent conceptual purposes, and with that also addressed the research question directly. At the same time, the pattern model provides a visual summary of all patterns and may guide potential users how to best apply the concepts for their project.

More work to further validate the here proposed collection is needed. As suggested, this can be approached by building a prototype that better simulates both the authentic workflow and presence of a team, and observing users operating the system. From our research into CSCW and collaborative web applications, we know that the patterns are to a large extent applicable to not only ATSs, but can also be relevant for a wider scope of systems. For neither purpose the collection is complete however, and there is definitive opportunity for further exploration in the future.

References

Adlin, T., and Pruitt, J. 2010. *The Essential Persona Lifecycle: Your Guide to Building and Using Personas.* Morgan Kaufmann.

Alexander, C., Ishikawa, S., and Silverstein, M. 1977. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press.

Alexander, C. 1979. *The Timeless Way of Building*. Oxford University Press, Oxford, UK.

Bartram, D. 2000. "Internet recruitment and selection: Kissing frogs to find princes." *International Journal of Selection and Assessment* 8(4), 261–274.

Baxter, K., Courage, C., and Caine, K. 2015. *Understanding Your Users: A Practical Guide to User Research Methods*. Morgan Kaufmann.

Bhattacharyya, D. L, and Kumar, D. 2009. Organizational Systems, Design, Structure and Management. Mumbai: Himalaya Publishing House.

Björk, S., Lundgren, S. and Holopainen, J. 2003. "Game Design Patterns". In Copier, M. & Raessens, J. (Eds.) *Level Up - Proceedings of Digital Games Research Conference* 2003, Utrecht, The Netherlands, 4-6 November 2003.

Borchers, J. O. 2001. *A Pattern Approach to Interaction Design.* John Wiley & Sons.

Brettel, M., Friederichsen, N., Keller, M., Rosenberg, M. 2014. "How Virtualization, Decentralization and Network Building Change the Manufacturing Landscape: An Industry 4.0 Perspective." *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering* 8, No:1, 2014 . http://www.waset.org/publications/9997144.

Cassidy, T. 2011. "The Mood Board Process Modeled and Understood as a Qualitative Design Research Tool". *Fashion Practice,* 3:2, 225-251.

Carlsson, D. 2004. *A Categorization of HCI Patterns*. Umeå University.

Chapman, C. N., and Milham, R. P. 2006. "The Personas' New Clothes: Methodological and Practical Arguments against a Popular Method." *PsycEXTRA Dataset*. doi:10.1037/e577592012-003.

Chapman, C. N., E. Love, R. P. Milham, P. ElRif, and J. L. Alford. 2008. "Quantitative Evaluation of Personas as Information." *Proceedings of the Human Factors and Ergonomics Society Annual Meeting Human Factors and Ergonomics Society.*

Meeting 52 (16): 1107–11.

Cooper, A., Reimann, R., Cronin, D., and Noessel, C. 2014. *About Face: The Essentials of Interaction Design.* John Wiley & Sons.

David, B., Delotte, O., Chalon, R., Tarpin-Bernard, F., and Saikali, K. 2003. *Patterns in Collaborative System Design, Development and Use*. Ecole Centrale de Lyon.

Dearden, A., and Finlay, J. 2006. "Pattern Languages in HCI: A Critical Review." *Human-Computer Interaction 21 (1)*: 49–102.

Dourish, P., and Bellotti, V. 1992. "Awareness and Coordination in Shared Workspaces." *In Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work - CSCW '92*. doi:10.1145/143457.143468.

Dow, S., MacIntyre, B., Lee, J., Oezbek, C., Bolter, J. D., and Gandy, M. 2005. "Wizard of Oz support throughout an iterative design process," *in IEEE Pervasive Computing*, vol. 4, no. 4, pp. 18-26, Oct.-Dec. doi: 10.1109/MPRV.2005.93.

Ebersbach, A., Glaser, M., Heigl, R., and Warta, A. 2008. *Wiki: Web Collaboration*. Springer Science & Business Media.

Enström, Henrik. *Interview by Johannes Lundqvist and Anna Weiss*. Gothenburg, Sweden, 21.01.2016.

Feldman, D.C., and B.S. Klaas. 2002. "Internet job hunting: A field study of applicant experiences with online recruiting." *Human Resource Management 41(2)*, 175–192.

Furtmueller, E., C. Wilderom and R. van Dick. 2010. "Sustainable e-recruiting portals: How to motivate applicants to stay connected throughout their careers?" *International Journal of Technology and Human Interaction 6(3)*, 1–20.

Furtmueller, E., Wilderom, C., and Tate, M. 2011. "Managing recruitment and selection in the digital age: E-HRM and resumés." *Human Systems Management (30)*, 243–259.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Reading.

Green, P., and Haas-Wei, L. 1985. "The rapid development of user interfaces: Experience with the Wizard of Oz method." *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* October 1985 vol. 29 no. 5 470-474. doi: 10.1177/154193128502900515.

Gross, T., and Koch, M. 2006. "Computer-Supported Cooperative Work: Concepts and Trends." In: *Proc. Conf. of the Association Information And Management (AIM), Lecture Notes in Informatics (LNI)* P-92.

Guerrero, Luis A., and Fuller, D. A. 2001. "A Pattern System for the Development of Collaborative Applications." *Information and Software Technology 43 (7):* 457–67.

Gueutal, H.G. and Stone, D.L. 2005. *The Brave New World of e-HR: Human Resources Management in the Digital Age,* Jossey- Bass, San Francisco, CA.

Gutwin, C., and Greenberg, S. 2010. "A Descriptive Framework of Workspace Awareness for Real-Time Groupware." *Computer Supported Cooperative Work,* Kluwer Academic Press.

Holm, A. B. 2012. "E-recruitment: Towards an Ubiquitous Recruitment Process and Candidate Relationship Management". In *Zeitschrift für Personalforschung, 26(3)*, 241-259.

Hübscher, C., Stefan L. Pauwels, Roth, S. P., Bargas-Avila, J.A., and Opwis, K. 2011. "The Organization of Interaction Design Pattern Languages alongside the Design Process." *Interacting with Computers 23 (3)*: 189–201.

IDEO.org. 2015. Field Guide to Human-Centered Design.

Imine, A. 2009. "Coordination Model for Real-Time Collaborative Editors." In *Lecture Notes in Computer Science,* 225–46.

ISO. 2010. ISO 9241-210:2010(E). *Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems.* Geneva: ISO.

Johns, T. and Gratton, L. 2013. "The Third Wave of Virtual Work." *Harvard Business Review.* http://www.harvardbusiness.org/sites/default/files/HBR_Third_Wave_of_Virtual_Work.pdf.

Kelley, T. 2001. *The Art of Innovation.* Profile Books, Croydon, Surrey.

Koong, K. S., Liu, L.C., and Williams, D. L. 2002. "An identification of internet job board attributes." *Human Systems Management 21(2)*, 129-135.

Kruschitz, C., and Hitz, M. 2010. "Human-Computer Interaction Design Patterns: Structure, Methods, and Tools." I*nternational Journal on Advances in Software vol 3 (1 & 2)*.

Laudon, K. C., Laudon, J. P. 2011. *Management Information Systems.* 12th Edition. Prentice Hall.

Lee, I. 2007. "An architecture for a next-generation holistic e-recruiting system." *Communications of the ACM*, 50(7), 81-85.

LUMA Institute. 2012. *Innovating for People: Handbook of Human-Centered Design Methods.*

Lundgren, S., Fischer, J. E., Stuart, R., and Torgersson, O. 2015. "Designing Mobile Experiences for Collocated Interaction." In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '15.* doi:10.1145/2675133.2675171.

Nielsen, L. 2013. *Personas - User Focused Design.* Springer Verlag.

Norman, D. A. 2005. "Human-Centered Design Considered Harmful." *Interactions 12 (4)*: 14.

Preece, J., Sharp, H., and Rogers, Y. 2015. *Interaction Design: Beyond Human-Computer Interaction.* John Wiley & Sons.

Redström, J. 2006. "Towards User Design? On the Shift from Object to User as the Subject of Design." *Design Studies 27 (2)*: 123-39.

Schmidt, K. 2002. "The Problem with 'Awareness': Introductory Remarks on 'Awareness in CSCW'." *Computer Supported Cooperative Work: CSCW: An International Journal 11 (3-4)*: 285-98.

Schümmer, T., and Lukosch, S. 2007. *Patterns for Computer-Mediated Interaction*. John Wiley & Sons.

Scott, B., Neil, T. 2009. *Designing Web Interfaces*. O'Reilly Media.

Tenenberg, J., Roth, Wolff-Michael, and Socha, D. 2015. "From I-Awareness to We-Awareness in CSCW." *Computer Supported Cooperative Work: CSCW: An International Journal.* doi:10.1007/s10606-014-9215-0.

Stone, D. L., Deadrick, D. L. 2015. "Challenges and opportunities affecting the future of human resource management." *Human Resource Management Review 25 (2015)* 139–145. http://dx.doi.org/10.1016/j.hrmr.2015.01.003.

Thompson, L. F., Braddy, P. W., and Wuensch, K. L. 2008. "E-recruitment and the benefits of organizational web appeal." *Computers in Human Behavior, 24(5)*, 2384-2398.

Tidwell, J. 2005. *Designing Interfaces.* O'Reilly Media.

van Welie, M., van der Veer, G. C., and Anton, E. 2001. "Patterns as Tools for User Interface Design." In *Tools for Working with Guidelines*, 313–24.

van Welie, M., Gerrit C., van der Veer. 2003. *Pattern Language in Interaction Design.* Vrije Universiteit.

Vora, P. 2009. *Web Application Design Patterns.* Morgan Kaufmann.

Wang, J., and Da Wei, C. 2012. "Survey on Collaborative Awareness Model for CSCW and Trends of the New Age." *Applied Mechanics and Materials* 155-156: 357–62.

Wentzlaff, I., and Specker, M. 2006. "Pattern-Based Development of User-Friendly Web Applications." In *Workshop Proceedings of the Sixth International Conference on Web Engineering - ICWE '06*. doi:10.1145/1149993.1149996.

Williams, A. 2009. "User-Centered Design, Activity-Centered Design, and Goal-Directed Design." In *Proceedings of the 27th ACM International Conference on Design of Communication - SIGDOC '09*. doi:10.1145/1621995.1621997.

Web References

Acas. Recruitment and Induction. Accessed 16.01.2016 through http://www.acas.org.uk/media/pdf/8/b/B05_1.pdf

Asana. Accessed 19.02.2016 through https://asana.com/product.

Axure. Accessed 22.04.2016 through https://axure.com.

Basecamp. Accessed 19.02.2016 through https://basecamp.com.

Bychkov, D. 2013. "Desktop Vs. Web Applications: A Deeper Look And Comparison". Segue Technologies. Accessed 10.02.2016 through http://www.seguetech.com/blog/2013/06/07/desktop-vs-web-applications-deeper-comparison.

Coggle. Accessed 16.02.2016 through https://coggle.it.

Doodle. Accessed 19.02.2016 through http://doodle.com/?home.

Fallon Taylor, N. 2016. "Hiring in the Digital Age: What's Next for Recruiting?" Business News Daily. Accessed 05.02.2016 through http://www.businessnewsdaily.com/6975-future-of-recruiting.html

Figma. Accessed 16.02.2016 through https://www.figma.com.

Flaherty, K. 2015. "How Much Time Does It Take to Create Personas?" Nielsen Norman Group. Accessed 28.01.2016 through https://www.nngroup.com/articles/persona-budgets.

Google Drive. Accessed 18.01.2016 through https://drive.google.com.

Grudin, J., and Poltrock, S. N.d. "Computer Supported Cooperative Work" in The Encyclopedia of Human-Computer Interaction, 2nd Ed. Accessed 03.02.2016 through https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/computer-supported-cooperative-work.

Hansen, R. S. 2013. "Have Applicant Tracking Systems (ATS) Ruined Recruiting, Hiring, and Job Search?" LiveCareer. Accessed 21.02.2016 through https://www.livecareer.com/quintessential/applicant-tracking-systems-report.

ISO. 2016. "Ergonomics of human-system interaction -- Part 1: Introduction to the ISO 9241 series." Accessed 02.02.2016 through http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=55486.

Jobvite. Accessed 06.02.2016 through http://www.jobvite.com/products/jobvite-hire.

Jobylon. Accessed 04.02.2016 through https://emp.jobylon.com.

LinkedIn. 2015. "A Brief History of LinkedIn." Accessed 07.02.2016 through https://ourstory.linkedin.com.

Mobile Patterns. Accessed 04.04.2016 through http://www.mobile-patterns.com.

Nielsen, J. 2010. "Interviewing Users." Nielsen Norman Group. Accessed 21.01.2016 through https://www.nngroup.com/articles/interviewing-users.

Nielsen, J. 2012. "A/B Testing, Usability Engineering, Radical Innovation: What Pays Best?"

Nielsen Norman Group. Accessed 10.02.2016 through https://www.nngroup.com/articles/ab-testing-usability-engineering.

Pttrns. Accessed 04.04.2016 through http://pttrns.com.

Recruiter Box. Accessed 08.02.2016 through http://recruiterbox.com/recruitment-software-features.

Sigmund. N.d. "The History of Recruitment: Recruiting then and now". Accessed 03.03.2016 through http://www.sigmund.eu/en/tests/home/news/2013/09/06/the-history-of-recruitment-recruiting-then-and-now.

Smith, J. N.d."Desktop Applications Vs. Web Applications". Streetdirectory.com. Accessed: 10.02.2016 through http://www.streetdirectory.com/travel_guide/114448/programming/desktop_applications_vs_web_applications.html.

TalentBin. "The History of Recruiting Technology". Accessed 03.02.2016 through http://marketing.talentbin.com/acton/attachment/3800/f-0008/1/-/-/-/-/History%20of%20Recruiting%20Technology%20Infographic.pdf

Teamtailor. Accessed 05.02.2016 through http://www.teamtailor.com/en/product.

Trello [1]. Accessed 19.02.2016 through https://trello.com/tour.

Trello [2]. Accessed 19.02.2016 through http://help.trello.com/article/808-searching-for-cards-all-boards.

Warren, S. Style Tiles. Accessed 29.02.2016 through http://styletil.es.

Welie.com. Patters in Interaction Design.

Wikipedia. Last modified 07.01.2016. Accessed 15.02.2016 through https://en.wikipedia.org/wiki/Wikipedia.

Workable. Accessed 12.02.2016 through https://workable.com/features-overview.

WordPress. Accessed 19.02.2016 through https://wordpress.org.

Wrike. Accessed 19.02.2016 through https://www.wrike.com/tour/worksmart.

Zuberi, B., and Waleed, C. 2014. Web UI Design Patterns 2014. UXPin.

Appendix 1: Interview Script

Interview script with questions for the recruiters at Software Skills

General

What is your position at Software Skills?
What is your educational and work background?
Why are you interested in working as a recruiter?
Why did you choose this career?

What is your favorite part of the job?
What tasks do you tackle first?
What do you usually procrastinate on? What is your least favourite task?
What makes you feel like you have achieved something?
(conducting an interview, presenting the potential candidates to the client company, one of your candidates being hired?)
What makes a bad day for you?
What helps you make decisions?

Workflow

What was the first thing you did when you came in this morning? What did you do after that?
Can you describe your daily routine tasks?
What is an unusual event or task, that you only do rarely?
How do you conduct your interviews?
What do you do to prepare for an interview?
How do you evaluate your applicants?
- What do you look for online vs. at a meeting?
- What is most important, after fulfilling the requirements?
- How do you process the information in the CV?
- How much does the CV mean when you have all these test etc.?
- Do you ever hire a person, who does not do well on a test or fulfill the requirements?
- How important is the "human-evaluation" in terms of personality, work-ethics?
- Do you read cover-letters? What do you look for there?

How do you currently establish the ranking of candidates?
- On what grounds?

Is it common that you headhunt candidates?

Do you often look up candidates on Facebook, LinkedIn etc?
What are you looking for there?

On which aspects do you collaborate on with the other recruiters in the office? And how do you do it? (within the system, verbally, etc.)
What are the current problems with that? What do you feel is lacking here?

ATS Specific

Could you please describe your current workflow with Trello and Google sheets?
What is the most tedious task/annoying task that you do here?
What is the task you most commonly do?

Have you used other applicant tracking systems before?
Which one(s)?
What did you like/dislike about it?

Could you describe the features you appreciate most with Trello?
Why do you like them?
How do you use them?
What annoys you most with the current system?

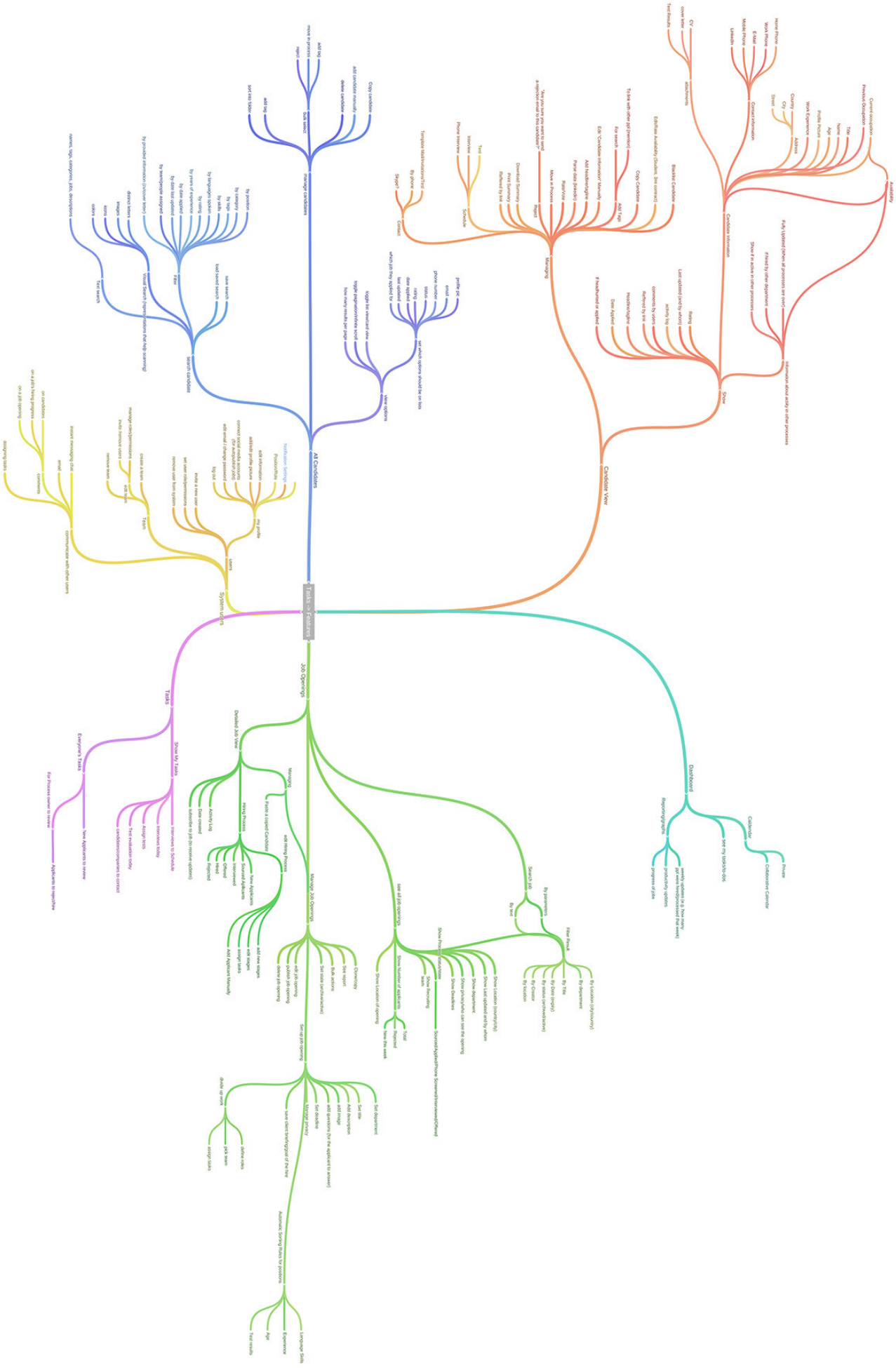
How do you sort old Trello boards to find candidates etc.?
Do you ever remove old ones?

Describe the features you appreciate most about Google Sheets.
What value do they bring and how do you use them?
What do you like best about the functionality?
What annoys you most with the current system?
How often do you use Google Drive?
What kind of files do you access there?

Are there impossible cases, like things you have to tedious workarounds for?

Do you see any ethical issues regarding information or other with the current system?

Appendix 2: Requirements Mind Map



Appendix 3: Usability Test Scenario

Script with scenario use for usability testing.

Intro
Thank you for taking part in our short usability test. Just to remind you, this is to give us feedback on the prototype. We are not testing you, so there is no way for you to be wrong or make a mistake.

Also, can we ask you to think-aloud? This means that you just narrate what you are thinking.
For example, I see this grey box, I'm not sure what it does, but I'm gonna try to click on it… and so on.

Keep in mind that this is a click-prototype, so not all buttons and functionalities work. But it is still interesting to see which elements in the interface you would want to use.

General
What do you see here? Where in the application do you think you are (which view)?
Can you go around the screen and name the different elements you see?

Testing Scenario
You are on the phone and are told a candidate's name, now you quickly want to look up that name, but you don't know which stage he or she is in or any other information. How would you do this without using the search box?

Can you find the candidate Angelina Jonsson (without using the filter or search functionalities)?

Can you move this candidate to the Reviewed stage?

As you moved your candidate to Reviewed, you want to compare her to other candidates in this stage, try to access the list view of the Review stage.

Now go back to the column view and move Angelina Jonsson back to the New stage.

What are your general thoughts on how the information is structured? Does it make sense to you?
Is there anything that feels weird?
Do you have any suggestions for us?

Appendix 4: Excluded Patterns

The following are a selection of patterns that were close to being included in the final pattern collection. They may still be relevant for consideration in other cases of developing an ATS or collaborative web application. All of them are adapted in a very summarized form from Schümmer and Lukosch (2007) and the corresponding page numbers are referenced respectively.

Active Neighbours

(Schümmer and Lukosch 2007, p. 332)

Problem

Not knowing which artifacts other users are working right now.

Context

Apply this pattern when it is relevant to know where others are active in the system.

Solution

Provide indicators of who's working on any artifact encountered in the system.

Why this pattern did not make it into our pattern collection:

Using this pattern induces a state of hyper-awareness that is not needed in the system that we built. It would instead be likely to lead to information overload and consequently to distraction. It would also raise the stress level of users, as they would feel under surveillance as they work and would contradict the concept of openness and personal responsibility. Its less intrusive brethren *Floor Control* addresses the issue of conflicting work done in collaborative spaces, without the hyper-awareness issue.

Activity Indicator

(Schümmer and Lukosch 2007, p.363)

Problem

Not knowing what other users are working on during a collaborative session.

Context

Apply this pattern when it is important to know what exactly the other user(s) is/are doing at a given time.

Solution

Provide information of the current activities of the users.

Why this pattern did not make it into our pattern collection:

Using this pattern induces a state of hyper-awareness that is not needed in the system that we built. It would instead be likely to lead to information overload and consequently to distraction. It would also raise the stress level of users, as they would feel under surveillance as they work and would contradict the concept of openness and personal responsibility.

Remote Selection

(Schümmer and Lukosch 2007, p. 348)

Problem

Multiple users work on the same artifact at the same time, which leads to coordination problems.

Context

Apply this pattern when there is more than one user active on a given artifact at the same time.

Solution

Only allow one user to access a certain artifact at a time.

Why this pattern did not make it into our pattern collection

The functionality prevents collaboration and efficient work progress since important artifacts in the system are locked for use. The problem is already addressed by the *Floor Control* pattern.

Message Read Indicator

(Schümmer and Lukosch 2007, p. 271)

Problem

Not knowing whether information has been received by other users.

Context

Apply this pattern when it is problematic that users ignore invitations, messages, or other notifications.

Solution

Provide feedback that shows whether information has been received. For example: "Message seen yesterday 22:56"

Why this pattern did not make it into our pattern collection

This sort of feedback can be misinterpreted and relied upon too much. The stress of always being under surveillance can affect user to not check notifications due to fear of being misunderstood. Also, in the Honeycomb ATS we built some of the messaging functionality is tied to external mail service providers, which will not be able to indicate whether the message has been read, making the feedback lopsided.

Timeline

(Schümmer and Lukosch 2007, p. 377)

Problem

Not knowing who has been involved in a work process over a given time, so it is unclear who is actually available for collaboration there.

Context

Apply this pattern when there are many users collaborating asynchronously, which causes artifacts to change frequently.

Solution

Show the changes that have occurred on a specific artifact over time and who is responsible in a timeline format.

Why this pattern did not make it into our pattern collection

This problem is already covered to a large extent by the *Contextual Activity Log* pattern.

Periodic Report

(Schümmer and Lukosch 2007, p. 383)

Problem

"Changes in indirect collaboration are only visible by inspecting a changed artifact. Users want to react to actions on artifacts but they cannot predict when these actions will take place." (Schümmer and Lukosch 2007, p. 383)

Context

Apply this pattern when there are many users collaborating asynchronously, which causes artifacts to change frequently.

Solution

Send a periodic report to users so they are regularly informed about changes on specific artifacts. This can also include statistics on the system as a whole and can be utilized and expanded on according to business goals.

Why this pattern did not make it into our pattern collection

In Honeycomb ATS, the users are informed of changes and statistics through the *Contextual Activity Log* and other items of the *Interaction Springboard*. Also, what is the periodic report may be very redundant, because as major work tool, the users are active in the system most of the working day and will know about most of these changes already.

Flag

(Schümmer and Lukosch 2007, p. 287)

Problem

There is a large number of artifacts in the system and it is difficult to distinguish them.

Context

Apply this pattern when artifacts and their value in the system need to be sorted and communicated.

Solution

Provide a function to visibly mark important content so that other users can easily see it.

Why this pattern did not make it into our pattern collection

We turned this bookmarking into a personal function (bookmark is only visible to that one user), so it has nothing to do with awareness or collaboration anymore. But others may consider to include this pattern as a system-wide or team-wide functionality. Also, this functionality can be addressed by the mentions feature of *Shared Annotations* pattern.