UNIVERSITÀ DEGLI STUDI
DI MILANO

SCUOLA DI DOTTORATO IN INFORMATICA
CICLO XXIV

Tesi di Dottorato

Hive-Mind Space: A Meta-design Approach for Cultivating and Supporting Collaborative Design

Li Zhu

Relatore: Prof. Stefano VALTOLINA
Correlatore: Prof. Ernesto DAMIANI

Il Direttore della Scuola di Dottorato in Informatica: Prof. Ernesto Damiani
Anno Accademico 2011/2012

Acknowledgements

Working on a PhD is a lonely endeavor. It is, however, not possible without help and courage from others. My research was supported by the DESIRE network, which allowed me to work in a multidisciplinary field and to make choices that would not have been possible otherwise. I would like to thank Prof. Piero Mussio, who gave me the opportunity to pursue this journey. I am grateful to Prof. Stefano Valtolina for looking after the DESIRE project and especially for helping me with my first design study and to Prof. Ernesto Damiani for giving me the opportunity to work with his team. I enjoyed working with Prof. Paolo Ceravolo, Fulvio Frati, Isabella Seeber and Gabriela Waldhart on the Aristotele project.

I would like to thank Prof. Gerhard Fischer and all the researchers at L3D Centre. In particular I would like to thank Prof. Fischer, who has always been extremely generous in sharing his insightful thoughts, valuable resources, and access to the scientific communities I wanted to belong to as well as asking questions that forced me to define and defend my choices. I am deeply grateful to Ingrid, Stefanie, and Eva for making me feel at home, far away from home. Thanks to Jason Zietz, Holger Dick, and Hal Eden for inspiring meta-design discussions about their energy project.

I would like to thank Prof. Thomas Herrmann for inviting me to work with his group, providing valuable feedback, and engaging in constructive discussion. He has been very supportive and patient in providing guidance and advice, which has significantly reshaped my thesis. Many thanks to IMTM researchers Alexander Nolte, Marc Turnwald, Michael Ksoll, Jan Nierhoff, Martin Degeling, Rainer Skrotzki, Nina Sendt, Isabel Schaller, Yvonne Borowiak, Nadja Nikulin, and Dennis Liedtke for participating in my experiments and the wonderful company they offered me.

A special thank you goes to Prof. Anders Mørch, who made valuable contributions to my thinking about the ideas described here and to how to present them, as well as spotting the value of my thesis more than I myself had. His ‘application units’ provided a valuable starting point for my own work. I would like to thank Prof. Maria Francesca Costabile and Antonio Piccinno. Without their ‘software shaping workshop’ methodology, my work would not have been possible. Thanks, too, to Barbara Rita Barricelli, who was very kind in helping me with my research and numerous, tedious bureaucratic issues over the past several years.

I am thankful to those who helped me through the inevitable gloomy moments: Harpreet Brar, Nadia EL-Imam, Alberto Cottica, Winnie Luo, Layda Gongora, Chunhui Zhang, Rocio von Jungenfeld, Lian Xue, Matthew Heins, Navid Ahmadi, and all my friends who believe in me and love me for who I am. Many thanks go to David Ryan for his help in amending my English and for being such a great friend.

I would like to express my gratitude to my parents for being there and supporting me to pursue my passion.

Finally I would like to thank Ivan, for his love, his trust, and believing in me more than I do. He has challenged my thoughts, put up with my chronic PhD depression, and supported me both emotionally and intellectually. I couldn't possibly ask for more. Thank you.

Table of Contents

Abstract

The ever-growing complexity of design projects requires more knowledge than any individual can have and, therefore, needs the active engagement of all stakeholders in the design process. Collaborative design exploits synergies from multidisciplinary communities, encourages divergent thinking, and enhances social creativity.

The research documented in this thesis supports and deepens the understanding of collaborative design in two dimensions: (1) It developed and evaluated socio-technical systems to support collaborative design projects; and (2) It defined and explored *a meta-design framework* focused on how these systems enable users, as active contributors, to modify and further develop them.

The research is grounded in and simultaneously extends the following major dimensions of meta-design: (1) It exploits the contributions of social media and web 2.0 as innovative information technologies; (2) It facilitates the shift from consumer cultures to cultures of participation; (3) It fosters social creativity by harnessing contributions that occur in cultures of participation; (4) It empowers end-users to be active designers involved in creating situated solutions. In a world where change is the norm, meta-design is a necessity rather than a luxury because it is impossible to design software systems at design time for problems that occur only at use time. The co-evolution of systems and users' social practices pursued in this thesis requires a software environment that can evolve and be tailored continuously.

End-user development explores tools and methods to support end users who tailor software artifacts. However, it addresses this objective primarily from a technical perspective and focuses mainly on tailorability. This thesis, centered on meta-design, extends end-user development by creating social conditions and design processes for broad participation in design activities both at design time and at use time. It builds on previous research into meta-design that has provided a strategic overview of design opportunities and principles. And it addresses some shortcomings of meta-design, such as the lack of guidelines for building concrete meta-design environments that can be assessed by empirical evaluation.

Given the goal of this research, **to explore meta-design approaches for cultivating and supporting collaborative design**, the overarching research question guiding this work is:

> *How do we provide a socio-technical environment to bring multidisciplinary design communities together to foster creativity, collaboration, and design evolution?*

To answer this question, my research was carried out through four different phases: (1) synthesizing concepts, models, and theories; (2) framing conceptual models; (3) developing several systems in specific application areas; and (4) conducting empirical evaluation studies.

The *main contributions* of this research are:

- The ***Hive-Mind Space*** model, a meta-design framework derived from the "software shaping workshop" methodology and that integrates the "seeding, evolutionary growth, reseeding" model. The bottom-up approach inherent in this framework breaks down static social structures so as to support richer ecologies of participation. It provides the means for structuring communication and appropriation. The model's open mediation mechanism tackles unanticipated communication gaps among different design communities.
- ***MikiWiki***, a structured programmable wiki I developed to demonstrate how the hive-mind space model can be implemented as a practical platform that benefits users and how its features and values can be specified so as to be empirically observable and assessable;
- ***Empirical insights***, such as those based on applying MikiWiki to different collaborative design studies, provide evidence that different phases of meta-design represent different modes rather than discrete levels.

Chapter 1

1. Introduction

1.1 Research Context

The ever-growing complexity of design projects requires more knowledge than any individual can have, and therefore calls for actively engaging all stakeholders, problem owners, and developers in the design process (Fischer 2000; Costabile et al. 2008a). Collaborative design exploits synergies from multidisciplinary communities, encourages divergent thinking, and enhances social creativity. In my research, collaborative design is two-fold: (1) developing socio-technical systems (Rice 1958; Trist 1981; Fischer and Herrmann 2011) and (2) conducting collaborative projects supported by socio-technical systems.

1.1.1 Cultures of Participation – Hive Mind

Social media, web 2.0, and advanced information technologies have profoundly changed the way that knowledge is created and distributed, thus transforming users from information consumers to content producers (Benkler 2006; Bruns 2008) and encouraging cultures of participation (Jenkins 2006; Fischer 2009a).

Cultures of participation exist not only by virtue of new technologies but also because of a fundamental mindset shift. The key notion of cultures of participation is the notion of empowerment that comes along with do-it-yourself, open source, and a sharing culture. Cultures of participation democratize design and innovation (von Hippel 2005) by shifting power and control to users, helping them act as both designers and consumers (Tapscott and Williams 2006). In recent years, wikis, social tagging, and media-sharing platforms (Wikipedia, Delicious, Flickr, YouTube, and so forth) have been geared towards creating user-generated content mechanisms, as well as collaborative design environments.

The concepts of "*produser*" (Bruns 2008) and "*prosumer*" (Tapscott and Williams 2006) highlight users' having become producers of the shared knowledge base, acting as both producers and consumers. With new information technologies, users become much more actively involved in shaping their own media and network usage (Shirky 2010). Herz

describes new-media-enabled cultures of participation, in analogy to social insects, as the "hive mind" (Herz 2002). A networked *hive mind* is a distributed but coordinated community, organized not according to the directions of a central authority, to which all other nodes in the network are subordinate, but according to the community's own protocols of interaction (Bruns 2008). The hive mind, harnessing little quanta of intelligence from a large number of internet-connected users, transcends individual minds (Kelly 1994).

1.1.2 Social Creativity

The knowledge associated with dynamic and complex collaborative design problems is tacitly distributed among the various design communities (Rittel and Webber 1984; Fischer 1999b; Fischer 2000). Therefore, in order to foster social creativity, all stakeholders must be involved in solving problems, be given the chance to construct their own understanding and be allowed control over how problems are described. The more creative we are in design, the greater the probability of designing useful and usable software applications and computer systems (Taylor et al. 1958).

All the stakeholders participate in several loosely affiliated groups with shared practices who, together, form Communities of Practice (CoPs) (Wenger 1998). CoPs come together in order to collaborate, bringing their own expertise and background and thus forming a Community of Interest (CoI) (Fischer 2004).

The collaborative design process can be seen as the meeting between CoIs, emphasizing the idea of bringing together different disciplines and perspectives. CoIs provide an example of the importance of combining voices from different communities (Fischer 2001). Bringing divergent viewpoints together helps stakeholders discover new insights and new alternatives, hence solving problems more creatively (Fischer 2001). The centers of creativity tend to be at the intersection of different cultures, where it is easier for individuals to discover new combinations of ideas (Csikszentmihalyi 1996).

1.2 Research Challenges

Challenges to collaborative design come not only from technology *per se*, but also from social and cultural factors, as well as from the difficulty of choosing appropriate research methods.

1.2.1 Communication Gaps

Design teams have members with heterogeneous cultural and professional backgrounds. Design teams from different cultural backgrounds use different systems of signs, languages, and representations. They may have different perceptions, in addition to different interpretations, even for the same images (Petre and Green 1993; Snow 1993).

Communication is crucial in order for them to achieve a common understanding about the messages they exchange. The challenge to collaboration consists of bridging the

communication gaps among them, enhancing mutual development (Andersen and Mørch 2009; Mørch and Andersen 2010), and supporting emergent practices and requirements during the collaborative design process. Current design projects are especially challenging, as new media and new types of messages enter the field, which affects the nature of collaboration in computer-mediated information systems.

The term 'social-technical gap' describes "*the great divide between what we know we must support socially and what we can support technically*" (Ackerman 2000). Technical systems are rigidly predefined and resist change, whereas social practices are fluid, open to interpretation, and constantly evolving (Hewitt 1986; Bentley 1992; Grudin 1994b; Fischer and Herrmann 2011). Because of the social-technical gap, computer-mediated collaborative systems only rarely provide technical support that continuously matches users' creative needs.

1.2.2 Co-evolution

From a design perspective, creative design is not about first fixing the problem and then searching for a satisfactory solution concept. Rather, it seeks to develop and refine the formulation both of the problem and of ideas for how to solve it by continuously iterating processes of analysis and evaluation between the two 'spaces' – problem and solution (Dorst and Cross 2001). As such, problem space and solution space co-evolve. In practice, the tension between the goal-oriented management of design projects and the dynamic nature of design requires an evolutionary approach to tackling wicked design problems (Rittel and Webber 1973). Hence, it challenges traditional software system design.

From a software-development perspective, users and systems have to have an open environment that supports any unforeseen needs (Nielsen 1993; Bourguin et al. 2001). Future uses and future problems cannot be completely anticipated at the time systems are designed. Only at use-time will users discover a gap between their needs and what the existing system supports (Fischer and Giaccardi 2006). Dealing with this uncertainty requires constant improvement and adjustments.

Therefore, interactive systems for collaborative design must be planned so they evolve along with the collaborative practices the systems support. To meet this challenge, the software environment has to support open development that allows users to create their own situated solutions. This co-evolution of systems and practices for collaboration is something that developers and users of computer-mediated collaborative systems are still striving to achieve (Rogers 1994; Dourish 1995; O'Day et al. 1996; Andriessen et al. 2003).

1.3 Research Aim and Questions

In response to the co-evolution challenge, end-user development (EUD) explores tools and methods to support end users to tailor software artifacts (Costabile et al. 2003b; Lieberman et

al. 2006). EUD allows users to tailor software artifacts with different granularity (Mørch 1997) and to change the systems/environment at use time according to their own needs. Nevertheless it addresses this objective primarily from a technical perspective and mainly focuses on tailorability (Pipek 2005).

Meta-design (Fischer and Scharff 2000b; Fischer et al. 2004a; Fischer and Giaccardi 2006; Fischer 2010), an evolving conceptual framework, aims to create social conditions and design processes for broad participation in design activities both at design time and at use time. Meta-design is therefore oriented to fulfill not only software artifacts but also social contexts that allow different stakeholders to participate in the design process (Youngblood 1986; Fischer et al. 2004a). As such, software engineers do not design the final application as in traditional design. Instead, they create software environments through which different stakeholders can contribute to the design of the final application (Costabile et al. 2007a). Although meta-design provides a strategic overview of design opportunities and principles, specific guidelines are needed for building concrete meta-design environments that can be assessed by empirical evaluation.

The aim of my research is to explore meta-design approaches for cultivating and supporting collaborative design. The expression "cultivating" was chosen because it is an apt analogy to carefully planting seeds (Wenger 2002). It implies creating an environment that requires mindful seeding to encourage design communities to participate, to perform meaningful design tasks, and to learn and grow over time.

The research question is:

> *How do we provide a socio-technical environment to bring multidisciplinary design communities together to foster creativity, collaboration, and design evolution?*

This question can be further broken down into two smaller questions:

- What essential features of a socio-technical system support cultures of participation and foster social creativity?
- How can we support evolving design activities, in diverse communities, through the appropriation of design artifacts?

Answering these questions will deepen our understanding of collaborative design in two dimensions: (1) to develop and evaluate socio-technical systems that support collaborative design projects and (2) to define and explore a meta-design framework that focuses on how these systems can be modified and further developed by users who actively contribute to them.

1.4 Research Approach

This methodology of research combines two approaches, one rooted in computer science that provides a system solution to deal with co-evolution, future computational artifacts, the other

rooted in the social sciences that provides a deeper understanding of existing work practices and processes. The main method is from computer science. The social science methods supplement it and do not strive for the rigor with which they are carried out in the social sciences.

Contextual inquiry and participatory design enabled me to gain insights into the object of my research and to observe ways that complex collaborative design can derive from using and repurposing very simple shared tools. It is important to understand that simple 'low-tech' tools, such as colored pencils, paper, and post-it notes can aid complex communication and rich interaction with great flexibility. However, with most collaborative software, achieving such flexibility is a challenge. The need for each software feature to be planned in advance tends to rule out such interactions and such communication.

The hive-mind space (HMS) model (Zhu et al. 2010b) is proposed with the aim of supporting multidisciplinary design teams' collaboration. This model extends the software shaping workshop (SSW) methodology (Costabile et al. 2003b; Costabile et al. 2006b; Costabile et al. 2008a;) and integrates the seeding, evolutionary growth, reseeding (SER) process model (Fischer et al. 1994; Fischer and Ostwald 2002) with a focus on fostering creativity.

The implementation of the meta-design model, MikiWiki (Zhu et al. 2011b), is incremental since new features are constantly added into the system at each step of its development, as well as at use time for various design studies. Employing web service-based development allows users to reuse and modify tools, data, and services within the system. Always in perpetual beta, MikiWiki can be modified by end users and enriched by empirical studies. Reference points of my approach to this research include reflection-in-action (Schön 1983; Schön 1987) and studies of how the meta-design conceptual model may be implemented and can evolve in various collaborative design contexts.

1.5 Research Contributions

My research is grounded in and simultaneously extends the following major dimensions of meta-design: (1) it exploits the contributions of social media and web 2.0 as innovative information technologies; (2) it facilitates the shift from consumer cultures to cultures of participation; (3) it fosters social creativity by harnessing the contributions occurring in cultures of participation; (4) it empowers end-users to be active designers who are involved in creating situated solutions.

The *main contributions* of this research are:

The ***Hive-Mind Space*** model, a meta-design framework rooted in SSW methodology and the SER model. The bottom-up approach inherent to this framework breaks down static social structures to support richer ecologies of participation. It provides the means to structure communication and appropriation, over time. The open mediation mechanism in

the HMS model tackles unforeseen communication gaps among different design communities. Additionally, the model extends the boundary-object concept as a collaboration medium that can be shaped by social interaction and, through meta-design, by unexpected use.

- ***MikiWiki***, a structured programmable wiki, which demonstrates how the HMS model can be implemented as a practical platform that benefits users and how its features and values can be specified so as to be empirically observable and assessable. Moreover, MikiWiki extends the traditional unstructured wiki to a structured programmable wiki that supports open implementation opportunities on the client side. MikiWiki thus not only supports general collaboration but also can evolve at the time of use.
- ***Empirical insights,*** such as those based on applying MikiWiki to varied collaborative design studies, provide evidence that different phases of meta-design represent different modes rather than discrete levels.

1.6 Thesis Structure

To achieve my objective, this research was carried out in four different phases: (1) synthesizing concepts, models and theories; (2) framing conceptual models; (3) developing several systems in specific application areas; and (4) conducting empirical evaluation studies. These phases trace through the following chapters.

Chapter 2, *Theory and Related Research*, offers an overview of relevant literature, concepts and models for approaching collaborative design, and key concepts that inspired this research. Chapter 3, *Research Design and Preliminary Exploration*, presents the methodologies I use to conduct this research. In the second part of this chapter, I provide an overview of a set of preliminary studies that I carried out and the informal findings that led me to research boundary objects and eventually to develop the HMS model.

Chapter 4, *Hive-Mind Space Model*, introduces a meta-design conceptual model derived from SSW methodology. This chapter describes how the model integrates different theories, models, and frameworks. It lays out the reasons for choosing the criteria and the ways each of the HMS model's attributes supports various aspects of creative collaborative design.

Chapter 5, *MikiWiki,* introduces several related concepts: existing wikis, features wikis lack, and how features of the HMS model map onto MikiWiki. Chapter 6, *Architecture and Implementation*, explains the principles of MikiWiki architecture, the reasons behind certain technological choices, and how different features were implemented.

Chapter 7, *Design Case Study-1: Energy Feedback System Mockup Environment Design*, presents a student team project focused on designing an energy related iPad application using a mockup environment built within MikiWiki. The design of the energy feedback mockup environment and its final application interfaces were a joint effort of the Computer Semiotics Laboratory (CSLab) and the Center for Lifelong Learning & Design (L3D) lab in the University

of Colorado. Chapter 8, *Design Case Study-2: Collaborative Writing*, presents part of the European ARISTOTELE project for studying collaborative writing and modeling in virtual teams. Students and researchers from Universita' degli studi di Milano and Universität Innsbruck evolved MikiWiki to support their design activities, share documents, discuss business process models, and finalize system specifications for future software development (ARISTOTELE 2009).

In Chapter 9, *Evaluation*, I describe how MikiWiki was used to redesign a micro-survey tool in collaboration with the Department of Information and Technology Management, Institute of Applied Work Science, Ruhr-University of Bochum. The whole cycle is video recorded. I analyze different levels and iterations of design activity and present insights into how creativity is supported by MikiWiki. The outcome is documented as an empirical, explorative evaluation. Chapter 10, *Conclusions and Future work*, reviews the aim of my research, the HMS model features, MikiWiki as the prototype of the HMS model, pros and cons, and proposes future work.

Appendices

A glossary and all the abbreviations used in this thesis are listed at the end. Appendix A - Nuggets provides instructions for using MikiWiki. Appendix B - MikiWiki API provides a simple API for more advanced users to explore extending MikiWiki and meta-design activities. Appendix C - Publications provides a list of published papers related to this research.

Chapter 2

This chapter provides an overview of relevant literature, concepts and models for approaching collaborative design and some key concepts that influenced this research. The aim of this chapter is to give an overview of the existing body of knowledge and draw synergies from different fields.

2. Theory and Related Research

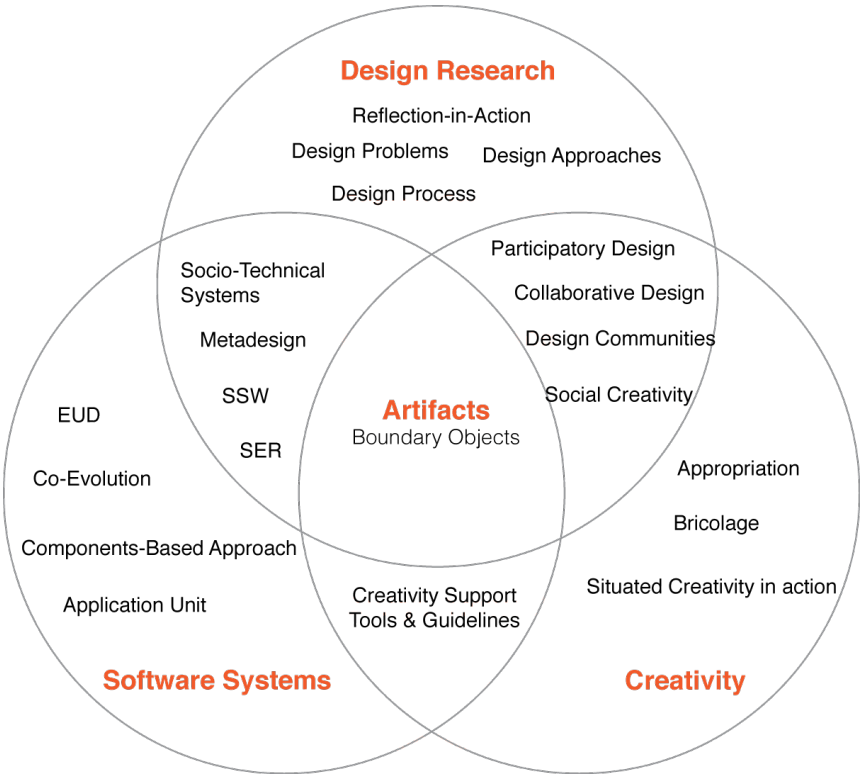


Figure 1 Research dimensions

This research is an interdisciplinary research and rooted in three main research fields - respectively, design research, software development and creativity (Figure 1). Section 1 provides an overview of design, design problems and design process with a focus on

participatory design and collaborative design. Section 2 addresses EUD and meta-design, socio-technical systems and how different models, theories and techniques shape the software building process. The boundary object concept, an important theoretical construct to understand the facilitating role of artifacts in collaborations, is introduced in Section 3. Section 4 introduces creativity-related concepts, and focuses on social creativity. It then emphasizes appropriation and bricolage concepts, which embody creativity from a generic aspect in a specific context: situated creativity in action. Finally, to bring design, software development and creativity together, an overview of design guidelines for a creativity support tool is introduced.

2.1 Design

This research is design-based, aiming at "*research through design*" but also "*design through research*" (Brown 1992; Bærenholdt 2010).

2.1.1 Introduction

One of the most cited design literature books is Herbert Simon's *The Science of the Artificial* in which he defines design as "*courses of action aimed at changing existing situations into preferred ones*" (Simon 1996; Atwood et al. 2002). According to Simon, natural sciences are concerned with how things are, while design is concerned with how things ought to be (Simon 1996).

Lawson's studies of design behavior (in particular, comparing problem-solving strategies of designers with those of scientists) suggest that scientists problem-solve by analysis, while designers problem-solve by synthesis (Lawson 1979). Schön describes design as a reflective conversation that concentrates on the structuring role of the designer, setting the task and outlining possible solutions all in one "framing" action (Schön 1983). "Framing" is reflected in Cross's design definition - applying existing knowledge through a structured methodology to solve an existing problem (Cross 1993).

Design problems are widely recognized as being ill-defined, ill-structured, or "wicked" (Rittel and Webber 1973; Rittel and Webber 1984). This implies that the process of solving problems is a learning process. In contrast to problems that scientists and engineers usually focus on, which are well defined and with all the necessary information, design problems are not subject to exhaustive analysis and there is no guaranteed correct solution (Rittel and Webber 1973; Cross 2006).

However the designers' task is to produce "the solution" in terms of "*a conjectured solution that the problem can be contained with manageable bounds*" (Hillier 1979). Similarly, Simon views design as a process of "*satisficing*" rather than optimizing or producing satisfactory solutions (Simon 1996). Designers intend to seek or impose a primary generator, which both defines the limits of the problem and suggests the nature of its possible solution (Darke

1979). The rise of user-centered design (Norman and Draper 1986) and stakeholder-thinking (van Aken 2007), engaging stakeholders for long-term value creation, have pushed forward collaboration in design projects.

2.1.2 Participatory Design

The movement of participatory design is end-user and design-process oriented. Pelle Ehn describes design as "*a democratic and participatory process*" (Ehn 1993). Participatory design has its roots in the 1970s in Scandinavia, and seeks to involve users in the design process by empowering them to propose and generate design ideas (Nygaard 1986; Floyd 1987; Bodker and Gronbaek 1991; Greenbaum and Kyng 1991; Schuler and Namioka 1993; Muller and Kuhn 1993).

Participatory design has been defined primarily as a method for refining specific technological tools for particular work environments and with significant union involvement. The main strength of participatory design is examining technology in real work practice. In participatory design, end-users are considered experts in doing some specific task – thus bringing their tacit knowledge and skills into the development process.

This suggests a mutual learning process, an important part of participatory design (Bratteteig 1997). Tone Bratteteig uses this notion to accsess the Florence project (Bratteteig 1997; Bjerknes and Bratteteig 1987), in which nurses and designers learn from each other during the design process. The learning deals with knowledge about the application area and the work that the future computer system is supposed to support, the technology itself and possible applications of new technology.

Participatory design exploits different techniques to support communication and collaboration within interdisciplinary teams - for instance, Future Workshop (Bodker and Pedersen 1991), Organizational Games, Mock-up Design and Cooperative Prototyping (Greenbaum and Kyng 1991; Bødker et al. 1993).

Nevertheless, participatory design has focused on the system development at design time rather than extending it to use time (Fischer 2003), as development is carried out by software developers, while end-users are only involved in prototype evaluation (Costabile 2001). The EUD movement aims to tackle this issue, and will be discussed in detail in Section 2.

2.1.3 Collaborative Design

Technology and systems have increasingly become complex and dynamic. Design has thus become a complicated activity, requiring collaboration between various domain, process and technical experts (Fischer 2004). Nunamaker et al. describe several challenges of collaboration and group work ranging from distraction to dominance and information overload (Nunamaker et al. 1996).

Design as a collaborative process has various meanings, such as collaborative, co-operative, concurrent, user-centered, participatory, socio-technical and community design (Grudin 1994a; Verbeke 2001; Scrivener 2001).

Computer supported collaborative design (CSCD) has gained more attention from the research community and industry as a way to address requirements due to increasingly complex product development and high customer expectation (Shen et al. 2008). CSCD, called Cooperative Design, Concurrent Design, or Interdisciplinary Design, is the process of designing a product through collaboration among multidisciplinary stakeholders associated with the entire product lifecycle. A typical design process includes preliminary design, detailed design, manufacturing, assembly, testing, quality control, and product service (Sprow 1992).

Collaborative design is performed by multiple participants, each potentially capable of proposing values for design issues and/or evaluating these choices from their own specific perspective (Klein et al. 2003). Collaborative design can be viewed as a *purposeful joint effort to create a solution*. Collaboration means a co-evolution of the artifact by the stakeholders with a shared understanding of the goals and means of achieving them (Piirainen et al. 2009).

(Popovic 1996) identifies three levels of collaborative design:

(1) Collaboration between designers (designers-to-designers collaboration), which is achieved by sharing design tasks and skills between designers in the same domain;

(2) Collaboration between designers and other participants in the design process, especially professionals from different domains working in large scale projects (Scrivener 2001), often requiring the development of shared understanding, efficient communication and coordination;

(3) Collaboration between designers and end-users, who interact directly with the end-product constructed by the designers.

Synergy from all the levels implies appropriate solutions for conflicts between participants. In collaborative design, reaching an agreement on solutions is not only based on technical problem-solving criteria, but also results from compromises between stakeholders through negotiation and communication.

Collaborative design teams often work in parallel and independently with a wide variety of tools either co-located or distributed across various time zones, aka distributed collaborative design (Verbeke 2001). The rapid development of new communication technologies was supported and driven by the geographical spread of organizations. This led to new computer supported cooperative work technologies. Computer Supported Cooperative Work (CSCW) (Greif 1988; Schmidt 1991; Miller et al. 1992; Grudin 1994a) has been an active research field during the past two decades. The most widely used CSCW techniques in collaborative design systems include *groupware* techniques for supporting communication among design teams

and context awareness techniques for enhancing coordination among them (Piirainen et al. 2009).

Web 2.0 (O'Reilly 2006) has emerged to support social computing (Kellogg 2007). It focuses on collaborative design environments, social media, and social networks creating feasibility spaces for new cultures that allow people to participate rather than being confined to passive consumer roles (Brown and Campione 1994). The participative culture of web 2.0 is organized bottom-up, based on coincidence, free will and spontaneous contribution (Herrmann 2008).

On the other hand, design collaboration requires a higher sense of working together in order to achieve a holistic creative result. It requires cooperation, coordination and collaboration (Kvan 2000). Kvan argues that "*working together, even effectively, is not necessarily collaboration nor should it be",* as collaboration is time consuming and requires a greater commitment to a common goal (Mattessich and Monsey 1992) than co-operation or compromising.

Despite much research in CSCW focusing on standardization and standards for understanding how complex collaborations create information systems, it is important to study the improvisation that is necessary to innovate in collaborative design in general (Lee 2007).

2.1.4 Design Research

Design research has been gaining more popularity in recent years. The term was first introduced as a new methodology "*for carrying out formative research to test and refine educational design based on principles derived from prior research"* (Brown 1992; Collins 1992; Collins et al. 2004). Buchanan states that the twentieth century brought a new agenda for design research. The importance of design research, as John Thackara pointed out, is that *"[m]any of the troubling situations in our world are the result of design decisions […] if we can design our way into difficulty, we can design our way out*." (Simon 1996; Thackara 2005).

Various design research approaches attempt to extend the boundary of collaborative design both theoretically and practically:

- Participatory Design, as mentioned above;
- Ethnographic Fieldwork for the (re)design and evaluation of information communication technology applications from computer supported cooperative work (CSCW) (Randall et al. 2007; Crabtree 2004; Crabtree 2003; Nardi 1997; Grudin 1994a);
- Lead User approach (von Hippel 2005) to invite lead users to help researchers and designers to jointly improve existing products or to develop new products;
- Empathic Design from a business-like approach (Leonard and Rayport 1997) to a more creativity-like approach (Koskinen and Battarbee 2003);
- Open Source Design in technology (Ghosh 2006);

- Co-designing or Co-creation inviting future end-users together with researchers and designers to jointly articulate ideas, playing with concepts, evaluating sketches and tinkering with prototypes (Giaccardi 2004; Sanders 2000; Sanders and Stappers 2008).

Different forms of collaboration aim to create not only products and technologies, but also support for creative appropriation, new processes and services. At a result, the boundaries between designers, engineers, and users, as well as different disciplines have become blurred (Bærenholdt 2010).

In summary: From a design perspective, as design problems are "wicked" (Rittel and Webber 1973), they are not subject to exhaustive analysis, and have no guaranteed correct solution. This research is characterized by progressively refining both theory and design (Collins 1992). It therefore works through incremental designing and design works through research, examining the research theory and design artifacts in practice within social contexts, with a reflection in action manner (Schön 1983). It emphasizes putting design in practice and constantly revising it on the basis of experience. As such, design becomes as a medium and a research process (Bærenholdt 2010).

Cultivating collaborative design pursues engaging all stakeholders, bringing out synergy from all three levels (Popovic 1996), providing opportunities to allow design teams to cope with dynamic design problems and supporting their collaboration in an ongoing manner.

2.2 End-User Development

The research approach of this thesis - to support creative collaborative design - closely follows the principles of EUD and Meta-Design.

2.2.1 Introduction

The emergent paradigm of EUD is based on the vision of HCI evolving from making systems easy to use to making systems that are easy to develop and tailor (Lieberman et al. 2006; Costabile et al. 2006a).

EUD tries to cover the broad spectrum of EUD approaches and addresses the widened scope of new issues. As Spahn et al. (2008) state, the term EUD has evolved over time and complements many other research fields; they consider end-user programming, end-user computing, and end-user tailoring as predecessors of EUD (Spahn 2008).

The idea of enabling end-users to tailor and create software artifacts by programming stems from the field of end-user programming (Fischer and Girgensohn 1990; Cypher 1993; Bell and Lewis 1993; Eisenberg and Fischer 1994; Repenning et al. 1999; Lieberman 2001) and dates back to non-computer science students creating BASIC programs.

The development of applications by end users gained more popularity during the 80s and became a key management and research concern. It is known as end-user computing.

(Leitheiser 1986) define end-user computing as *"the use and/or development of information systems by the principal users of the systems' outputs or by their staffs."* End-user computing is "*defined as the adoption and use of information technology by personnel outside the information systems department to DEVELOP software applications in support of organizational tasks"* (Brancheau and Brown 1993). The best-known examples include spreadsheets, the LOGO "turtle language" for children (Lipsitz and Reisner 1973), and LABVIEW virtual instruments for laboratory automation (Instruments 1983).

End user tailoring as defined by Henderson and Kyng addresses the flexibility of the software applications and approaches that allow users to modify the artifact they use by changing their stable aspects (Henderson and Kyng 1991).

Traditional software development has mainly focused on development from the production perspective and not at use time. EUD calls for a deeper understanding and sophisticated methods for studying the situated development of software in the use context.

To make applications fully usable, end-users often have to adapt them to their specific needs because software developers have not anticipated those needs. There is certainly a growing need for new methods, activities, techniques and tools for end user development (Nardi 1996; Fischer et al. 2004a). The gap between what developers can provide at design time and what end-users really need at use time can be addressed with EUD, at least in principle.

EUD is defined as "*a set of methods, techniques, and tools that allow people who are non-professional software developers at some point to create, modify or extend a software artifact*" (Lieberman et al. 2006). (Repenning and Ioannidou 2006a) point out that EUD is of relevance to potentially large segments of the population, including most end-users of traditional computer applications but also for information technology associated with ubiquitous computing. Therefore, EUD needs to provide answers to adaptation challenges such as the wide range of applications, devices, contexts and user needs.

An important motivation for EUD is that design as a process is tightly coupled to use and it continues during the use of the system (Henderson and Kyng 1991). According to Henderson et al., tailoring a system (that is, continuing designing in use) is an activity different from initial design, which happens before use time. It is an activity related to specific use situations, modifying a system to the tasks on hand rather than creating a new system. The need for providing possibilities for tailoring should be addressed in the initial system design process. (Mørch 2011a) has extended EUD continuous design-in-use to evolutionary application development, further addressing the development of content (rather than merely tools) and professional software developers' activities.

With the emergence of network applications, supporting collaborative activities such as communication, cooperation or knowledge exchange, the need for tailorable software artifacts increased (Bentley and Dourish 1995; Wulf and Rohde 1995). The study of cooperative work

as a socially-situated activity led to a focus on providing mechanisms that reflect on existing work practices, requiring systems to provide a 'medium' that can be tailored to suit each user's need and their work. Several research prototypes have been built as proofs by construction following the principle of radical tailorability (Malone et al. 1992). Important examples for Domain-oriented Design Environment are JANUS (Fischer and Girgensohn 1990; Fischer et al. 1992), Prospero (Dourish 1995), and CoCoWare (Kruse et al. 2000).

EUD represents a shift to interpreting software as a continuous evolving artifact, which considerably contributes to the progress of software development. Today, applications are customizable and extendable (Powell and Moore 2002) and the development of applications by end users is more and more common. Consequently, modern software architectures need to provide more opportunities for tailoring software artifacts in the use context.

2.2.2 End Users

Cypher defines end-users as people who use a computer application as part of their daily life or daily work, but are not interested in computers per se (Cypher 1993).

Nardi and Miller classify end users as:

1) Non-Programmers, who have no or only little programming education and lack an intrinsic interest in computers;

2) Local Developers, who are domain experts and usually have a good knowledge of particular programs;

3) Programmers, who have a good education in computers and therefore a broader technical knowledge than other groups (Nardi and Miller 1990).

Åsand and Mørch define the spectrum of EUD activities as ranging from regular use to professional development (Åsand and Mørch 2006; Mørch 2011a). In this spectrum there are:

1) Regular Users. These are workers who want to use the tools offered by the system in order to perform their tasks, but who are not interested in tailoring the system itself.

2) Super Users. These are domain-trained workers, skilled with computers. They are available to teach other users how to use the system and are interested in exploring meta-tools. They are boundary spanners (Volkoff et al. 2002), gurus (Gantt and Nardi 1992) and translators (Mackay 1990) between regular users and local developers. Local developers, gardeners, brokers, and gatekeepers are all roles assumed by super users (Mørch et al. 2007).

3) Local Developers. These are domain-trained workers, technically more skilled than the 'super users' and with programming knowledge. Other terms, such as *prosumers* (Tapscott and Williams 2006) and *professional amateurs* (Leadbeater and Miller 2004) to refer those

who are both consumers and producers and participate in software development, have been used.

4) Professional Developers. These are IT workers who can develop new software applications or new versions of existing ones. Mørch further refines this spectrum with *customizers* (Mørch 2011a), who use the system and make persistent changes without any programming (Figure 2).

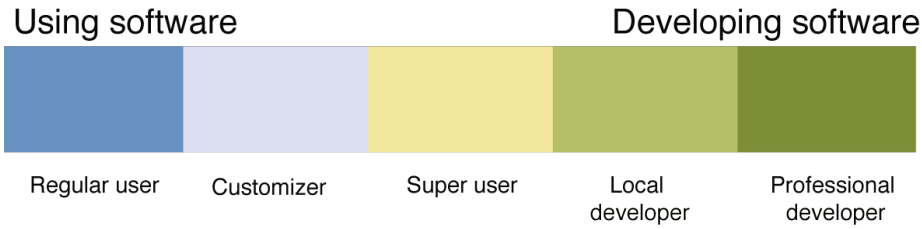


Figure 2 The multiple roles in end-user development. Adapted from (Mørch 2011a)

Ye and Fischer did not classify users into categories, but instead observed a trend that the distinction between users and developers is disappearing. Figure 3 illustrates how pure end-users, who are the owners of the problem, are on the left side of the spectrum, while professional software developers, who build the software systems for the end users, are on the right side the spectrum.

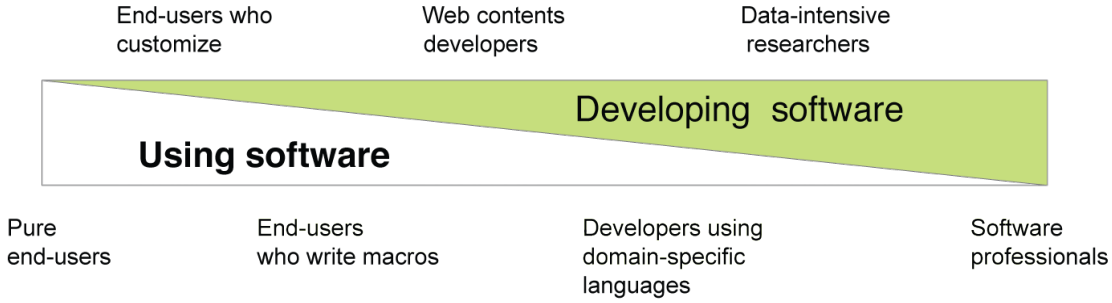


Figure 3 Spectrum of software-related activities. Adapted from (Ye and Fischer 2007)

(Costabile et al. 2008b) further elaborated the research on children as end-user programmers, for whom the goal is not programming but playing, constructing and deconstructing (Petre and Blackwell 2007) and they introduced the notion of the *unwitting software developer (*Figure 4*)*.

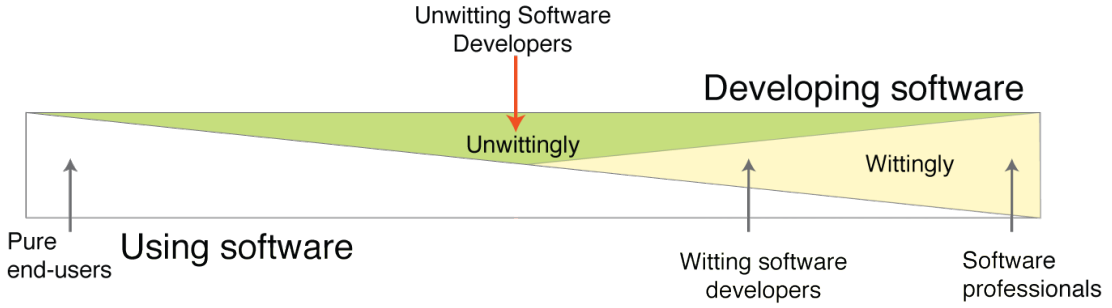


Figure 4 From end-users to software professionals. Adapted from (Costabile et al. 2008b)

Unwitting developers are end users who are not expert in computer science, nor are aiming to be, and use computer systems for their daily work activities. They want easily accessible software environments, which they can tailor to their needs, tasks and habits without being aware of programming.

This complication in defining the role of end-users and professionals is due to various factors. Advanced information technologies foster cultures of participation, ease the barrier of switching between different roles and provide space for new types of users to emerge, thus further tuning EUD activities with a finer level of granularity. All these developments require new observations to redefine the concept of end-user. End users challenge professionals not only by refining their goals of design but also by reacting to the plans they envisioned.

2.2.3 EUD Approach

There are many different approaches in the field of EUD.

(Trigg et al. 1987) define a system as adaptable if it "*enables user-customizable behavior*". They outline four ways that a system can be adaptable: (1) It can have a flexible underlying conceptual model; (2) Its behavior can be parameterized; (3) It can be integrated with other facilities; and (4) It can be tailorable, i.e. users themselves can add new functionality.

Henderson and Kyng define tailoring in terms of modifications made to the subject matter of the tool (so-called "*use*" activities) and modifications to the tool itself ("*tailoring*" activities) (Henderson and Kyng 1991). The tailoring mechanisms may assume many forms ranging from simple customization options by parameter setting to more complex features, such as writing scripts or interactive websites (Sutcliffe and Mehandjiev 2004).

One of the most common classifications is the three tailoring levels: customization, integration, and extension defined by Mørch (1997):

(1) Customization (level 1) allows users to select among a set of predefined configuration options.

(2) Integration (level 2) allows users to add new functionalities to an application by linking together predefined components without accessing the underlying implementation code. It is a form of end-user programming, as end-users can create macros and record script.

(3) Extension (level 3) refers to adding new functionality to applications and textual application components, which however cannot be anticipated by software developers at design time. Extensions include simple extension, complex extension and restructuring patterns.

MacLean et al have observed a large gap between customization (parameter modification) and extension (programming) and between people using corresponding mechanisms (MacLean et al. 1990). This gap however can be bridged by providing a spectrum of tailoring mechanisms, with a smoothly ascending curve of complexity and power (Bentley and Dourish 1995).

(Spahn 2008) organizes EUD approaches by combining the dimensions *complexity of design principle* with the *adaptation power of design principle* to provide a gentle slope of complexity to different user groups.

Complexity / **Adaptation Power**	**Customization**	**Integration**	**Extension**
Programmers			Programming
Local Developers		Component swapping at runtime; Separated tailoring interface	Natural programming; Scripting
Non-Programmers	Interface customization; Parameterization	Programming by demonstration (PBD); Accountants paradigm; Integrated tailoring interfaces	

Table 1 Classification of EUD Approaches (Spahn 2008)

The complexity of design principles is defined based on the required technical knowledge of the end-users to use a tailoring mechanism. The adaptation power dimension is based on Mørch's classification (Mørch 1997) (Table 1).

Interface customization is one of the most widespread EUD approaches and it is comparatively simple. For instance, *Buttons* provides basic modifications, i.e. changing a button's coordination on the desktop or changing labels and icons of a button (MacLean et al. 1990).

Parameterization enables users to adjust some settings via an options menu. For instance the Buttons system provides the possibilities of influencing a button's functionality by setting parameters, e.g. pressing a button executes a particular action (MacLean et al. 1990).

Programming by demonstration involves scripting. For instance, writing a macro enables end-users to record actions for the automation of recurring tasks. Both *Watch What I Do: Programming by Demonstration* (Cypher 1993) and *Your Wish is My Command: Programming by Example* (Lieberman 2001) provide a comprehensive overview of this approach.

Accountants paradigm utilizes the self-explanatory characteristic of a tabular data representation to support unwitting programming (Spahn 2008). For instance, spreadsheet users can write programs by using formulas that connect different data cells easily (Nardi and Miller 1990). This addresses an approach of improving the handling and cognitive perception of information.

Integrated tailoring interfaces aim to provide a seamless transition between an application's design-time and runtime views. For instance, spreadsheet applications integrate

both views into one without switching between a design-time and a runtime (Nardi and Miller 1990).

Component swapping at runtime enables end users to create tailorable systems by composing pre-defined building blocks. For example *FreEvolve* (Stiemerling 2000) is based on the JavaBeans component model during runtime and allows end-users to adapt their systems during use time. *Yahoo!Pipes* allows users to compose applications by connection services.

Separated tailoring interfaces are used in more complex versions of adaptable systems. In contrast to integrated tailoring interfaces, this approach provides a clear interface separation between design-time and runtime in that it allows the construction of more powerful and specialized adaptation functions (Dittrich et al. 2006). Bentley and Dourish (1995) distinguish between user interface features and deeper system behavior (Dourish 1995). Oppermann and Simm (1994) distinguish between functionality and interface (Oppermann 1994).

Natural programming is a user-centered approach in that it provides programming languages and environments matching the way end-users think about their problems and more natural to them (Myers et al. 2004; Lieberman and Liu 2006). For example, Myers et al. use the HANDS system to demonstrate how children express their ideas in a more natural way (Myers et al. 2004).

Scripting languages are intended not for writing applications from scratch but rather for combining components (Ousterhout 1998). They are generally typeless in order to simplify the task of connecting components, for instance, Visual Basic, JavaScript, Perl and so on. However, embedded scripting languages mostly offer the usage of predefined objects.

2.2.4 Component-Based Software Development

Component-based technologies have emerged as a key element in the development of complex software systems (Hopkins 2000), offering a promising direction for further work in EUD (Mørch et al. 2004a). Hopkins defines components as "*a software component* [that] *is a physical packaging of executable software with a well defined and published interface.*" Similarly, Szyperski defines components as "*a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties."* (Szyperski 2002)

The main criteria in the development of component-based systems are:

- Reusability - reusing existing components to create a more complex system.
- Evolutionary quality - creating a highly componentized system, in which the changes will be localized and can be made to the system with little effect on the remaining components (Hopkins 2000).

As such it implies components that can be easily used, a component model that supports the assembly and interaction of components, and a process and architecture that support component development.

Component-Based Software Development

Mørch et al. introduced Component-Based Software Development (CBSD) into EUD (Mørch et al. 2004a). CBSD *"involves multiple roles. Framework builders create the infrastructure for components to interact; developers identify suitable domains and develop new components for them; application assemblers select domain-specific components and assemble them into applications; and end users employ component-based applications to perform daily tasks."* (Vitharana 2003; Mørch et al. 2004a).

CBSD shifts the focus from new software development to the integration of existing components to carry out new tasks and addresses scalability in coupling, distribution, and multiple platforms (Hopkins 2000). CBSD and EUD have been used in combination for web application development, supporting incremental development and end-users to assemble, deploy and run applications (Ginige et al. 2005).

However, typical CBSD environments are oriented to professional developers. In the context of EUD, there is a big gap between runtime and design time, using an application and modifying it with an integrated development environment. There is a lack of scaffolding support for using and building with software components (Mørch et al. 2004a).

Several approaches have been adapted to bridge the gap between use and programming, for instance by utilizing *different levels of tailoring* to gradually bridge the gap (Mørch 1997). Hence, *customization* is to modify the parameters of existing components, *integration* is to create or modify assemblies of components, and *extension* is to create new components by writing program code. Another technique is *direct activation*, supporting finding a tailoring function when it is needed (Wulf and Golombek 2001). Tailoring becomes a collaborative process between end-users and professional software developers.

Several prototypes were developed to explore how to ease the transition between using an application and tailoring it with different granularity. For instance, *ECHOES*, a collaborative system, allows the end user to tailor a workflow application. *BasicDraw* is a generic individual productivity tool oriented to drawing. Both systems are based on multiple representations and application units (Mørch 1995; Mørch and Mehandjiev 2000).

Multiple representations provide multiple views to focus on different aspects of the application, allowing it to be seen and changed in different representations. Multiple representations not only provide a shared development context among developers and users but also empower users to take in charge of an application system's functionality.

The representations can be code, control flow diagrams, design rationales or documents produced from earlier tailoring processes. As such this provides a better overview of the software artifact, design process and design decisions. Visual experience is a key factor for allowing end-users to familiarize themselves with the manipulation of software artifacts (Wulf et al. 2008). Additionally, domain oriented visual languages support unwitting programming, conveying tacit knowledge (Costabile et al. 2006a, 2007b) and allowing end-users to learn as well as grow into the role of 'casual programmer' (Wulf et al. 2008).

Application Units

Application units break down a complex complete application system, creating cognitive chunks that are easier to comprehend and manage for end users (Mørch and Mehandjiev 2000).

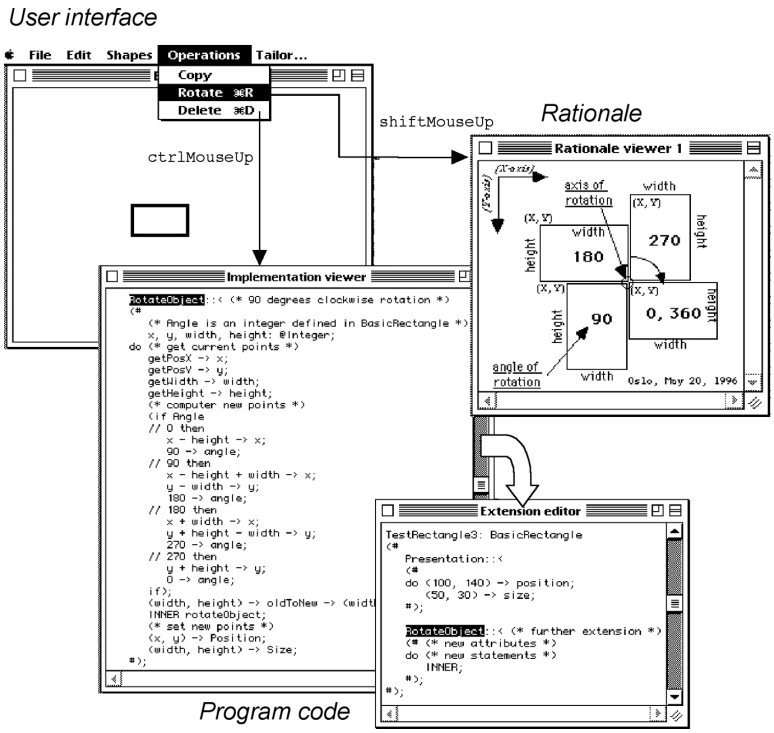


Figure 5 Modifying an application unit at three different levels of abstraction (Mørch and Mehandjiev 2000)

Figure 5 demonstrates an application unit example, integrating three different levels of abstraction - respectively, user interface, rationale, and program code. It shows a Scale application unit for rectangular objects. Users can make changes to each of the aspects. This provides a gentle slope to customization in a drawing program. The modification data is stored in an initialization file and reinstated when the program is started (Mørch 1998; Mørch et al. 2004a).

Application units are defined as the smallest self-contained units to be useful in the design and implementation of end-user tailorable applications, such as word processors, drawing programs, and e-mail systems (Mørch 1995; Mørch and Mehandjiev 2000; Mørch 2003).

Application units are visual components, and they integrate multiple representations as mediators between designers and users, in that each representation presents a different aspect of the system. The goal of the multiple aspects is to simplify tailoring for end users. The interface for the end-user is the convergence point between different multiple representations.

Application units are proposed as a solution to software artifacts' maintenance and reuse, and create generic software applications, addressing end-user tailoring issues (Mørch 1995). One of the key principles behind the application units is to provide a way to minimize the effort involved in understanding complex software applications. Application units are generic, and thus can be reused from one application to another and tailored to specific end-user needs (Mørch 1995).

The "*component approach*" to EUD is different from the "*programming approach*" since end users utilize visual builders to select, modify and connect components using high-level operations rather than writing code in a text editor (Mørch et al. 2007).

Notably, however, components are usually brought into the system by developers. This research approach differs in that it allows end-users themselves to integrate resources in the environment when they need it and during the activity. It addresses several CBSD challenges, for instance providing the application environment without requiring users to write programs or move outside of the use context; and combining different levels of tailoring with existing components (Mørch et al. 2004a).

2.2.5 Meta-Design

Nevertheless, EUD research has been criticized for primarily focusing on tailoring (Pipek 2005), while more complex social aspects need to be taken into consideration (Fischer et al. 2004a; Stevens 2009).

Meta-design is a conceptual framework that aims to create socio-technical environments to empower users to actively engage in the continuous system development process rather than being passive users (Mumford 1987; Costabile et al. 2008b; Fischer and Herrmann 2011).

"Meta-design characterizes activities, processes, and objectives to create new media and environments that allow users to act as designers and be creative." (Fischer and Scharff 2000a; Fischer et al. 2004a)

Fischer and Herrmann suggest five key principles characterizing a meta-design framework for the development of socio-technical systems (Fischer and Herrmann 2011):

- **Cultures of participation**: where several roles and stakeholders can contribute with respect to their interests and find a space for communication and collaboration to exchange their perspectives (Fischer et al. 2001).

- **Empowerment for adaptation and evolution:** by helping end-users or their supporters (software developers, administrators, power-users, facilitators and so on) to modify a software design with respect to their needs (Mørch 1997).
- **Seeding, evolutionary growth and reseeding** (Fischer et al. 2001): Seeds represent basic structures rather than complete and precise ones, leaving space and options for the development of concrete details.
- **Underdesign**: This provides seeds as design opportunities at use time. Representations of solutions (e.g. models or prototypes) do not only include determined specifications but also preliminary, incomplete or imprecise specifications so that designers and end-users are inspired to think about variations or to add further ideas. This aligns with a semi-structured modeling method (Herrmann 2009a).
- **Structuring of communication for "designing the in between"** (Fischer and Giaccardi 2006): Meta-design aims to support not only existing social networks but also shape new ones. It delivers methods of suitable communication support, for instance strategies and methods for conducting participatory workshops, and for facilitating communication among stakeholders with differing perspectives. Walkthrough-oriented facilitation (Herrmann et al. 2007; Herrmann 2009a) as an example supports the integration of various perspectives, the negotiation of design decisions and the integration of knowledge.

Other relevant characteristics of meta-design and its comparison with traditional design are described in (Fischer and Giaccardi 2006), who discuss supporting the evolution of systems that have contingent characteristics from a design perspective.

Meta-design aims to create open systems at design time that can be modified by their users acting as co-designers, requiring and supporting more complex interactions at use time. Meta-design promotes sustainable co-designing by way of seeding and evolutionary growth that is incrementally refined by end-users; it is akin to tending to a software system as if it were a living entity (Fischer et al. 1994).

Meta-design addresses the following features of socio-technical environments:

- They are flexible and evolve because they cannot be completely designed prior to use;
- They evolve to some extent at the hands of the users;
- They are designed for evolution (Fischer and Scharff 2000a).

The new design space underpinned by meta-design consists of three levels (Fischer and Giaccardi 2006).

- **Designing design**: Given the co-evolution issue, this design level addresses underdesign in that it focuses on structures and processes rather than on fixed objects and contents. Meta-designers play an important role in establishing the conditions that empower users to become designers.

- **Designing together**: Designers and users collaborate on the design activity both at design time and at use time, ensuring a full participatory process.
- **Designing the "in-between"**: Defining how co-evolutionary processes and co-creative behaviors can be sustained, e.g. emotional seeding and agency patterning.

This design space aims at integrating (1) a technical infrastructure that is evolvable; (2) a learning and work environment empowering users to be active contributors, and (3) a socio-technical system where users can engage in personally meaningful activities. The three levels are interdependent. A similar three levels of design can be seen in (Costabile et al. 2006a) with a focus on the creation of interaction systems.

2.2.5.1 Seeding, Evolutionary Growth, Reseeding

The Seeding, Evolutionary growth, Reseeding (SER) model is a conceptual process model that describes the development of systems and information repositories that evolve over time (Figure 6) (Fischer et al. 1994; Fischer et al. 2001). This process model helps in designing complex and open-ended systems.

In the seeding process, environment developers and users (domain designers) collaboratively create an initial system that can grow over time. "*A seed is built by customizing the domain-independent design environment architecture to a particular domain through a knowledge construction process"* (Fischer et al. 2001). A seed is explicitly designed to capture design knowledge during use (Girgensohn 1992; Fischer et al. 2001).

According to Fischer et al., *"a seed is a collection of knowledge and procedures capable of growing — of sustaining growth — through interaction with domain designers during day-to-day use. It stimulates, focuses, and mediates discussion — and thus knowledge capture — during the incremental growth phase."*

They point out that a seed does not necessarily have to be complete, correct or with specific information, since an underdesigned seed encourages input from designers. Providing a seed not only engages users to use the system but also encourages meta-design by creating environments that empower users to be active designers through extending, refining and augmenting the existing system (Fischer et al. 2004b).

During the evolutionary growth phase, the users of the application domain use and extend the initial seed according to their needs as each new project contributes new information to the seed. However, users might focus on solving specific problems and creating problem-specific information rather than on creating reusable information. Therefore information added during this phase might not be well integrated with the rest of the information in the seed.

The reseeding phase occurs when the incremental changes in the evolutionary growth phase are no longer practical or stop proceeding smoothly. During this phase, environment developers help domain designers to organize, reformulate, generalize and incorporate

incremental changes from the second phase into the initial seed to support the next cycle of evolutionary growth and reseeding.

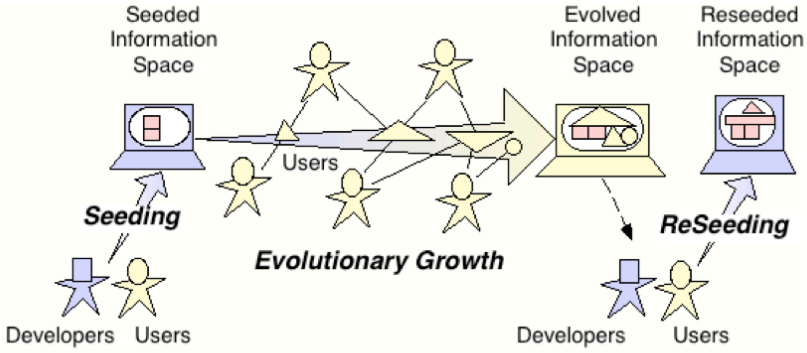


Figure 6 The Seeding, Evolutionary growth, Reseeding model. Adapted from (Fischer et al. 2001)

Cycles of evolutionary growth and reseeding continue as long as developers and designers actively develop and use the system. The reseeding phase implies a centralized process of deliberately selecting and filtering information in order to create a useful information repository.

The SER model suggests that systems that evolve over a sustained time span must continuously alternate between periods of unplanned evolutions and periods of deliberate restructuring and enhancement. It encourages system designers to conceptualize their activity as meta-designers (Fischer and Scharff 2000a). It empowers users as knowledge workers (Drucker 1994) in charge of their own problems, and designers of the system in use (Henderson and Kyng 1991).

The SER model makes a distinction between design time and use time (Andersen and Mørch 2009b). Meta-design aims to bridge these two types of software development activities. As such, meta-designers create socio-technical environments at design time in which other users can be creative in their own domain at use time (Andersen and Mørch 2009b).

The SER model has been applied in various application areas - for instance, in *Envisionment and Discovery Collaboratory* (EDC) (Arias et al. 2000) for urban planning, knowledge is seeds in the EDC environment; in Renga for creating interactive art, seeds are the contents (Anzai 1994); in CodeBroker (Ye 2001), an Open Source Software System, the source code can be viewed as seeds that can be created, evolved and remixed over time.

Andersen and Mørch elaborate on evolutionary growth, reseeding and the dynamic interaction between them to study various stages of software development and mutual development between customers and professional developers (Andersen and Mørch 2009; Mørch and Andersen 2010).

2.2.5.2 Software Shaping Workshop

The Software Shaping Workshop (SSW) is a methodology to support collaborative evolutionary design of interactive systems that support end-users to become designers of their tools (Costabile et al. 2006a; Costabile et al. 2007b; Barricelli et al. 2009b). End users in SSW methodology are domain-expert users (Fischer 1999a) in a specific field (e.g. medicine, geology, mechanical engineering). They are not necessarily experts in computer science, but they are expert users of computer environments to perform activities on a daily basis (Costabile et al. 2003a).

(Costabile et al. 2006a) identify five major phenomena affecting the HCI process and frame them as the SSW methodology-building context:

- User diversity
- The *grain* introduced by tools (Dix 2003) – meaning the mismatch between the tools and the users' mental model
- The communication gap between designers and users (Majhew 1992)
- Implicit information (Mussio 1991) and tacit knowledge (Polanyi 1966)
- Co-evolution of systems and users (Nielsen 1993; Bourguin et al. 2001)

In analogy with artisan workshops, where the artisan finds only essential tools to perform his activities, the SSW consists of various environments for different domain experts. Similarly, domain experts using a virtual workshop find "*only the tools required*" for developing their activities by shaping the software they use. The SSW supports EUD by allowing users to develop software artifacts with "*high-level visual languages*" tailored to their needs. Therefore, the way users work in the software environment is similar to the way they manipulate physical objects in the real world. To this end, the affordances (Norman 1990) provided by workshops are similar to virtual "*habitable environments*" (Alexander et al. 1977; Borchers 2001). Such environments offer their users a "*quality without a name*" that allows them to develop their activities following strategies not prescribed a priori but dictated by the current situation.

In the SSW approach, the system is organized into various workshops (software environments), each one for a specific community. The design and implementation of application workshops is incremental according to the needs and requests of the hosted communities (Costabile 2001).

SSW methodology considers the development of two different kinds of workshops:

- The **application workshop.** This is a workshop customized to each member of the community according to role, culture and device in use.

- The **system workshop.** This is a workshop that allows software developers to design the application workshop to users' preferences, characteristics and needs.

In order to support co-evolution SSW identifies three levels of activity in EUD. Workshops are organized into a three-level network (Figure 7), in which each member of the design team (software engineers, HCI experts and domain experts) collaborates to design and develop virtual environments customized and tailored for their activity domain and tasks to be performed.

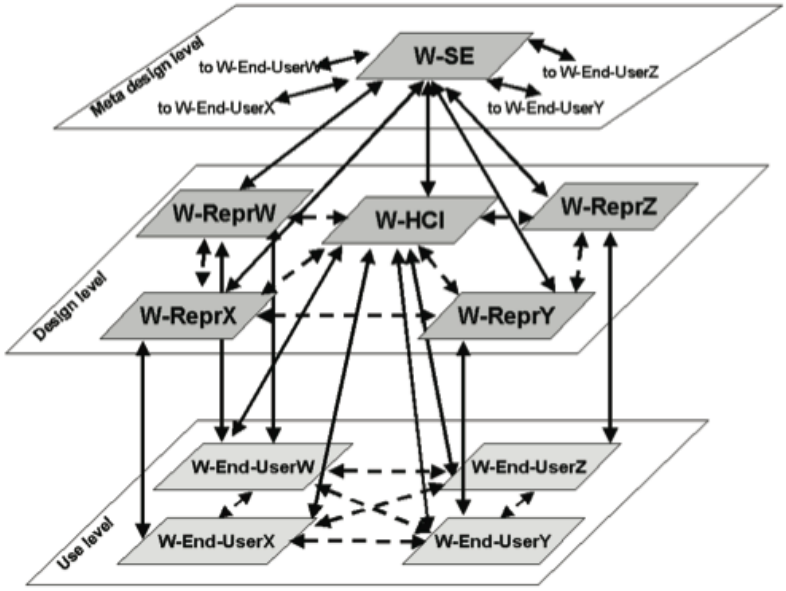


Figure 7 The SSW network (Costabile et al., 2007)

SSW methodology covers all three levels of tailoring - customization, integration and extension (Mørch 1997).

(1) **Meta-design level** (the top level), where software engineers use a system workshop to create other system workshops in order to permit other software engineers, HCI experts and domain experts to collaborate, design and develop application workshops.

(2) At the **design level**, representatives of the end users (domain experts) and HCI experts collaborate using their own system workshops to design and implement application workshops; the domain experts are end-user developers who are in charge of designing and developing systems to be used by other end-users.

(3) At the **use level**, end-users use the application workshops designed and developed at the design level in order to perform their task.

The system and application workshops are first presented as seeds, and are then evolved into new system and application workshops according to users' interaction. However, they remain separate in that a clear distinction between the design and the use level supports users focusing on their activities.

The arrows in Figure 7 represent communication flows between design communities. Dashed arrows indicate the communication paths that exist among the communities that work at the

same level in the SSW network. In this example, at the design level, HCI experts and domain experts exchange their design results to collaboratively develop the application workshops. At the use level, data related to the use activity is exchanged between workshops. The solid arrows indicate lower levels communicating with the higher ones and vice versa. This communication is supported by the use of annotation tools (Fogli et al. 2004) that allow the users to annotate their problems and to send their comments to the experts who are available in the SSW network.

In the context of SSW methodology, meta-design is refined as "*a design paradigm that includes end-users as active members of the design team and provides all the stakeholders in the team with suitable languages and tools to foster their personal and common reasoning about the development of interactive software systems that support end users' work*" (Costabile et al. 2007a). It means that design environments are developed in a collaborative way and provided to end users, empowering them to shape their application environments. End users have two main roles in the lifecycle of designing the interactive software system: performing their working tasks and participating in the development of software environments as stakeholders of the domain. To this end, SSW methodology does not try to differentiate design time and use time, but rather tries to make a distinction between design activities.

Another important aspect of SSW methodology is to overcome the communication gap between designers and users by a gentle slope approach to the design complexity (Costabile et al. 2006a). This focuses on providing personalized environments to all the stakeholders, in terms of language, notation, layout and interaction possibilities. The mediation process and the mediation mechanism are addressed in detail in (Barricelli 2010; Ardito et al. 2011). Barricelli (2010) presents three different axes along which mediation can happen: role, culture and device. These can be seen as three different dimensions of localization (Esselink 2000), adapting an environment to different circumstances. Nevertheless, role, culture and device can be changed through social interaction over time. According to Rommetveit states, every communicative act builds upon the commitment to "*a temporarily shared social world*" (Rommetveit 1974). A temporal dimension (situateness) and dynamic social interaction therefore need to be considered in the mediation mechanism.

SSW methodology has been applied to various interactive software systems, for instance the medical domain (Fogli 2005; Piccinno 2005; Costabile et al. 2006a, 2007b), the mechanical engineering domain (Costabile et al. 2007b), and the tourism and cultural heritage domain (Marcante and Provenza 2008; Barricelli et al. 2009a; Zhu et al. 2010b).

Figure 8 illustrates applying SSW methodology to the medical domain. At the use level, neurologists and neuro-radiologists use application workshops, W-Neurologist and W-NeuroRadio. At the design level, representatives of them (d-experts, e.g. domain-experts) use

two system workshops (W-UserNeu and W-UserRa) to create and maintain the application workshops, and two system workshops (W-CompNeu and W-CompRa) to create and maintain components to be used in the other system workshops for their design activities.

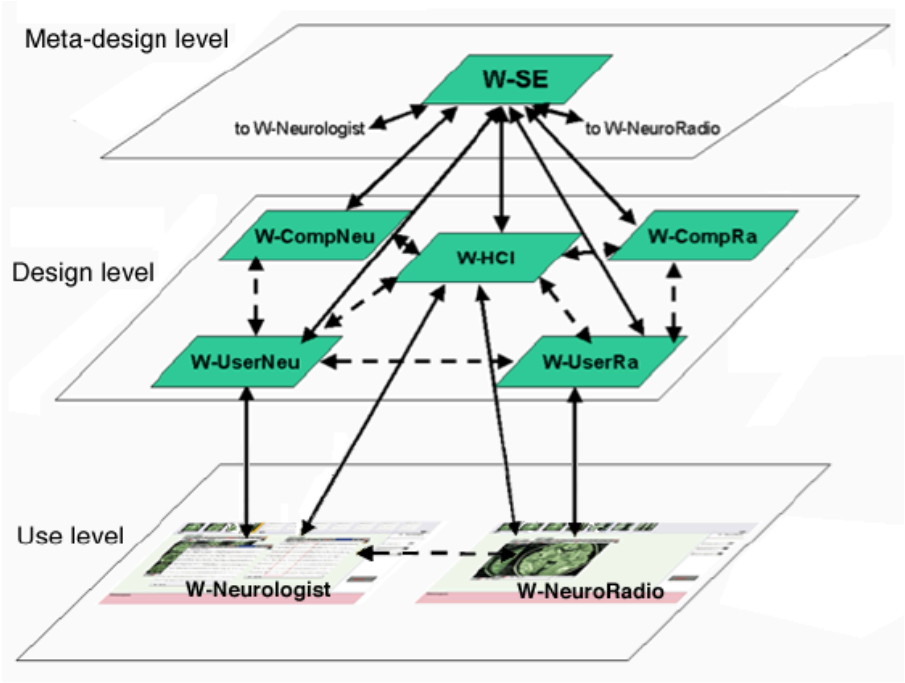


Figure 8 The network of SSWs involved in the medical case. Adapted from (Costabile et al. 2007b)

HCI experts use the W-HCI system workshop to optimize application workshops interaction design. HCI experts and d-experts collaborate with software engineers (who use the system workshop W-SE at the meta-design level) collaboratively create and evolve the two application workshops.

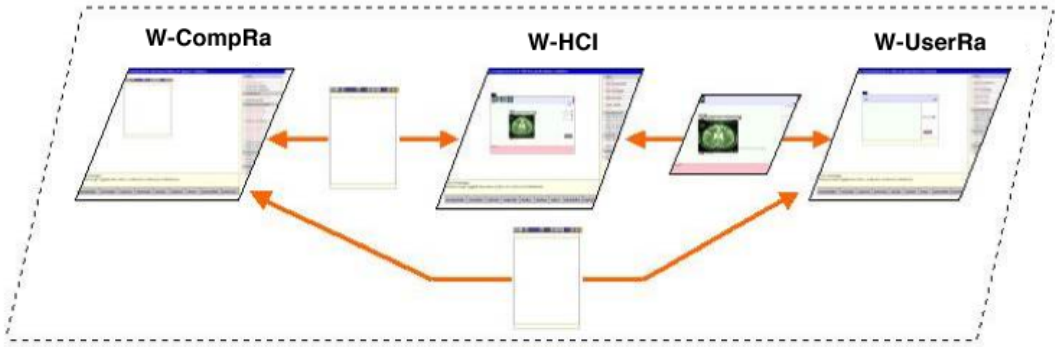


Figure 9 Communications among senior neuro-radiologists and HCI experts at the design level. Adapted from (Costabile et al. 2007b)

Figure 9 illustrates communication among HCI experts and senior neuro-radiologists at the design level. The exchange interface is localized differently according users' role and culture, e.g. W-CompRa, W-HCI, W-UserRa for different users. In the application workshops, end-users can use annotation tools to annotate on the graphic interface to request new features or

report usability issues. The whole creation is a rather complex collaboration process. Each design community can only access its own workshop, thus hard to gain new perspectives or have an overview of the creation process. Furthermore, to address creativity and encourage ad-hoc activities, it requires a more flexible, agile and open system.

In summary: Since the circumstances of our actions are continuously changing around us (Suchman 1985), it is necessary to design systems to accommodate the unforeseeable contingences of situated actions (Suchman 1985). To cope with co-evolution, both users and software systems and both problem space and solution space require software systems to be open and flexible enough for users to appropriate in use, extending and evolving them according to varying skills and needs.

I focus on EUD and meta-design to provide socio-technical environments (Herrmann 2009b; Fischer and Herrmann 2011) in which users with suitable artifacts can pursue collaborative design and be creative. In addition, I adopt SSW methodology from Costabile et al. (2006) and highlight the seeds concept from the SER model of Fischer et al. (2001) embodying seeds as social artifacts with certain attributes. Technique wise, I approach EUD with component based software development, adopting application units from Mørch and Mehandjiev (2000) not only building flexible evolvable software artifacts but also providing different views as mediators among stakeholders as well as different levels of participation and tailorability (Mørch 1997). Complex systems are built out of simple elements in order to be easily evolved during use time.

2.3 Boundary Objects

Boundary object is a term coined by Star and Griesemer (1989). They define boundary objects as objects that are "*both plastic enough to adapt to local needs and constraints of the server parties employing them, yet robust enough to maintain a common identity across sites.*" (Star and Griesemer 1989)

The concept of boundary object has been considered a useful theoretical construct to understand the facilitating role of artifacts in collaborations and to study the interaction among different CoPs. According to Carlile, effective boundary objects have three characteristics (Carlile 2002):

(1) Establishing a shared syntax or language for individuals to represent their knowledge;

(2) Providing a concrete means for individuals to specify and learn about their differences and dependencies across a given boundary;

(3) Facilitating a process whereby individuals can jointly transform their knowledge.

Boundary objects have been broadly studied and used in different fields, for instance engineering sketches (Henderson 1991), standardized reporting forms (Bowker and Star 1994), engineering drawings (Bødker 1998), physical prototypes (Carlile 2002), Gantt charts

(Yakura 2002) and project timelines, as well as more abstract forms such as concepts (Wenger 1998) and definitions (Bechky 2003).

One of the most outstanding and accepted characteristics of boundary objects is that they enable joint activity by acting as common information spaces since they "*inhabit several intersecting social worlds*". Wenger emphasizes the brokering role of boundary objects in collaboration among CoPs (Arias and Fischer 2000). Boundary objects allow multiple perspectives of a single information artifact (Lutters and Ackerman 2002) and have interpretative flexibility (Carlile 2002). Boundary objects are often discussed in the context of negotiation and creation (Lutters and Ackerman 2002; Lee 2005).

Arias and Fischer further suggest that artifacts can serve as boundary objects for two main purposes:

1) To support the interaction and collaboration between different CoPs

2) To serve the interaction between users and computational environments

Arias et al. (Arias et al. 2000) address the importance of design artifacts or 'externalizations' in collaborative design. Boundary objects serve as externalizations that capture distinct domains of human knowledge and hold the potential to lead to an increase in socially shared cognition and practice (Jennings 2005). They carry information and context and can be used to translate, transfer and transform knowledge between design communities (Carlile 2002). Utilizing boundary objects to mediate exchange messages among CoPs is explored in (Ardito et al. 2011). Since the information carried by boundary objects can be implicit, the annotations allow each actor to explicitly explain the modification introduced in the boundary objects.

In a collaborative design context, the evolutionary characteristic of a boundary object and its ability to carry information and context allow different actors to communicate, coordinate and collaborate with each other (Fong et al. 2007). To support the creation and evolution of active boundary objects, it is important to provide systems that can: (1) create awareness of each other's work; (2) afford individual reflection and exploration; (3) enable co-creation (in multiple forms: simultaneous, parallel and serial); (4) allow participants to build on the work of others; and (5) provide mechanisms to help draw out the tacit knowledge and perspectives (Arias and Fischer 2000). These guidelines reveal how boundary objects can be understood through the lenses of meta-design, providing users with the support needed to become designers.

(Engeström and Miettinen 1999) further explore the connection between the role in artifacts for constituting a CoI by integrating the boundary objects into Activity Theory as a tool for "analyzing and transforming networks of culturally heterogeneous activities through dialogue and debate." Two activity systems are the minimal unit of analysis, as the boundary object is common to more than one activity system and brings them together. Co-configuration presents the new understanding of work and production, with the central idea that *"co-configuration work never results in a 'finished' product. Instead, a living, growing network*

develops between customer, product and company" (Victor and Boynton 1998; Engeström 2004). Mørch and Andersen combine co-configuration and meta-design as an integrated model of EUD to study software development and mutual development over time (Andersen and Mørch 2009; Mørch and Andersen 2010).

Boundary objects emerge from communication and meta-negotiation (Lutters and Ackerman 2007), and a set of common or joint activities. Boundary objects are situated and dynamic, in that their validity emerges from how CoPs use them. On the other hand, boundary objects are not just created to support communication, but the structure they build significantly influences dynamic relationships among CoPs. Gal et al. (2004) takes a dynamic approach, examine boundary objects according to the social infrastructure in which they are embedded and to the social identities of the groups that share them. To this end, boundary objects are used not only as a translation device to bridge information and practical gaps between communities, but also as a resource to form and express social identities (Gal et al. 2004). Ehn points out that artifacts modifying the space for interaction are boundary objects in participatory design and infrastructures in meta-design, which can be explored as socio-material frames for controversies, ready for unexpected use, opening up new ways of thinking and behaving. Notably, the same artifacts become infrastructures in the meta-design context (Ehn 2008).

Henderson, however, argues that the flexibility of the boundary objects was "*paralyzed*" by the introduction of "*interlocking*" computer systems and databases, because of their large size and perceived fixity (Henderson 1991). Lee argues that the boundary objects' concept is incomplete in that "*artifacts exist within a web of standardized processes and that disorderly processes are to be treated as special cases*". In addition, she proposed the "*boundary negotiating artifacts*" concept as an amendment to boundary objects (Lee 2005). Subrahmanian et al. point out that organizational structure, information flow and technologies seriously affect the viability of boundary objects (representations, drawings, models – virtual and physical) and call for the synthesis of new common grounds to accommodate the needs of new interfaces (Subrahmanian et al. 2003).

It is certain that in respect to the complexity of collaborative design activities, and the dynamic characteristics of the ongoing design process, the boundary object concept fails to serve as a translation tool and is not enough to negotiate shared understanding. Boundary objects need to be refined in order to continue to satisfy the emergent information and new situations during the collaboration process.

2.3.1 Boundary Zone

Konkola defines a boundary zone as "*an area which resembles a 'no-man's land,' free from prearranged routines or rigid patterns. It is also a place where each activity system reflects its own structure, attitudes, beliefs, norms and roles"* (Konkola 2001).

Konkola proposes the boundary zone as a place where elements from several activity systems are present (Konkola 2001). Activity theorists observe that when two activity systems interact, the shared space has the richest potential for generating new activities that can lead to a transformation of the activity system itself (Engeström 2001; Konkola 2001). Engeström uses "*boundary crossing*" to describe experts engaging in multiple activity systems (Engeström 2001). This emphasizes the learning and transferring process.

The concept of "*trading zone*" coined by Galison also describes a place where participants from different cultures communicate using a stripped-down interlanguage much as traders created pidgins or creoles that operated on the interface between social groups (Galison 1997). He uses "*trading zones*" to express that the interaction is not limited to objects but also includes processes and activities.

A boundary zone is characterized by alternative or competing discourses and points of view that afford opportunities for the transformation of conflicts and tensions into rich zones of learning (Engeström 1999; Engeström 2001). A boundary zone is therefore a hybrid, polycontextual, multi-voiced and multi-scripted context (Tuomi-Gröhn et al. 2003; Mørch and Andersen 2009) describe the boundary zone as an organizational feature where, for example, a development company and a client can exchange improvement requests and software enhancements. As such, CoPs are able to extend their activity and to create shared boundary objects between them.

A common understanding here is that this unique space involves an encounter of two or more perspectives, which not only results in conflicts and debates, but also opens up a new space for thinking, being and acting (Tsui and Wong 2009).

***In summary**:*

In this research, I adapt Star's (1989) notion of the boundary object as a mediating structure between different design communities and I focus on its situated, dynamic and evolutionary characteristics. However, I intend to consider boundary objects (1) as a medium for social interaction within which different content (e.g. text, audio, video and so on) can be both the communication media and the content of the communication and (2) as an infrastructure for unexpected use, opening up new ways of thinking and behaving in the meta-design context. Identifying and designing boundary objects to support participation and social interaction may enable boundary crossing from one community to another and enhance a shared common understanding. The opening up of a rich boundary zone where CoPs or activity systems interact is however not something that can be assumed; it rather needs to be mindfully embedded within the system.

2.4 Creativity

Creativity is an important aspect of this research. It is crucial in solving complex and dynamic problems, and (Taylor et al. 1958) argue that the larger the number of ideas produced, the greater the probability of achieving an effective solution.

2.4.1 Introduction

Creativity has been studied over decades in terms of the creative process (Boden 1994), the creative person (Guilford 1950; Gough 1979) and the creative product (Amabile 1983).

One of the most enduring creativity models in the twentieth century was proposed by Wallas (Wallas 1926; Arieti 1976) and outlines the creative process as involving the stages of Preparation, Incubation, Illumination and Verification. There are many other creative process models, such as Osborn's seven-step model (Osborn 1963) and Amabile's five-step model (Amabile 1983). Nevertheless, they share a common theme, namely purposeful analysis, imaginative idea generation and critical evaluation (Warr and O'Neill 2005).

Shneiderman further extends the creative process model by introducing the social aspect, the *Donate* stage, in which disseminating results might lead to new ideas generated by the reviewing community (Shneiderman 2000). It is noted that the creative process is more recursive than linear (Csikszentmihalyi 1996) .

2.4.2 Social Creativity

"An idea or product that deserves the label creative arises from the synergy of many sources and not only from the mind of a single person." (Csikszentmihalyi 1996)

Creativity does not happen within a person's mind, but in the interaction between a person's thoughts and a socio-cultural context (Csikszentmihalyi 1996; Engeström 2001). Creativity is a central theme in design research, since design is occupied with making things and processes, although there are very different interpretations (Bærenholdt 2010).

Csikszentmihalyi (1996) defines creativity as "*a social construction that is a result of an interaction between the producer and the audience*." (Norman 1993) states that "*the heart of intelligent human performance is not the individual human mind but groups of minds with tools and artifacts*."

(Edmonds et al. 1999) point out that much human creativity arises from activities that take place in a social context, in the process of interaction with other people and the artifacts that embody group knowledge and previous thinking. They suggest that a promising approach to support social creativity might be arranging informal ways for stakeholders to share experiences, in order to articulate their collective knowledge.

Design activities require collaboration among different CoPs and therefore are characterized by a symmetry of ignorance (Rittel 1984), in that no CoP knows better than others and the expertise as well as ignorance is distributed over all participants as a wicked problem.

Exploiting the power of the symmetry of ignorance can lead to intelligent and creative results (Engelbart 1995; Fischer 2000). This can be achieved by providing users with opportunities to construct their own situations and have control in the description of problems.

Additionally, bringing divergent viewpoints together and creating a shared common understanding will certainly help stakeholders to discover new insights and alternatives, and hence to solve problems more creatively (Fischer 2000).

This research explores a meta-design/socio-technical approach to creativity (Fischer et al., 2004). Exploiting the socio-technical approach means that this research addresses the creativity of groups and communities rather than of any specific individual (Costabile et al. 2007a). The challenge of fostering social creativity requires easing the communication gaps (Section 1.1.2) among different CoPs and enhancing mutual development (Andersen and Mørch 2009; Mørch and Andersen 2010).

2.4.3 Appropriation

Appropriation is a common theme in ethnography of the workplace and the home. The study conducted by (Sproull et al. 1991) shows how users adapted email systems to their particular needs and exploited the opportunities which it introduced in ways not envisaged by management. Artifacts provide mediums for actions, since they can be adapted to support users' specific requirements. When technology change creates new social situations, traditional expectations and norms lose their power. People invent new ways of behaving.

From a sociocultural point of view, appropriation has been defined as "*the process of taking something that belongs to others and making it one's own*" (Wertsch 1998).

From an art perspective, readymades, a term coined by Marcel Duchamp (1913), are ordinary manufactory objects that achieve the status of art merely through the process of selection and presentation by the artist (Tomkins 1996). This notion destroys traditional perceptions of fine art, most notable in Duchamp's famous and provocative *Fountain*, borrowing and appropriating a porcelain urinal. Mørch integrates the readymades of art objects in software design and explores a "*software readymades*" vision to enhance end-user tailorability (Mørch 2001).

In the CSCW context, appropriation might involve customization in the traditional sense - the reconfiguration of the technology in order to suit local needs - but it might simply involve making use of the technology for purposes it was not originally designed for (Bentley and Dourish 1995). Pipek describes appropriation as a collaborative effort of end users, who invent or change software usages to make sense of software in their work context (Pipek

2005). Appropriation relies on flexibility in both practice and technology, and in particular flexibility in the way in which the technology can be mapped onto user needs. A system's flexibility however is determined by the ease with which the structures and policies embedded in the system can be customized. Customization in this sense implies not only the ability to mold and manipulate structures within the system, but also the ability to appropriate them and use them in new ways; support for customization is support for innovation (Bentley and Dourish 1995).

Dix (2007) associates appropriation with improvisation. Improvisation is crucial to "*get things done*," especially in a situation where users have to work with what they have, but lack exactly the right tool at hand, or have limited learning time. The improvisation and adaptation around technology demonstrates that technology has been domesticated, and that users understand and are comfortable with the technology to use it in their own ways. True appropriation occurs when the technology has become the users' own, leading to creative *"misuse"* and not simply what the designer envisaged - for example, using email as a bookmark tool communicating with oneself or to store files.

(Carroll et al. 2002) focused on understanding the process of technology appropriation and its reasons, as well as the resulting technology-in-use acting as a foundation for designing new artifacts that are then appropriated by users. They argue that appropriation is a process of evaluation, exploring the possibilities, testing how far users can shape and modify the tools.

Appropriation research has primarily taken a social perspective, for instance Orlikowski's study of the adoption of Lotus Notes (Orlikowski 1995; Orlikowski 1996), Grudin and Pelen's work on electronic calendars (Grudin and Palen 1995; Palen 1999). This research has revealed various social and organizational issues effecting systems deployment. Dourish, however, explores appropriation from a technical perspective and proposes an initial set of design principles for appropriable technologies, which are related to the component-based approach to system design (Dourish 2003). He suggests three design principles: supporting multiple perspectives on information, preserving visibility; and making information sharing an application matter rather than an infrastructure matter.

Stevens et al. provide a technical infrastructure to support appropriation work from a '*design for usages*' perspective (Stevens et al. 2009). The infrastructure provides communication channels to support: 1) users to reflect upon the usage of their software as well as share their appropriation knowledge, and 2) communication between users and developers to bridge the gap between product-oriented and process-oriented flexibility. Their approach is based on the assumption that "*a team of designers and engineers deal with the modularized code for maintenance and redesign purposes. Communication channels (*Figure 10*) should allow users to express design requirements towards the software team referring to the modularized structure of the user's interface"* (Stevens et al. 2009). This approach is EUD focused and

aims to involve end users in a continuous software development process, enhance their communication relevant to the software usage, and thus collaboratively support and enforce their interests.

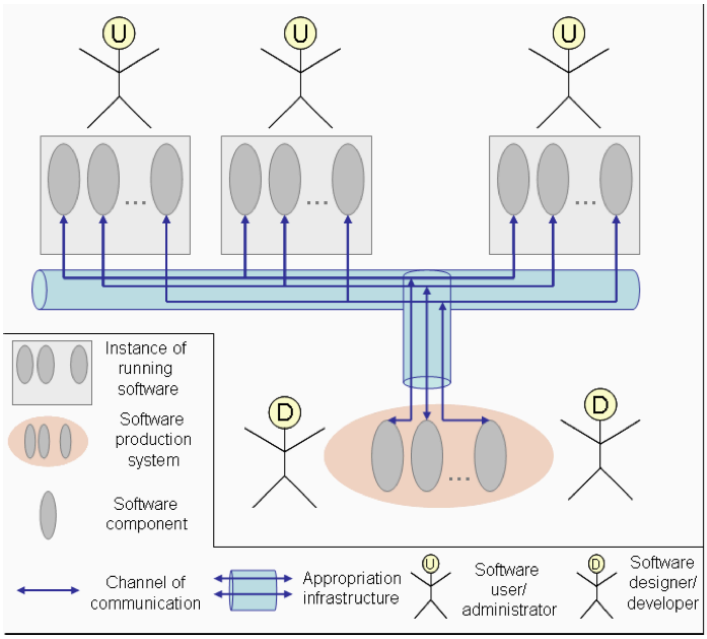


Figure 10 Communication channels between users and between users and developers (Stevens et al. 2009)

Although appropriation is very important, it is not well understood and there is lack of guidance for designing for appropriation. The first attempt to systematically extract appropriation knowledge and to present it in an applicable form is in (Dix 2007). In particularly, the *Plugability and Configuration* principle suggests an idea of creating a system bottom-up from small end-user combinable components (Ciborra 1996a; Ciborra 2002; Newman et al. 2002). In this case, "*users, not designers manage coupling*" (Dourish 2006). This principle aligns with the EUD and meta-design key concept, empowering users to take charge of their problems.

Appropriation is a situated intervention. A common thread to support appropriation derived from the abovementioned research is "*openness*" and "*flexibility*", creating artifacts that can be used in unexpected ways.

2.4.4 Bricolage

In *The Savage Mind*, Lévi-Strauss defines the concept of "*bricolage*" as a method of expression through the selection and synthesis of components obtained from surrounding culture. He describes the "*bricoleur*" as a person who engages in bricolage, making creative and resourceful use of whatever tools and materials at hand. The bricoleur speaks through the medium of things and through reappropriation of a collection of oddments left over from

human endeavors (Lévi-Strauss 1968). As such, objects become the way of communication and expression for the bricoleur.

Hebdige (1979) cites Hawkes's (1977) clarification of Lévi-Strauss's original anthropological definition of bricolage:

[Bricolage] refers to the means by which the non-literate, non-technical mind of so-called 'primitive' man responds to the world around him [sic]. The process involves a 'science of the concrete' (as opposed to our 'civilized' science of the 'abstract') which, far from lacking logic, in fact carefully and precisely orders, classifies and arranges into structures the minutiae of the physical world in all their profusion by means of a 'logic' which is not our own. The structures, [are] *'improvised' or made up […] as ad hoc responses to an environment …*

As a result, bricolage, as a creative process, is provocative, since it challenges the linear thinking of literate and technical minds; bricolage is emergent in that the bricoleur continuously reuses, recombines and reappropriates resources and creates new contexts for new processes and improvisation.

Bricolage can be a hobby (Lehrich 2005), interrogating tools and materials available and making use of and building things around them. Bricolage thus implies situated creativity, and the ability to cope with limited resources and to explore between the problem space and the solution space.

Bricolage is a concept that was used in the LOGO community. Seymour Papert defines bricology as a type of tinkering in contrast to analytic thinking (Harel and Papert 1991). Papert used the example of a child building a vibrating walker with Lego to suggest a different manner of working in which those who like bricolage, staying close to the object at hand, can do as well (and occasionally better) as those who prefer a more analytic formal style. Papert puts more emphasis on the aspect of resources combination (Papert 1993). He uses the concept of bricolage to serve as a source of ideas and models for improving the skill of making, fixing and improving construction.

Ciborra states that software engineering hacking is an analog process to bricolage. Hacking is an ingenious activity that through iterations, reuse, and reinterpretations of the existing programming environment leads to the implementation of new solutions. It transcends the orthodox, centralized and staged methods of software development in favor of an evolutionary and distributed approach (Ciborra 2002). It is noteworthy that the rigid boundaries of software engineering, such as user, reader, bug reporter, debugger, peripheral developer, active developer (Nakakoji et al. 2002) are blurred, since the participants in the hacker community are able to switch between the different roles according to the situation of the project (refer to Section 2.2.2 end-user classifications).

Bricolage can be seen in many different disciplines, such as, jazz, visual art, collage, DIY culture, and video mashups (Frere-Jones 2005; Laventure 2011). To a certain extent,

bricolage can be seen as appropriation; however, bricolage suggests creative actions in that it is emergent whereas processes and artifacts continuously require new knowledge. It is about reusing or recombining for new processes and artifacts, appropriating available resources to create new things, while appropriation can be merely using things or technologies differently.

Bricolage and Improvisation

Creativity in designing clearly involves improvisation beyond detailed plans and outlines, in making 'things right' in the day-to-day (Hallam and Ingold 2007). The cognitive view of improvisation stems from Suchman's study of situated action. She defines action as "*contingent on a complex world of objects, artifacts and other actors located in space and time. And this is an essential resource that makes knowledge possible and gives action its sense […] the circumstances of our actions are never fully anticipated and are continuously changing around us… situated actions are essentially ad hoc*" (Suchman 1985, 1987). It is therefore necessary to design systems to accommodate the unforeseeable contingencies of situated actions (Suchman 1985). Bricolage should be encouraged and supported for dealing with this emergence in that it implies ad hoc actions according to the situation and pre-actions.

The power of bricolage lies in fully exploiting the local context and resources at hand, while pre-planned ways of operating appear to be less effective since they do not fit the contingencies of the moment (Ciborra 2002). To this end, bricolage is situated. The notion of situatedness highlights that the foundation of actions is not plans but local interaction with the environment, more or less informed by reference to abstract representation of situations and of actions.

Bricolage is viewed as improvisation (Brown and Duguid 1991; Orlikowski 1996b; Orlikowski 2007). It is seen as a form of situated action that is important in organizational breakdowns and emergencies (Weick 1993) and when operating in the turbulent environments that high-tech companies routinely face (Ciborra 1996b). Both creativity and expertise are needed to be able to react flexibly and effectively.

The importance of bricolage as Ciborra argued, is that through improvisation outside of traditional methods and structures, it can reveal "*new uses and applications of the technology and the things and yield innovation*" (Ciborra 2002).

Bricolage is an activity where composition and execution, thinking and doing, converge in time. The decision making process is situational, testing and creating on the spot. The temporal dimension is compressed from several connected time spans to moment-by-moment improvisations.

Bricology vs. creativity

(Innes and Booher 1999; Loarne 2005) argue that bricology can be considered as one type of creativity and is more precise than creativity, since it consists of a process and a result that

happens within a problem solving situation which is not planned by the bricoleur. In particular, the approach used to define the concept of bricolage can be helpful to analyze the concept of creativity thanks to new factors: the mode of resources selection and the way these resources are used.

Creativity and expertise are needed to be able to react smoothly and in a suitable manner that fits in with all the relevant aspects of a certain situation. Therefore, bricolage is a special case of situated creativity in action - highly contingent upon emerging circumstances and unifying design and action (Ciborra 2002).

Both bricology and appropriation shed light on aspects of creativity that are still unexplored. By viewing bricolage as one type of creativity, new ways of exploration to better understand the context and the concept of creativity are opened. I frame creativity within a specific context, a problem-solving situation under certain circumstances. The concept of creativity is a stable concept, whereas bricolage is situated and from moment to moment has a continuously shifting goal. Answering this question could help us to explain creativity within a specific context, a problem-solving situation under the perception of time pressure.

Bricolagy vs. CBSD

CBSD (Section 2.2.4) aims to achieve easy software reuse and to reduce the complexity of EUD activities. The dilemma of CBSD is the specification of communication protocols between components and clear input/output ports. Several open issues are discussed in (Madiajagan and Vijayakumar 2006):

- Black box components make it difficult to predict how the components will behave under different conditions.
- Component integration inflexibility and lack of interoperability standards.
- Significant effort may be required to build wrappers and the "glue" between components of different vendors and custom code.

Bricolage however highlights the reinvention of design solutions at the end-user level (Ciborra 1992). "*Reuse*" in bricolage focuses less on interfaces and more on the potential for tweaking and appropriation, providing simple, transparent and understandable code and allowing both developers and users to tweak/evolve code according to the situation.

My research explores the possibility for integration of "standalone" components, i.e. components that need to be close together for spatial (cognitive, social) reasons, but that do not necessarily have to be close together to interact (i.e. to send/receive data). Hence, interaction among components can be an activity left to the users (as perceived by them as part of their work). Instead of trying to predict interaction points, bricolage-influenced components are made easy and open to modify so that new interaction points can be added, or just bypassed, when the need arises.

To this end, what can be added to CBSD is how to support software components that could benefit interaction with other components within the bricolage approach, i.e. to combine the conventional and the bricolage approach.

***In summary*:** I emphasize social creativity in that creativity emerges from social interaction. As I approach this research in the context of meta-design and EUD, creativity is not an abstract concept, rather making sense and emerging through designing and hacking.

To embody creativity, I focus on appropriation and bricology concepts to frame creativity in a specific context with situated action and available resources. However, in collaborative design, given the intertwined creation process rather than linear process and various involved CoPs, creativity needs to be refined with granularity accordingly - for instance, coming up with an innovative solution for a collaborative design project; hacking a system as an ingenious activity through iterations, reuse, and reinterpretations of the system; or opportunistically implementing solutions to cope with as well as extend underdesigned existing software environments. As such, tuning creativity implies that socio-technical systems need to offer support and evolve accordingly. In addition, I introduce bricolage into CBSD, providing standalone components to cope with the interoperability dilemma, and thus to encourage end users tweaking/hacking.

Appropriation and bricology however are exploited mainly as individual efforts. Therefore, I address appropriation and bricology in a social context, fostering tinkering as a collaborative, iterative and continuous social effort.

2.5 Design, Creativity and Software Systems

Within the scientific community there is a growing interest to design and build IT tools to support, promote, accelerate and facilitate creativity (Shneiderman et al. 2006; Shneiderman 2007). Richard Florida's book *The Rise of the Creative Class* points out that creativity is as vital as economic prosperity and social transformation and is associated with the ability to adapt successfully to constant change, inventing novel modes of doing things (Florida 2002).

Since 2003, there has been a renewed interest in creativity support tools and understanding their design issues. Many researchers and practitioners (Miller et al. 1992; Fischer 1999a; Norbert et al. 1999; Arias et al. 2000; Masanori et al. 2004) have developed tools and environments to support creativity in design. These systems highlight the usage of digital artifacts as boundary objects, as they accommodate different perspectives and embody tacit knowledge externalizations, building common ground (Clark and Brennan 1991) and developing mutual understanding among stakeholders over time. It is important that stakeholders construct a coherent design context and specify design constraints by structuring the process iteratively and collaboratively. The collaborative design process is not merely about designing the products, but rather jointly designing the process itself (Seitamaa-Hakkarainen and Minna 2006).

In 2006, a National Science Foundation (NSF) sponsored workshop brought together 25 leading researchers to share their expertise on creativity support tools, with the goal to "*develop improved software and user interfaces that empower users to be more productive and more innovative*" (Shneiderman et al. 2006). One of the outcomes of this workshop is a set of "*design principles*" to guide the development of new creativity support tools. These principles are different from other user interface principles as they address easy exploration, rapid experimentation and adventitious combinations that lead to innovations.

(1) *Support exploration*: Searching and browsing a digital library for related resources and thus being inspired.

(2) *Low threshold, high ceiling, and wide walls*: Users should work on projects that grow out of their own interests and passions - which implies that creativity support tools need to support a wide range of different types of usage and projects.

(3) *Support many paths and many styles*: This is particularly important for interdisciplinary design teams' collaboration, in that they have different ways, skills and cognitive styles to externalize their ideas.

(4) *Support collaboration*: Supporting the integration and iteration of the contributions of team members with their different strengths and talents.

(5) *Support open interchange*: Supporting extensibility and exporting and importing from other conventional tools.

(6) *Make it as simple as possible*: Providing the simplest ways to do the most complex things.

(7) *Choose black boxes carefully*: Identifying the central goal and hiding unnecessary complexity.

(8) *Invent things that you would want to use yourself*: Providing systems that we enjoy using ourselves. If everyone involved enjoys using the technologies, it is easier to form communities to help each other with new technologies.

(9) *Balance user suggestions with observation and participatory processes*: Giving users control where control is needed and making a difference in their experiences.

(10) *Iterate, iterate and then iterate again*: Rapid prototyping, playing with prototypes, and improving them.

(11) *Design for designers*: Empowering users to be designers, a process that itself allows them to become more creative.

(12) *Evaluate your tools*: Measuring creativity is still difficult. Multi-dimensional long-term case studies are needed to gain deep insights (Resnick et al. 2005).

Shneiderman proposes eight creative activities that require support from user interface tools to support, respectively *searching, visualization, consulting, thinking, exploration,*

composition, reviewing and dissemination. He emphasizes the social processes to accelerate discovery and innovation, e.g. consulting with peers and mentors, and disseminating results to gain recognition (Shneiderman 2000; Shneiderman 2007).

Mamykina et al explore computer-based tools that support collaborative creativity in particular interdisciplinary teams. They highlight the importance of supporting communication within interdisciplinary teams - developing a shared language and a common shared understanding, in that to communicate and exchange creative ideas is an essential part of the creative process. One way to encourage communication is to provide lightweight tools to support multiformat articulation of ideas (Mamykina et al. 2002).

Other similar design principles are discussed - for instance support for meta-design and socio-technical environments (Herrmann 2008), and distributed situations via enhancing awareness (Farooq et al. 2007). In particular, for supporting collaborative creativity in a co-located context, Herrmann provides heuristic design guidelines: supporting the larger picture, malleability of shared material and stimulation of variations, supporting of convergence within evolutionary documentation, smooth transitions between different modes of creative collaboration, integration of communication with work on shared material and supporting role dynamics and varying modes of collaboration (Herrmann 2010).

Some common threads behind all the guidelines for different design principles – from meta-design to bricolage and creativity - are, for instance, supporting multi-style, externalization, sharing and storing, rapid prototyping, multi-levels of learning, learning space, tailoring possibilities, finding common ground and so on. Creativity and design are often cobbled together. In contrast to a focus on novelty and design, (Hallam and Ingold 2007; Bærenholdt 2010) suggest emphasizing improvisation and imitation to reach a more generative understanding of creativity, as this encourages collaboration and diversity and depends on epistemic, material skills, and situated practice.

***In summary*:** Design principles derived from my literature review are used as inspirations and guidelines for the conceptual model (Chapter 4) and the software system building (Chapter 5, Chapter 6). In particular design principles from the NSF workshop (Shneiderman et al. 2006) are important guidelines for this research. Design principles for supporting co-located collaborative creativity (Herrmann 2008) are essential for my evaluation, in that whether and how a software system supports creativity can be immediately observed. Cultivating and supporting creative, collaborative design with software systems, such systems should take both top-down and bottom-up approaches, providing infrastructure/context for collaboration and encouraging ad-hoc activities.

2.6 Conclusions

This research benefits from generalizing and combining existing concepts, models and theories suggested in the literature. For each section I provided a brief overview of concepts

related to my research and in the summary I discussed some directions that my research can benefit from or further explore.

The next chapter discusses research design and some preliminary studies that have been conducted in parallel with the literature review.

Chapter 3

The first part of this chapter introduces the different methodologies used in my research. It then outlines my research design, which is divided into four phases: design principles analysis, conceptual model framing, system building and empirical evaluation. The second part of this chapter describes in more detail of some empirical and exploratory studies conducted in Phase I.

3. Research Design and Preliminary Exploration

In the following sections, I will introduce different approaches relevant to and inspired by this thesis. Moreover, my design research outlines how I utilize mixed methodologies in different research phases.

3.1 Methodology

Qualitative investigation is central in this research, in that it allows investigating the needs and the factors for supporting creative collaborative design via the computational environment. Complex design problems are dynamic, wicked and ill defined in nature (Rittel and Webber 1973), and thus solutions change from moment to moment (Suchman 1985) according to the design context. Collaboration is not merely a technical issue but also is strongly tied to social aspects. Qualitative investigation can provide deeper insights into existing collaborative design problems as well as cope with these issues in an ongoing manner.

The qualitative method investigates the *why* and *how* of decision-making rather than just the *what*, *where*, *when*. Qualitative research tends to focus on exploring, in as much detail as possible, smaller samples which are seen as being interesting or illuminating more often needed than large samples, and aims to achieve “depth” rather than “breadth” (Sherman 1988).

To provide socio-technical systems, this research exploits a mixed methodology. The main one is rooted in computer science and aims to provide a system solution dealing with co-evolution, for future computational artifacts. Other methodologies, for instance observation, interviews, and participatory design are also used at different stage and for different purposes during this research.

Design Science

Design research addresses artificial rather than natural phenomena (March and Smith 1995) and is rooted as a discipline in the science of the artificial (Simon 1996).

"Design science research is a research paradigm in which a designer answers questions relevant to human problems via the creation of innovative artifacts, thereby contributing new knowledge to the body of scientific evidence. The designed artifacts are both useful and fundamental in understanding that problem." (Hevner and Chatterjee 2010)

This definition highlights the importance of the creation of innovative artifacts as the research outcome, as knowledge and understanding of a problem domain and its solution are achieved through designing the artifacts (Hevner et al. 2004).

My research follows a design science approach, extending previous research on meta-design framing a conceptual model, the HMS model, and building its prototype, MikiWiki, which can be evaluated and improved over time. Developing and interacting with MikiWiki changed and evolved the initial HMS concepts, grounding them in real-world constraints, reflecting upon them in both theory and the practical building process. With respect to the meta-design approach, the designed artifact in this research is underdesigned and its evaluation should be an ongoing process.

Observations - Contextual inquiry

With respect to field observation and its conduct, Pettinari and Heath suggest four categories - passive presence, limited interaction, active control and full participation (Pettinari 1998). In ethnographical studies, recommended observation targets are for instance, physical setting, type and characteristics of activities, artifacts and equipment, key events, and patterns of interaction. In field observation, the researcher can take on various roles, for instance the complete participant, the participant observer, the observer participant and the complete observer according to Gold's model (1958).

I participated directly in collaborative design activities (see Section 3.3), studing two web development groups working on industry projects over a period of four months. The observer role adopted in the design projects was both a complete participant and a participant observer (Gold 1958), due to the time and resources constrains. This can be considered as contextual inquiry (Glaser and Strauss 1967; Darroch and Silvers 1982), in that it consists of observing and talking with users in their workplaces as they do real work. Observing and talking with

people in the context of the users' work helps researchers gather more concrete data as it is based on in-the-moment experience (Preece et al. 2002).

Observing and participating in the design projects were particularly fruitful as it involved various stakeholders, different roles, several processes and concrete design goals to be achieved, thus giving me insights into those basic shared concepts I was looking for in the mediation mechanism. In particular, I had opportunities to observe work practices, how all practices were derived from the use and recombination of very simple shared tools, the appropriation and combination of different views, and how situated creativity occurred in a real world setting.

Semi-structured Interview

In-depth or semi-structured interviews are one of the main methods used in qualitative social research. The method is described as being a "*conversation with a purpose*" (Webb and Webb 1932). Interviews "*provide access to the meanings people attribute to their experiences and social worlds. While the interview is itself a symbolic interaction, this does not discount the possibility that knowledge of the social world beyond the interaction can be obtained*" (Miller and Glassner 1997).

One distinctive feature of the in-depth interview is its intention of combining structure with flexibility. The interviews are based on certain forms of guiding questions and topics to be covered during the interview while the structure is flexible to allow revision according to the situation (Legard et al. 2003). The goal of the in-depth interview is to cover both breadth and depth topics, which can be achieved by asking content-mapping questions and content mining questions (Legard et al. 2003). The role of the researcher is an active facilitator, in that the research should enable the interviewee to talk about their thoughts, views and experiences; to manage the interview process in order to ensure coverage of the required subjects; and to probe questions to trigger more valuable answers (Pettinari 1998).

In the early stage of my research, 13 semi-structured interviews (40-60 minutes for each) were conducted to understand how people from different domains collaborate with one another; what tools they use and what their common problems are, especially focusing on communication gaps. In some cases I ask them to imagine their ideal collaboration software systems. I value improvised dialogue with interviewees and try to avoid predefined scripted thinking. Respondents are from interdisciplinary backgrounds, the majority of them working in the web industry, including project managers, web designers and software developers.

These initial interviews provided me with a deeper understanding of the presence of communication gaps as well as the lack of flexible tools that could change in repsonse to practice and integrate several aspects of communication.

Semi-structured interviews were also used in the final evaluation design study (Chapter 9) to get participants feedback on using MikiWiki and the reasons behind it.

Participatory Design

As introduced in Section 2.1.2, participatory design started as part of the Scandinavian workplace democracy movement in the 1970s (Nygaard 1986; Floyd 1987; Greenbaum and Kyng 1991; Bodker and Gronbaek 1991; Schuler and Namioka 1993; Muller and Kuhn 1993). Its aim is to involve stakeholders in the design processes of software applications.

The engagement of multiple user/designer groups in the design process can be achieved, for example, by conducting design workshops (Kensing and Madsen 1991), using mock-ups and prototyping techniques in early stages to simulate the work situation (Ehn and Kyng 1991; Mørch et al. 2004b). This can help researchers to identify problems, related to breakdowns and changes in use practices. Design results can be discussed and evaluated in early stages being used as input for the design process (Ehn and Kyng 1991).

In the DESIRE workshop I provided a set of paper card nuggets to prototype web applications (Section 3.5). I observed that nuggets cards were not only used as GUI elements but also enabled social interaction. The energy feedback system (Chapter 8) and the final evaluation Creativity Barometer (Chapter 9) both exploit the use of mock-ups for rapid prototyping. Mockups provided valuable insights into participants' ideas for software systems. Design sessions for the Creativity Barometer project are recorded (audio or video) to carefully analyze verbal cues and reactions from participants in order to understand how MikiWiki supports participants' collaborative design.

Action Research

Action research is a framework for information system research (Avison et al. 1999) that includes the expansion of social scientific knowledge as well as practical problem solving in social settings (Herrmann 2009a). It is a cyclical process that builds learning about change into a given social system (Hult 1980).

Action research emphasizes that complex social processes can be studied best by introducing changes into these processes and observing the effects of these changes. The change-oriented approach is the fundamental aspect of action research (Baskerville 2001).

The essence of action research is a two-stage process:

1. The **diagnostic stage,** which involves a collaborative analysis of the social situation by the researcher and the subjects of the research. Theories are formulated concerning the nature of the research domain.

2. The **therapeutic stage**, which involves collaborative change experiments. In this stage changes are introduced and effects are studied (Blum 1955). Based on feedback from subjects and observations from the experiment, a five-phase cyclical process is used: *diagnosing*, *action planning*, *action taking*, *evaluating* and *specifying learning*, completing the

loop by identifying general findings (Baskerville 2001). Action research is primarily applicable for the understanding of change processes in social systems (Hult 1980).

During the final evaluation experiments (see Chapter 9) I acted as an *observer participant* and followed an action research approach. The diagnostic phase provided me an increased understanding of an immediate social situation, with emphasis on the complex and multivariate nature of this social setting. I observed how the system is used in a social context and users' behavior patterns. Semi-structured interviews were conducted afterwards to consolidate the observations and gain insights into users' experiences. In the therapeutic stage, I introduced meta-design changes into the next experiment, and observed their impacts on collaborative design. This phase allowed me evolving the system for further observation of the meta-design impact. As such, I concretized learning from action research approach by reseeding my findings into the meta-design process, building the design artifact.

3.2 Research Design

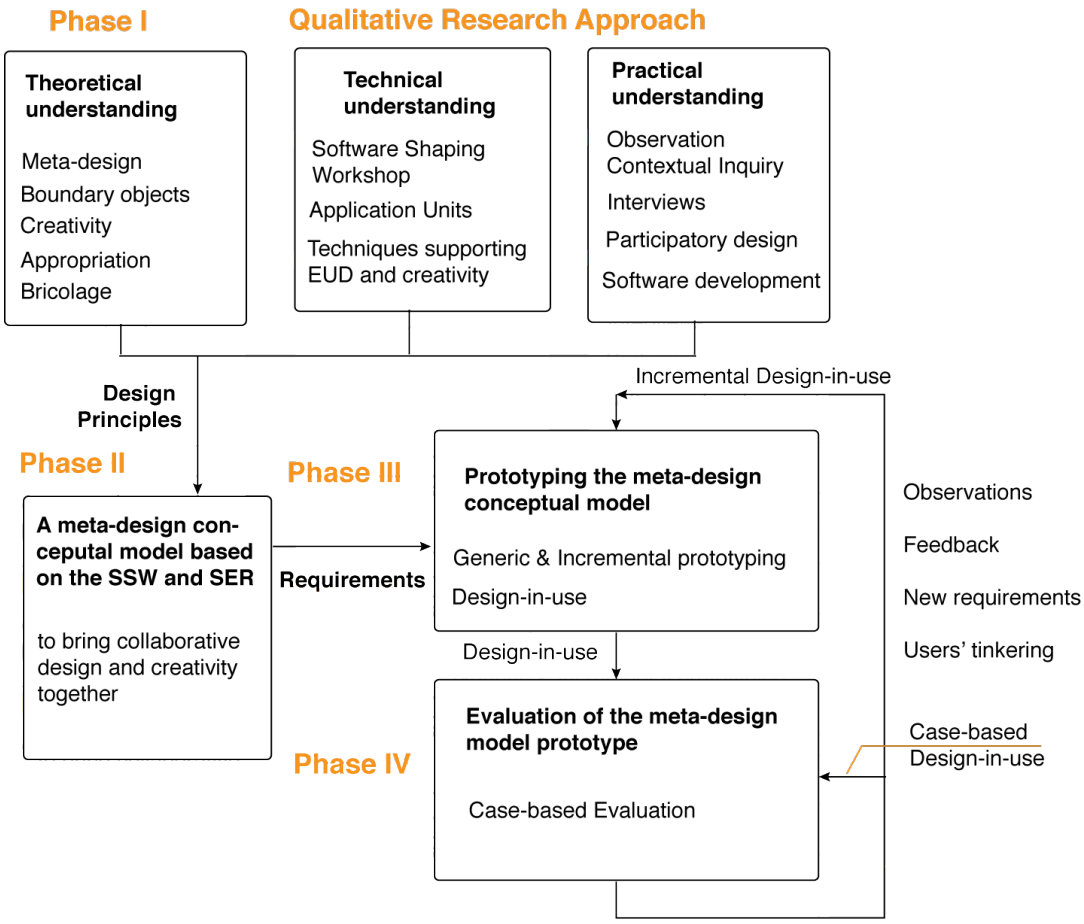


Figure 11 Research design

Research design "*involves combining the essential elements of investigation into an effective problem-solving sequence*." (Pelto 1970) Figure 11 outlines my proposed research design. It conceptualizes how each phase follows the previous one to build understanding and

eventually achieve my research goals. The research methods used in each of the four phases are discussed in detail below.

Phase I: The objectives of the first phase are 1) to better understand collaborative design and creativity from three different dimensions – the theoretical perspective, the technical perspective and the social practice perspective; and 2) to explore general topic areas of study and review the existing research and tools.

The main part of my contextual inquiry research uses participatory design and uses design studies to understand social phenomena in collaboration, as well as how the meta-design conceptual model should be formed and its system solution.

Collaborative aspects of web design companies are investigated. I use naturalistic observations and I am involved in several empirical collaborative design projects to identify when and how creativity occurs. This lets me gain a deeper understanding of the design features of the physical environment that support and enhance collaborative design and the rationale for the selection of the desired features.

Phase II: The objective of the second phase is to propose a new meta-design model for bringing collaborative design and social creativity together based on the reflection on Phase I study. I draw the synergy from these different perspectives and propose a meta-design conceptual model based on a theoretical understanding drawn from the literature, a technical understanding from the SSW methodology case studies, and EUD as well as a related literature review. I look at social understanding in terms of participatory observation and design. The conceptual model frames a set of principles that define desired functionality as well as its architecture and can be used as guidelines to implement meta-design systems.

A paper prototype boundary objects approach allows me to reason and evaluate design ideas in the early stage of the design process (see Section 3.4.4).

Phase III: The objective of the third phase is the implementation of the proposed abstract meta-design model. The implementation is incremental and reflects on the model features (Chapters 5, 6). In addition, the implementation is flexible enough to adapt to later different design cases. As the system is open and underdesigned to reflect its meta-design principles, the implementation phase can be extended and shifted to use time. As such, the generic wiki-based system is continuously designed and studied for the role-playing game, web mashups, software system prototyping and collaborative writing cases. As shown in Figure 11, Phase III and IV are iterative and intertwined.

Phase IV: The objective of the fourth phase is the evaluation of the prototyped meta-design model environment. A case-based prototyping approach not only aligns evaluation with the application domain, but also involves users collaboratively in design-in-use in practice (Chapters 7, 8, 9). Observations and open-ended questionnaires provide valuable feedback from users of the application domain, which can be used to incrementally evolve the

prototyped meta-design model environment. Participatory design methodologies, mockup design and prototyping are exploited. An action research approach is used to assess the conceptual model and its reference system (Chapter 9), which focuses on the process and meta-design introduced changes.

Design studies conducted over time, applying the system solution to different design cases, demonstrate the flexibility and adaptability of the system itself. The understanding gained from design studies in turn not only deepens my understanding of the conceptual model, but also contributes to evolving the socio-technical system. All these documented and presented factors show how an underdesigned open system has been continuously improved on over time.

In summary: the first part of this chapter describes different research methods. It outlines how I utilize mixed methodologies in different research phases. Interviews, observations and participatory design enable me to gain insights into my research context and collaboration problems. I then propose a conceptual model based on the qualitative research conducted in the first phase. Design science research focuses on building software artifacts. This research works through designing and designs through research, evaluating and refining both research theory and design artifacts in practice. Empirical design studies allow me to reflect upon the conceptual model and refine it accordingly.

3.3 The Avventurosa Community Project

Avventurosa is one of the multidisciplinary design projects I participated in and observed. This project aimed to build a social platform for independent travelers to explore and organize their trips. I participated during the inception and design phase of the web platform with the team for six days, mainly involved in defining design specifications, potential application interfaces and navigation structure.

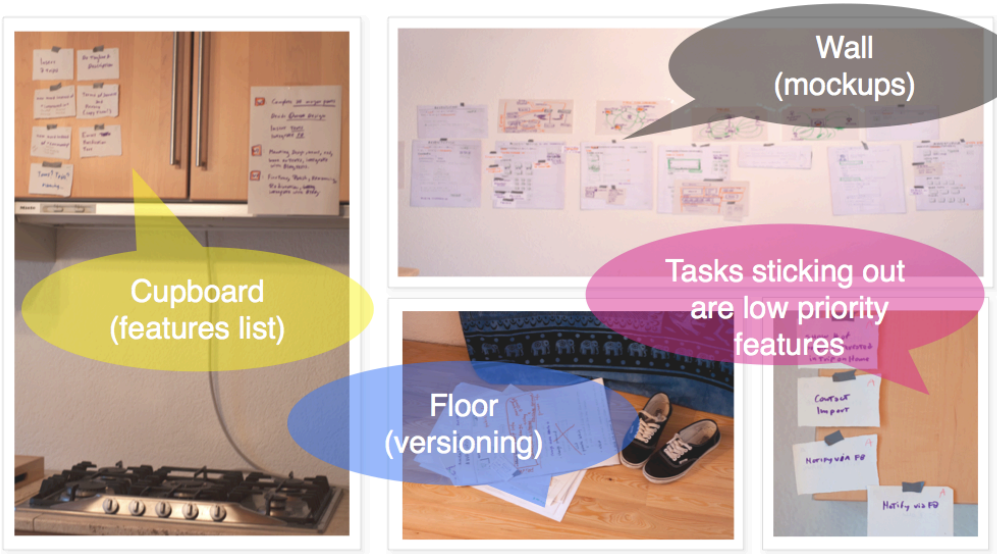


Figure 12 Utilizing physical space for arranging information

Embedded Social Context: A design studio is full of material objects and design artifacts. The arrangement of different artifacts in the studio is important to design activities and serves as organizational memory (Ackerman and Halverson 2004) and distributed cognition (Hutchins 1995) for design teams. Space and its structure are themselves artifacts that can be appropriated and reconfigured to mediate communication. Design teams are constantly adapting their design space, organizing things in space to prioritize their workflow and to get their work done. This concept further leads to the concept of inhabitable environments (Section 4.3.2).

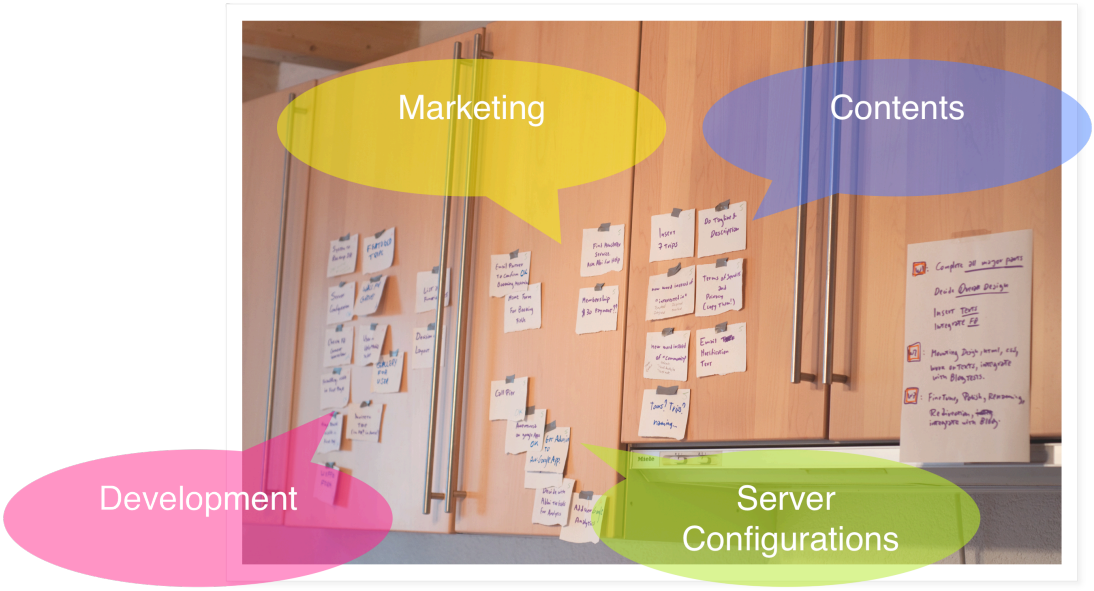


Figure 13 Using space to delimit communities of practice

Figure 12 shows how the design team used the wall as part of the design process, providing an overview of the design process as well as its status. Different spaces on the cupboard separate and structure different design contents. The wall is used to display the latest interface mockups. Older mockups are not thrown away, but piled on the floor at the foot of the wall, where they can be accessed at a later time in case there is a need to review an older version of the design. Notes partially sticking out from the cupboard imply nice-to-have features, which have a lower priority. Figure 13 illustrates how we used different cupboard panels to divide design tasks according to the roles and skills present in the team.

Figure 14 shows how the design team ranked features with red dots, each red dot representing a unit of estimated effort. Cards with tasks and red dot estimates were used to coordinate and to plan design tasks every day. Each red dot represented half an hour and the sum of the dots is the amount of time needed for each task. At a later stage people started circling dots when a task was partially done.

Figure 14 Task cards used to coordinate design activities

Appropriation: Figure 15 shows how bug-fixing tasks were stuck on the fridge door, which was a consequence of the fact that developers sat next to the fridge and they needed the high priority tasks close to them. The distance of notes from the developers became another explicit dimension that acquired meaning and could be manipulated as part of the process.

The ecological arrangement of artifacts in a workplace has two main usages. It allows work to be done (and thus comes before work), but also shapes and is part of how work gets done. Therefore artifact arrangement is mediated and part of the work itself.

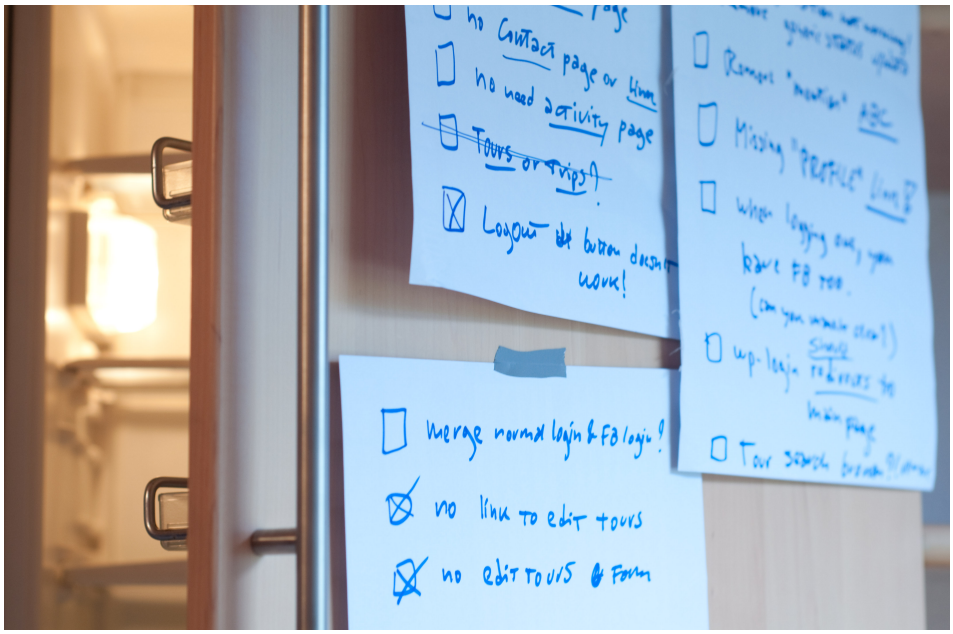


Figure 15 Information distance

Translation Device: Figure 16 illustrates how sketches became translation and social objects shared among the developers, the designers, the marketing manager and the project

manager. Information architects shaped up design ideas in the early design process through communication with freehand sketches, and designers communicated with clients through drawings in which visual sketches and textual descriptions were combined. The sketches became a design language that made sense to all participants, thus enabling mediation of the design work. They were effective as they were something to which stakeholders could refer to in discussion of the design and as a reminder pointing back to previous experience. They were effective and useful not because they mirrored real things, but because of the interaction and reflection they supported. The role of these design objects acting as boundary objects is introduced in Section 2.3.

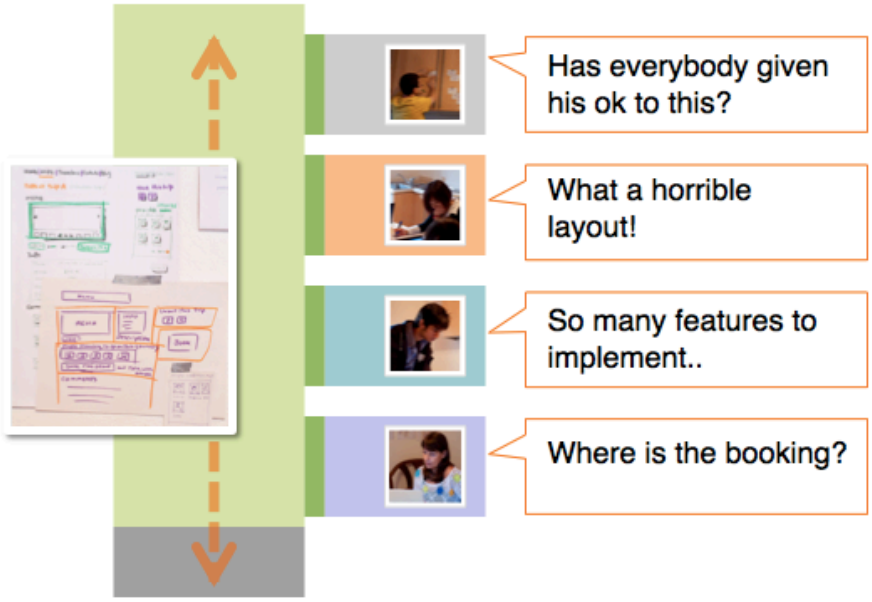


Figure 16 Exchange sketches

The design sketches are used as a means for storing history of the design process and serving as a collective memory.

Modularity: Notably, we broke down sketches into different components and labeled them, which suggested structures for documenting solutions. One could begin to refer to each item by name, using these names in discussions within the team. Names and reference numbers provided a roadmap for all the members in the project. Sketches represented different things to different people. An interaction designer creates detailed, well-organized specifications. A visual designer modularizes component artwork with similar boundaries. The developers were using component chunks as a roadmap for creating systems documents and test plans. The mindful organization of components could create efficiencies in communication and develop shared understanding.

Evolutionary Externalization: According to (Curtis 2009), artifacts, paper sketches, drawings, posters, cardboard, clay, foam models, and physical prototypes are examples of design externalization. Designers' externalization practices vary over time (at different stages

of design), in modality (from paper sketches to physical models), in purpose (exploratory or definitive) and are subject to individual preferences.

Figure 17 Evolving design artifacts over time

Figure 17 shows how designers built different fidelities of prototypes at different stages of their design process in the Avventurosa project. Figure 18 illustrates that complex collaboration and rich interaction were in fact supported by very simple low-tech tools, such as Post-It notes, color pencils, sketch papers and tape. It is the very simplicity of these tools that allows for their appropriation by design communities and encourages improvisation during the design and problem-solving process. However, achieving this degree of flexibility is a challenge with most collaborative software. Since each feature has to be planned in advance, this tends to foreclose such interactions and communications ahead of time.

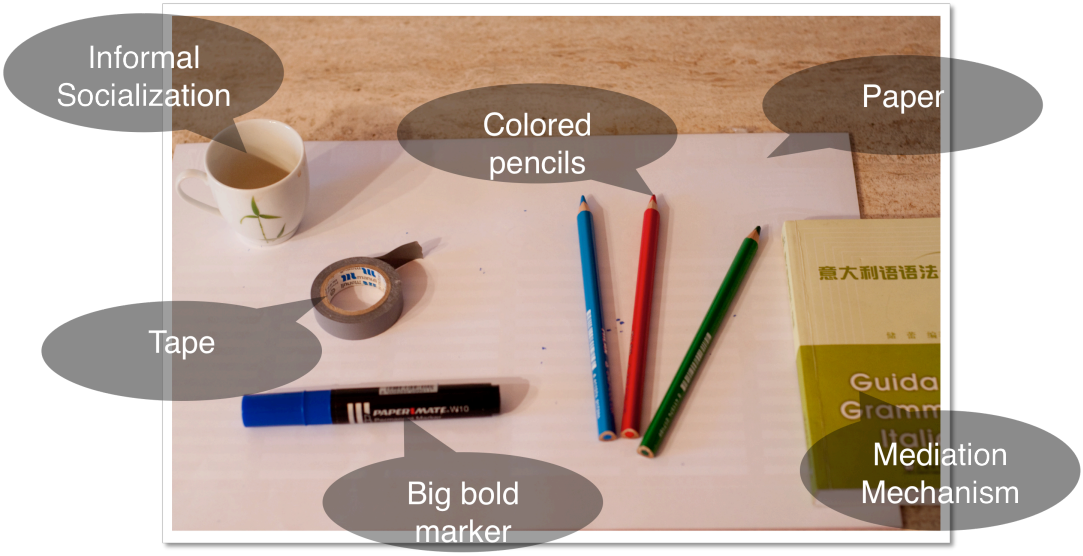


Figure 18 Small tools enabled collaborative design

At this stage an important aspect for me to explore is abstracting these patterns and introducing them in software design, demonstrating how complex collaboration and communication can be supported by simple tools.

3.4 Investigating Shared Aspects

This section describes the process of investigating basic, reusable units of sharing for collaborative social software. To extend boundary objects as a medium for social interaction and its evolutionary characteristics, I started analyzing popular social networking applications, trying to identify and to catalogue the common basic concepts and identifying what changed and what stayed the same across different platforms and mechanisms that support social interactions.

3.4.1 Harvesting Social Applications for Common Patterns

Most of the current popular social networking platforms are built around some basic object of common interest, socially augmented through various features. YouTube is video-sharing website built around videos (Hurley et al. 2005), Flickr is an image and video hosting website, and mainly built around pictures (Ludicorp 2004), and Last.fm is a music website built around songs as well as related content (Miller et al. 2002). In these networks, users become active knowledge contributors and organizers rather than being passive information consumers. Where platforms allow users to share and manage knowledge in real time, concurrent collaboration becomes possible.

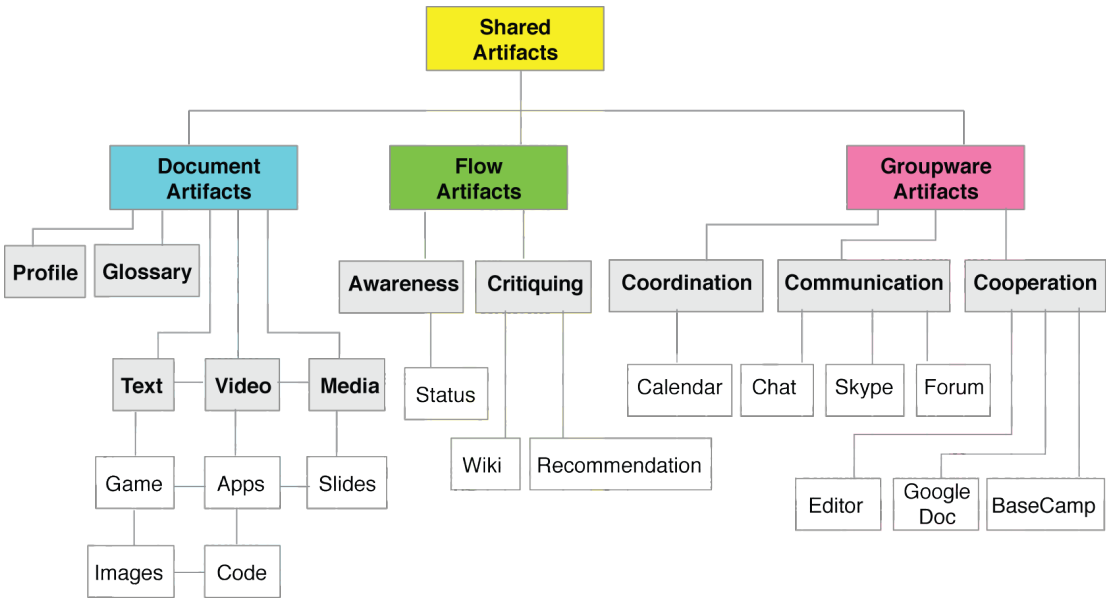


Figure 19 A tentative taxonomy of shared artifacts

Some of the examined social platforms were: Facebook (Zuckerberg et al. 2004), Last.fm, Flickr, YouTube, Twitter (Dorsey et al. 2006), LinkedIn (Hoffman et al. 2009), GitHub (Wanstrath et al. 2008), Del.icio.us (Schachter 2003), Digg (Rose 2004) and Wikipedia (Wales and Sanger 2010). The findings presented in this section are not to be considered exhaustive, but only a snapshot of the research direction at the time.

Social artifacts are the raison d'être of a social network and the main object of sharing: pictures, videos, songs, status messages (in Twitter), links (in Del.icio.us and Digg), user profiles (in LinkedIn), code on GitHub, and private messages. They usually include a title, description, author name and date of creation.

A specific platform can support several social artifacts - e.g. Facebook supports Status messages, Videos and Pictures. These artifacts can be – but do not have to be – user generated content (UGC). Figure 19 shows a rough early taxonomy of shared artifacts.

Document artifacts mainly mean finished content that can stand on its own and which is self-contained: text, video, audio, games, applications, slides, or images. All of these can be viewed as document artifacts.

Flow artifacts are artifacts designed to continuously change or be modified, such as user status, event calendars and stock market quotes and recommendations. These artifacts exist in a state of flow and change and the communication of these changes is central to their nature. Some of them can be *active artifacts*, changing based on external events rather than being changed by users (e.g. weather, temperature, stock prices).

Groupware artifacts artifacts are designed explicitly to support communication and collaboration. Communication artifacts such as VOIP applications and chat systems are centered on direct communication. Collaboration artifacts such as wikis and BaseCamp (37signals 2004) are focused on *knowledge accumulation*.

3.4.2 Social Elements

I identified several social elements that complement the main social artifacts and always have a graphical embodiment in the interface:

- Comments
- Annotations
- Follow button, Follower list
- Favorite button, Favorite list
- Liking / Ranking
- Tags

It is noted that although tags can be used to organize individual contents, they have strong social elements - for instance tagging photos in Facebook to communicate with friends, or finding relevant resources in delicious through shared tags.

These social elements are always UGC. They can be seen as social artifacts when associated with a primary social artifact.

3.4.3 Social Artifact Properties

Figure 20 shows an attempt at giving an early classification of shared artifacts and their properties. Shared artifacts can be either primary artifacts or user-generated content built on top of them. For example a video can allow comments and those comments could be commented on.

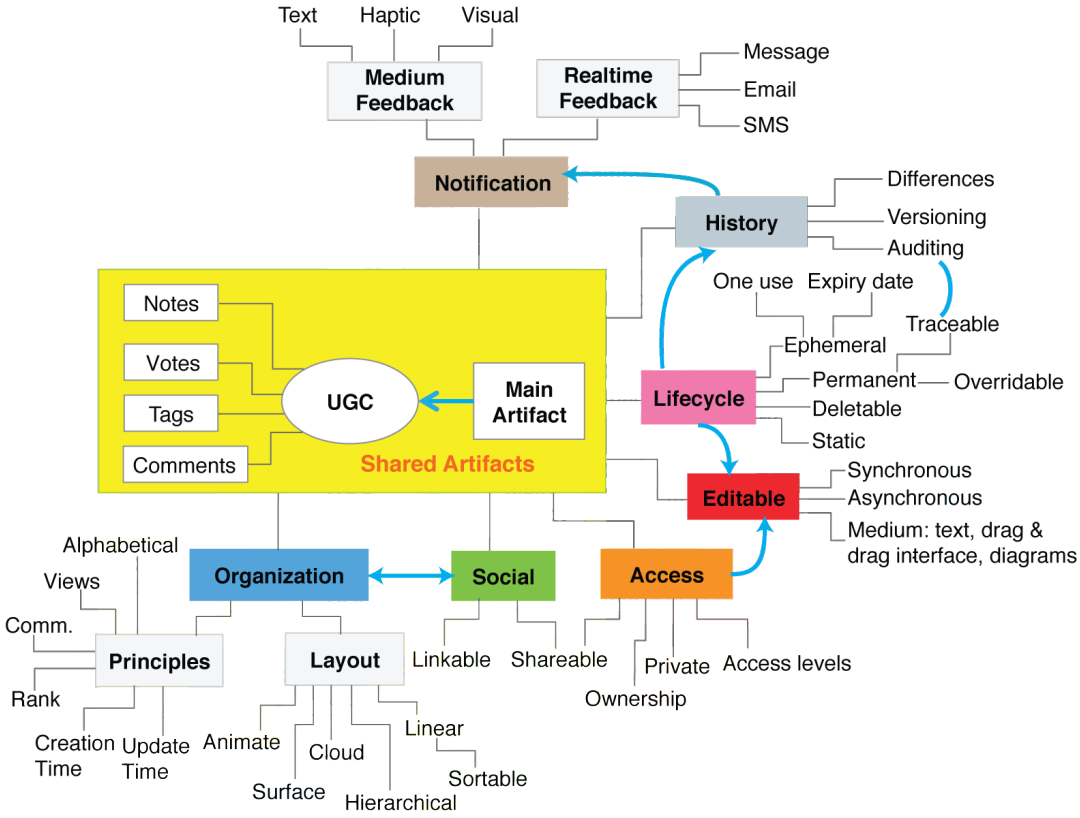


Figure 20 Classification of shared artifacts

Both primary artifacts and UGC can have certain behavioral properties:

- **Organization**: How artifacts are grouped, organized and sorted. This could be an open space to place artifacts, or they could be organized along a timeline or sorted according to any property of the artifacts or of the UGC associated with them, such as number of comments, author alphabetical order or tags. They could be organized hierarchically, or paged across several views. Cloud tags or animated views could be used for placing artifacts.
- **Social**: Artifacts can be linkable and sharable, allowing references to be passed on to the artifacts in other contexts.
- **Access rules and privileges:** Allowing both private objects and different granularity of sharing and read/write permissions.
- **Editable**: Artifacts could allow either synchronous or asynchronous editing and could support either locking or other concurrent editing policies. Also editing could be performed via different media e.g. text, diagrams and drag and drop interfaces.

- **Lifecycle**: Artifacts can be static, editable, overridable (replaced by successive edits) or undeletable. They could be ephemeral, be designed to be accessed only once, or carry an expiration date.
- **History:** History and versioning could be properties for editable objects. They could support auditing, to be able to track who interacted with them and what changes they made. Rollback, an option that returns to some previous state, could be supported.
- **Notifications:** Tracked changes can be notified by different means such as email, SMS or internal system messages. Notifications can provide awareness by using text, audio, visuals or haptic feedback.

3.4.4 Nuggets: A Palette of Social Components

Starting from these observations I designed a palette of icons representing social artifacts, elements and properties. I named them *nuggets*, to underline their concrete and self-contained nature. The intention was to use these nuggets as social components, a rough software materialization of boundary objects, as described in section 2.3. During the empirical experiments, I came to realize that these components can enable and encourage the social creation of boundary objects, rather than being boundary objects themselves.

Figure 21 Social Nuggets

Figure 21 illustrates the original palette of social components. Eventually they are translated to MikiWiki nuggets, digital remixable components within a wiki system, described in Chapter 5. The more abstract concept that underllies all nuggets is the "*HMS - Boundary Object*", one of the essential features of the HMS Model presented in Chapter 4.

Notably, the use of social nuggets to compose web services or interfaces is similar to web widgets. For instance projects such as Netvibes (Netvibes 2005) and iGoogle (iGoogle 2007)

focus on creating cross-platform reusable widgets. However, these widgets are generally hosted by third parties and thus cannot be easily accessed and modified by end users. These limitations can be overcome by integrated client-side programming and social sharing features.

3.4.5 Social Artifacts as an Analytical Tool

In this section I provide an example of using the concepts presented in the previous sections to analyze the structure of an existing social network – Flickr.

In Flickr pictures are the main social artifacts. Photos are associated with other elements: user generated content, awareness properties and organizational properties (see Index, Figure 22). Specifically, photos are organized according to their upload time.

In terms of UGC:

- Photos can be tagged and tags can be used as an organizational (index) principle, either as a tag cloud or by using alphabetical order.
- Users can annotate photos, and other users are notified when a new annotation is added to the picture.
- Photos can be commented, with the comments ordered by their creation time. Users are notified about the new comments.
- Photos can be flagged as favorites
- Number of views and view frequency related to photos can be inspected.

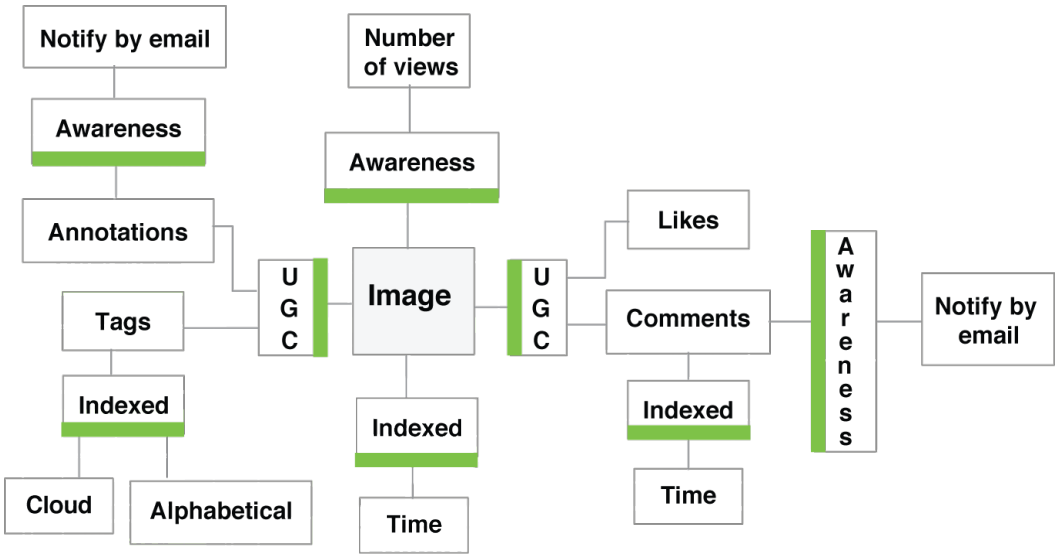


Figure 22 Flickr breakdown

The possibility of exhaustingly explaining the properties of a social environment by using this component-based approach motivated me to explore further in this direction. My intuition was that this analytical structure could be also put to use as a generative structure allowing users to assemble and tweak other similar systems.

3.5 Experimenting with Nuggets

In the course of a DESIRE Network meeting at Porto, I conducted a workshop together with another researcher. Workshop participants used the cards from the nugget palette in the course of a brainstorming session aimed at designing a collaborative application, e.g. car sharing and language learning web applications. The participants (PhD students, researchers from an interdisciplinary background) could use the pattern recipes and the basic nugget building blocks to produce the interface designs. The nugget cards seemed to be very good for brainstorming, sometimes being preferred to patterns and used in a bottom up fashion.

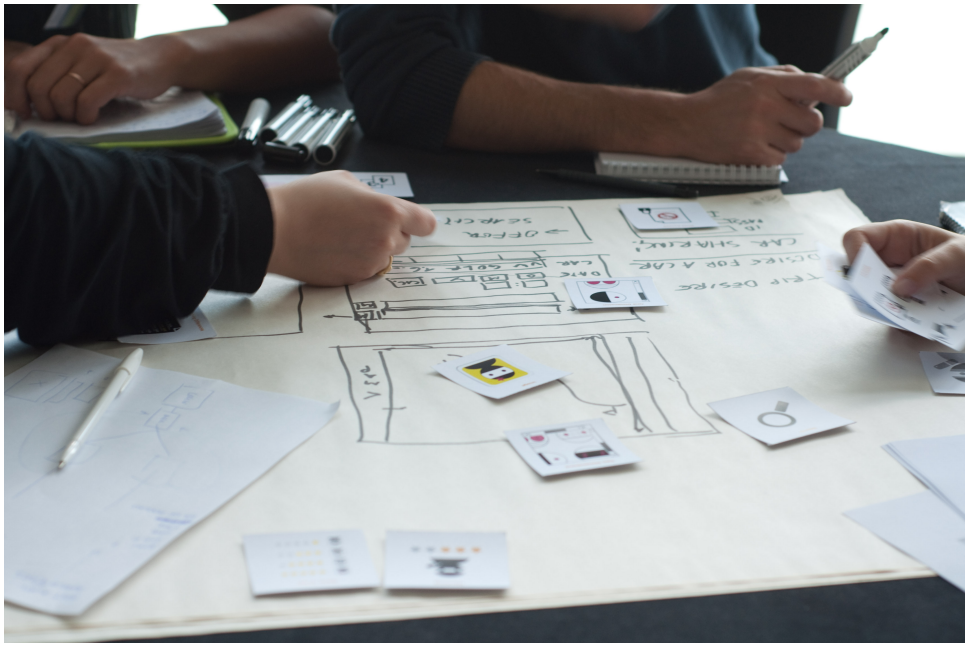


Figure 23 Mixing interface sketches with nugget cards

The way the cards are designed aims to filter unnecessary complexity and at the same time play an informative role, providing certain visual cues. They allowed participants to explore consistent ideas focusing on the design rather then being limited by technical constrains. During a brainstorming session, the fact that users could point to, discuss and pass around the cards supported communication, encouraging social interaction and fostering creativity (Figure 23, Figure 24).

The use of nugget cards led to the recognition of five ways in which they supported the collaborative design process (Carneiro and Zhu 2011):

Interdisciplinary conversation: When dealing with interdisciplinary teams, the nuggets can be used as a bridge between different languages and references. They provide a common visual vocabulary to support discussions, even when people come from different backgrounds.

Visual tool: Nuggets can be applied to illustrate an idea, and engage novices on the topic during the discussion and conception of well-grounded interactive objects.

Ideas encouragement: As they comprehend a large set of possibilities, nuggets can be used as seeds to stimulate conversation and open up possibilities when the participants run out of ideas.

Structure visualization: Nuggets support the structuring process of interactive systems hinting at connections and allowing them to be freely arranged.

Appropriation kits: Nuggets support unanticipated usage and appropriation. For instance, two researchers were using the square cards as a grid system rather than for their visual content. Users can easily interpret visual content, assigning new meaning to cards or defining ways of using the nuggets according to their design tasks.

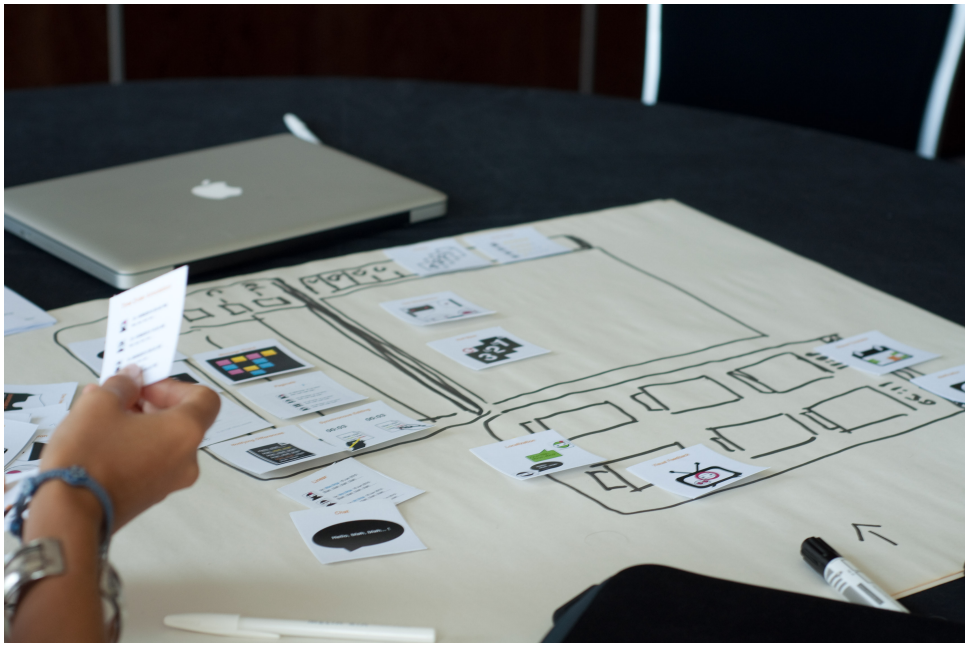


Figure 24 Reasoning on design with boundary object cards

3.6 Conclusions

This chapter provides a brief overview of the initial research directions that I followed and some detail on the informal findings that informed my research, leading me towards researching social artifacts as an analytical tool and nuggets as generative components. I later on organized these concepts within the HMS model, as described in Chapter 4.

Chapter 4

This chapter introduces the hive-mind space (HMS) model (Zhu 2010; Zhu et al. 2010a; Zhu et al. 2010b; Zhu et al. 2011a; Zhu 2011), a meta-design conceptual model to support collaborative design. It not only empowers end-users for design-in-use but also addresses social creativity. My intention is to draw upon differing concepts and models and to bring a synthesis of their features into the HMS model. I first lay out some key concepts from a review of the literature and explain how I integrated them into the conceptual model. I then explain how each attribute of the HMS model supports various aspects of creative, collaborative design, how it builds on and enhances the original theories.

4. Hive-Mind Space Model: A Meta-design Model for Collaborative Design-in-use

This chapter sets out the hive-mind space (HMS) model, a meta-design conceptual model for bringing diverse design communities together and fostering collaborative design and creativity. HMS is a meta-design model that builds on SSW methodology (Section 2.2.5.2) to encompass social creativity, as it integrates other desirable properties, especially from design and creativity objectives.

4.1 Why HMS is Needed

The Avventurosa project, described among preliminary studies in Chapter 3, presented an analysis of face-to-face interaction and work processes inside a diverse design team. The question here is what kind of system best supports the rich interaction and evolution identified in the course of that study. Most systems for creative collaboration focus on providing tools, whereas creativity and innovation are also about supporting flexible creation processes. Moreover, previous models for collaborative software do not address situated evolution, while appropriation and situatedness (Suchman 1985) are essential to collaborative creativity and

to tackling “wicked design problems” (Rittel and Webber 1984). Accordingly, those models that do take on the evolutionary growth of collaborative processes and artifacts fail to provide practical architectural guidelines for implementation.

Hence, there is a need for a model that provides guidelines for designing collaborative interactive systems that fulfill the following objectives:

1. Bringing heterogeneous design teams and different perspectives together
2. Enabling meta-design, to tackle co-evolution issues
3. Supporting design communities in collaboration and communication, thus enhancing their social creativity
4. Providing enough detail to create technology platforms
5. Empowering uses to create situated solutions.

Such a model is epitomized by the case of the design studio, where professionals from different domains come together to complete a project for their clients. This chapter often refers to the example of the architect’s firm, because of the familiarity of the situation, and to the web-design studio, because of the intrinsically digital nature of the artifacts.

4.2 Developing the Hive-Mind Space Model

The HMS Model implies a space that harnesses the ‘*hive mind*’ (Section 1.1.1). In analogy with self-organized systems such as ant colonies, a hive mind implies a bottom-up system, which does not have a clearly defined hierarchy, and follows the rules of organized complexity that could lead to emerging structure and collective intelligence greater than the sum of each individual. In this case, the hive mind not only implies cultures of participation but also ad-hoc activities.

The hive mind is a powerful process that enables individuals to create and share digital content and allows them to affect one another’s experiences. The spatial boundaries within a network afford different social groupings - for instance private environments and public environments, connecting distributed design teams. These environments are inhabited and shaped by design communities over time.

The major features of the HMS model that are presented in this section are:

1. Habitable environments
2. The boundary zone
3. HMS boundary objects
4. The mediation mechanism
5. Levels of participation
6. Open infrastructure
7. SER model

The HMS model is based on SSW methodology, in that it is a complete and explicit meta-design model. SSW emphasizes the need to provide personalized environments to all stakeholders, in terms of language, notation, layout, and interaction possibilities. The HMS inherits the network of workshops of SSW – here called habitable environments – while simplifying the means of communication by introducing the concept of boundary objects, which are shared and exchanged within a boundary zone. The three levels of design of SSW are also present, but they are pervasive within each environment, enabling meta-design to be accessible to all communities.

It is important to underline the fact that concepts such as boundary zone and boundary objects do not identify static entities, but fluid processes. While the fluid transition and transformation of boundary objects in face-to-face scenarios may never be fully equaled by software, the HMS provides a description of those concepts that are needed to investigate and replicate socio-technical systems.

The HMS frames design principles and structures for the design of interactive systems, in way that addresses the preconditions for social creativity by enabling meta-design processes, promoting cultures of participation, ad-hoc tinkering and coping with situated emergent socio-technical issues.

4.2.1 Habitable Environments

In order to participate in SSW workshops, design teams are provided with localized environments that are adapted to their culture, skills, and articulatory abilities (Costabile 2008).

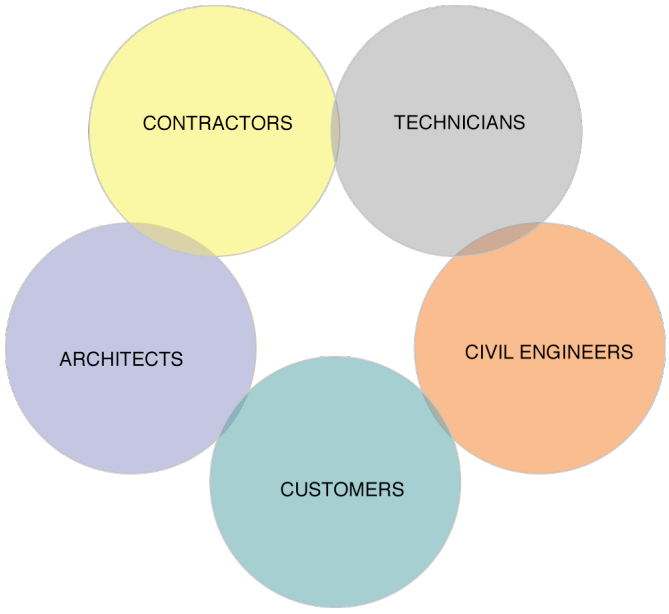


Figure 25 Design Communities

Habitable environments, according to the HMS model, can be seen as SSW workshops, tailored to different design teams' needs and equipped with the essential tools they need to

perform their tasks. The HMS model differs from the concept of the workshop, in that it allows design teams to build and refine new ways to communicate with other communities. The new affordances hold out the potential for introducing the metaphor of creating virtual habitable environments (Section 2.2.5.2), thus allowing CoPs to develop their activities in situ rather than following a pre-defined course.

Environments are *habitable* in the sense that they are not simply architectural features, but they are designed to be living evolving places. Each CoP has their own environment, in which they perform their activities and actively co-design the environment as well as improve it according to their needs. As a whole, this environment in turn evolves, reshaping the way inhabitants behave and offering them new ways of interaction during the process of use. It is noted that habitability is also about the quality of a system that makes it comfortable to learn and use (DeFanti et al. 1974).

Each environment is shaped according to the needs of a specific community in terms of the activities that have to be conducted within the environment, but also following the users' requirements in terms of their role and culture. Environments can be optimized to work on particular devices or under specific constraints. Figure 25 illustrates an example of different CoPs that need to collaboratively design an apartment. Each CoP has its own working environment, tailored to its needs. The following section will explain how it is important to bring diverse design communities together, to integrate the different work spheres, and consequently integrate the knowledge and experiences of all the stakeholders.

4.2.2 The Boundary Zone

Adding the boundary zone concept to the SSW model is the initial novelty of the HMS model. The boundary zone is common meeting space between different habitable environments, where CoPs can gather together to exchange knowledge. In addition, the focus on boundary objects in this meeting space is a means for effectively tackling collaboration difficulties.

In the SSW, specialized peer-to-peer protocols handle the exchange of artifacts – and the communication – between different workshops. The HMS boundary zone acts as a central communication channel that allows the exchange and management of boundary objects. Since all CoPs can access the boundary zone, this space supports communication by allowing the sharing of boundary objects via common protocols.

Figure 26 shows an example inspired by the Avventurosa project described in Chapter 3 and reinterpreted in the light of the HMS. Five communities from different domains can all access the same interface-mockup artifact, which acts as a boundary object by being a prompt and obvious referent for discussion. Each community is concerned with different aspects of the mockup-interface, and may even decide to focus on only one aspect of the boundary object, as introduced in the section on the *HMS-mediation mechanism*. Nevertheless, the shared

boundary object is always the same and has the valuable property of reflecting the latest changes and updates via the boundary zone for all the CoPs involved to be aware of.

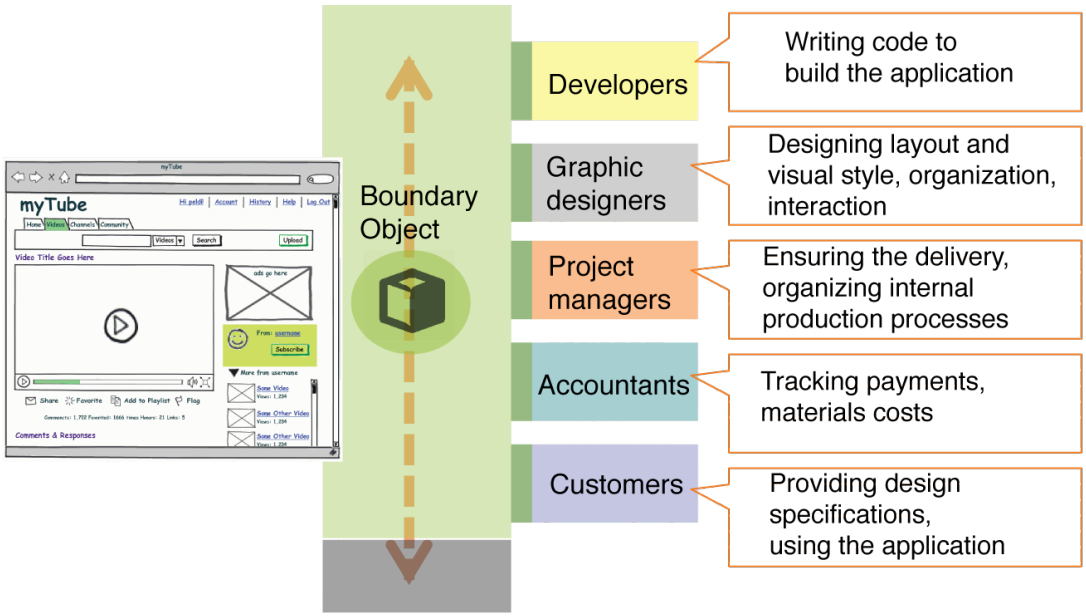


Figure 26 Boundary Zone, acting as common access point to shared resources

The communication channel provides the entire model with flexible ways of communication and interaction by flattening the hierarchy of communication presented in the SSW model.

4.2.3 HMS Boundary Objects

Utilizing boundary objects to facilitate communications is not a new idea. For example, architects use physical models or sketch-boards as boundary objects to express, discuss and reason their design ideas with clients, civil engineers and related groups.

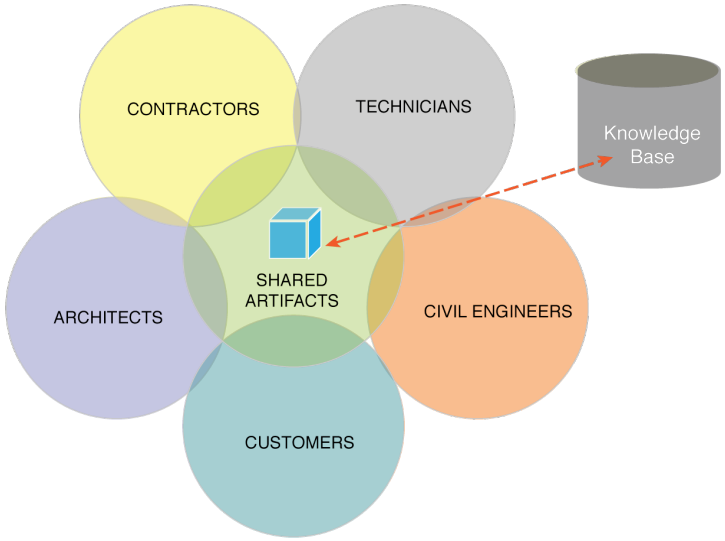


Figure 27 Artifacts shared among different design communities

Boundary objects can be dynamic and evolve over time (Star 1989). For instance, a map-based wiki can be viewed as a dynamic boundary object (Barricelli et al. 2009a): a tourist can

access a map-based wiki and enhance it with feedback on a visit to a certain region allowing the entire community to view and comment on that feedback. The map-based wiki therefore becomes a common ground connecting two or more separate networks of people and encouraging communication among them.

As illustrated in Figure 27, CoPs are able to extend their activity and to create shared boundary objects within the boundary zone. Artifacts are the natural boundaries along which people interact and communicate and can be used both as a bridge between communities and as object of discourse within communities.

Within habitable environments, each community of practice develops and experiments on certain artifacts that act as boundary objects between the members of the community. In order to guarantee efficient communication among these different participants, each environment can be equipped with a set of seed boundary objects that allow communication both within and between the environments. For instance, an annotation tool is a boundary object by means of which participants could make observations about the exchanged information and proposals about possible changes to the system in order to improve it. In this way, each member of the team can directly experiment with the system, inspect and critique the recommendations and notes of other CoPs and negotiate the system evolution. The same tool could be used within an environment to keep track of the evolution of an artifact that users from the same community are working on.

In the HMS model boundary objects are not only the content of an interaction, but also can form the very medium that enables and directs communication. The HMS boundary objects are artifacts that can be continuously modified, discussed and socially negotiated. Every communication in the HMS is mediated by boundary objects. The whole set of the boundary objects forms the HMS infrastructure for communication, interaction and partitioning of space as well as being a shared knowledge base and history of all interactions.

The default mode of boundary objects in an HMS-based system is to be shared artifacts, accessible by all the CoPs and representing the integration of knowledge, social interaction and different creation phases. A constellation of boundary objects provides form to the initially shapeless space of the boundary zone, refining it according to the needs of individual participants and communities, in time forming a new environment common to all the relevant CoPs.

While boundary objects can be identified with concrete well-defined artifacts, this does not always have to be the case. Boundary objects represent dynamic processes combining tight interaction and awareness and can map in different ways to concrete artifacts. For example, two different groups could use the same whiteboard artifact to communicate, effectively generating two different boundary objects. A good HMS reference architecture should aim at

providing technical support for this kind of fluid identity and reinterpretation of boundary objects

4.2.4 Mediation Mechanism

Boundary objects are the means to enable communication between participants in the HMS model. However, although different user communities may have a shared communication channel, this does not prevent communication breakdowns due to different interfaces, cultural expectations or different domain expertise and notations.

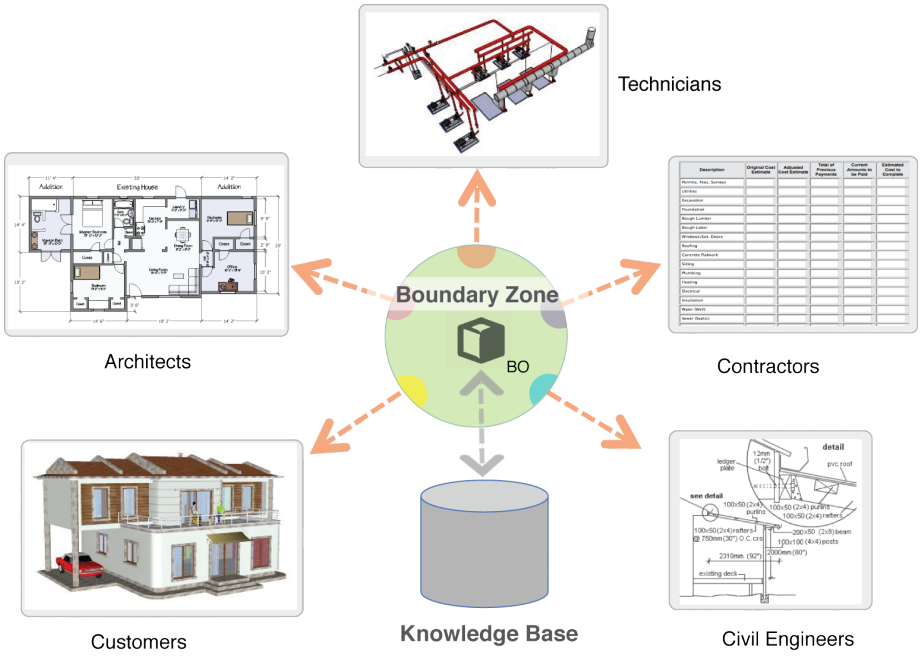


Figure 28 Different perspectives

The HMS model tackles this problem by providing the tools to make knowledge meaningful to diverse users and CoPs, in order to support them to reach a common shared understanding. For instance, when building a small apartment, the same information could be materialized differently to different users: floor plans for architects, construction details for civil engineers, pipeline constructions for technicians, 3D rendered models for the residents and spreadsheets and bills of materials for contractors (Figure 28).

Each environment allows its inhabitants to define a mediation agent, which adapts the exchange boundary objects across the boundary zone to the system of signs associated to the specific perspective (Figure 29). Thus, the mediation mechanism allows the same boundary object to be represented – and interacted with – differently in different environments. When the domain of interaction is clearly defined, designers and meta-designers can define the environments and the relative mediation agents in advance.

However, as discussed in section 2.2.5.2 a mediation mechanism needs to consider a temporal dimension (situatedness) and the intersubjectivity (Mørch 2007; Fugelli 2010) characteristic of social interaction. The HMS model allows leveraging divergent viewpoints in

a CoI for solving complex design problems and coming up with innovative solutions: users can experience different viewpoints by accessing different environments, where different mediation agents are activated.

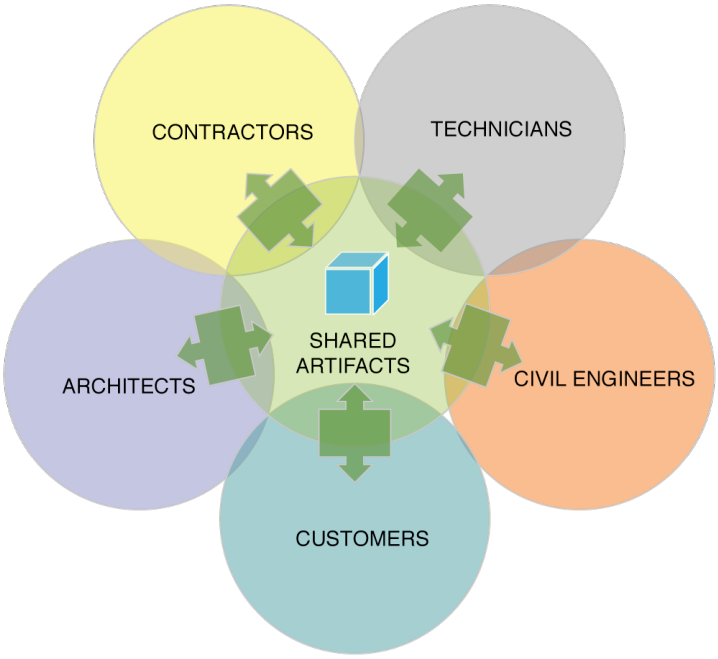


Figure 29 Mediation mechanism

The core concept of the mediation mechanism is that the same knowledge object can expose different *boundaries* to different participants, allowing different modes of representation and interaction. In actual collaboration, this mechanism has been shown to be useful in at least three different use cases:

- **Localization –** (Barricelli 2010) presents three different axes along which mediation can happen: role, culture and device. These can be seen as three different dimensions of localization, adapting an environment to a set of mostly static circumstances. The role may require the prioritization of different information or the availability of new types of interaction. For example, architects may wish to edit some features of a building, while contractors may need to edit the cost of materials. The culture can require changes in language, in the text layout or in the use of iconic symbols. Mediation according to the device needs to consider the bandwidth of the available channel, establish information priorities and different styles of navigation and interaction.
- **Opportunistic bricolage** – when a user encounters a perceived limitation of a boundary object, the mediation mechanism can be utilized to refine or tinker with its behavior, without creating any interference to the interaction for other communities. The underlying data remains the same, but rapid changes can be introduced in how it is visualized or interacted with. Design studies presented in Chapters 7, 8 and 9 provide some examples of how opportunistic bricolage has been used to augment artifacts with additional information, to prevent certain modes of interaction and to enable different navigation

modes for some artifacts. In other cases it was used to apply different drag and drop interaction techniques depending whether the user was accessing the artifact via a laptop or a tablet.

- **Exploratory branching –** the mediation mechanism can be exploited to explore new interfaces and representations for existing data. In this case the meta-designer creates a separate environment where she can experiment with and evolve alternative mediation mechanisms without affecting other environments. Once a stable version of the new mediation mechanism has been created, it can be made available to all other communities.

An important characteristic of the mediation agent is that it can be tailored and socially evolved over time, rather than remain predefined and unchangeable. As such the mediation agent acts as a boundary object in itself, being open to social negotiation and continuous evolution as social practices and understanding change in time.

4.2.5 Levels of Participation

The HMS model structure, as shown in Figure 30, presents three levels of participation and their typical users. Within the meta-design and EUD context, users are in charge of defining and composing their own specific modes of collaboration.

1) **Meta-design level**, which is accessed by software engineers and domain experts who either design and maintain the architecture of the overall system or create extensions to individual environments.

2) **Design level**, where domain experts design applications for end users, using the tools created on the meta-design level.

3) **Use level**, where end users tailor and use the applications created for them. Each CoP should be provided with a habitable environment that offers all and only the tools needed by its members to perform their activities.

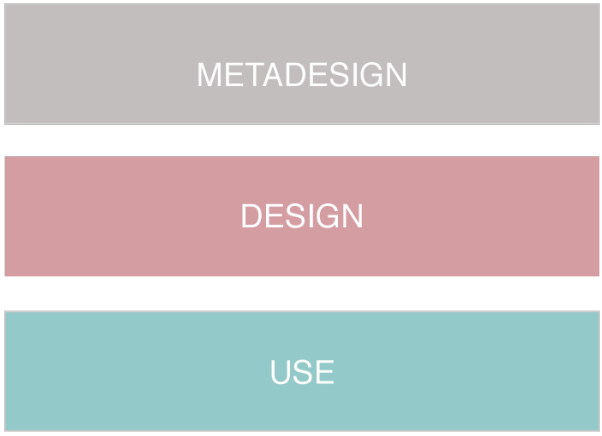


Figure 30 Three levels of participation

To return to the apartment-design example, software developers could act as meta-designers, providing architects with new tools that they can use to design different apartments. For example, meta-designers could add a resize feature to some *wardrobe* and *bookshelf* artifacts, but not to *chair* or *sink* artifacts. The apartment architects would then use these new tools to design a specific apartment and create a palette of available furniture. Finally, customers could visit a specific apartment created by the architects, and select as well as arrange the available furniture.

The roles associated with different levels of participation are indicative only of the associations with a traditional top-down design process. The HMS actively promotes a blurring of such distinctions, inviting participants to switch between different modes of participation and experiment with tailoring artifacts and the environment.

While in the SSW the use, design and meta-design processes happen within separate workshops, in the HMS model they happen on different design levels that can coexist within the same environment. To foster social creativity, a smooth transition between different design levels is needed (Fischer and Herrmann 2011), encouraging a richer ecology of participation: different levels of participation allow users not only to work at different levels, but also to start from simple and become more advanced over time.

4.2.6 Open Infrastructure

The HMS model has an open infrastructure, allowing adaptation and evolution over time to better support creative collaborative design. The initial space is appropriated by participants that give it form and differentiate it in different environments, producing new CoPs and new channels of communication within the overall framework described by the model.

The open structure of the HMS model allows adding more environments and restructuring the existing ones. Openness and simplicity are important characteristics as they encourage a degree of exploration and understanding that would not be possible in a closed or well-structured environment, requiring the study of its integrated infrastructure and patterns. These simple properties and principles are often lacking in most collaborative software, where every feature has to be planned, implemented and polished in advance, without any chance for situated adaptation.

Whereas other CSCW systems and models allow for changes and evolutionary growth, there usually is an assumption about basic roles, environments and processes. The example in Section 2.2.5.2 shows how SSW methodology provides a structured process to evolutionary growth in the context of the medical domain: software engineers as meta-designers create components that designers can use to compose interfaces for the end users, and feedback is regulated by the annotation tools. An open system is explicitly trying not to optimize towards specific processes, providing transparent access to components with less direct affordance (Wakkary 2009) that ideally could be more easily appropriated and modified.

Open Inclusive Structure

It is common for users to solve problems by leveraging various applications, optimizing tools for design tasks. However, it would be impractical and unnecessary to recreate all types of existing applications within the collaborative environment.

The HMS model provides interoperability by allowing access to external systems and by exposing its functionality as a service for external agents. The open infrastructure allows referring to artifacts from an external system by representing them as boundary objects within the HMS model. This assumes the availability of a common underlying protocol that allows some degree of interoperability between the different systems. As discussed in the following chapters, this has been done in practice by utilizing extensive web technologies, practices and conventions. Providing access to external services allows the system to extend, accommodate and build upon both pre-existing and new systems. At the same time, an HMS instance should be able to interoperate with other HMS instances and be accessed by external services. This is achieved by making each boundary object an entity that can be uniquely addressed, queried and interacted with, within the larger web ecosystem.

Emergent Structure

The HMS model frames users as owners of their problems and puts them in charge of solving them. Users are expected to evolve the system over time via iterative processes of appropriation and restructuring (Fischer et al. 2001). As new features become available within other HMS environments or external services, they can be integrated in the work practice and in the current environments.

Roles are initially non-existent, yet they can be brought into play as access lists for specific environments. Design and meta-design are accessible through the whole creation process as all boundary objects are open to inspection and modification. The very process of tailoring is handled on the same logical level: design tools and older versions of an object are available as boundary objects - open for inspecting, discussing and tweaking.

Each CoP is self-contained but interlinked with the others, allowing both local policies and global connectivity. The HMS model allows for tweaking and experimentation by exploiting the mediation mechanism. Mediation agents and the environments that host them are both represented as customizable boundary objects. Therefore each community can follow local rules of interaction and communication by modifying their environment and mediation agent. This decentralized control is meant to support global behaviors emerging from the interactions of the local communities.

4.2.7 SER Model

The SER model (Fischer et al. 2001; Fischer 2009b) is another inspiration for the HMS model. The underdesign principle provides opportunities to elaborate of existing artifacts at use time.

Underdesigned representations of solutions are preliminary, incomplete or imprecise specifications, through which designers and end-users are inspired to explore variations or to further advance their ideas. The HMS model does not provide out-of-the-box collaboration patterns, but simple seeds that act as tools and communication means. As noted before, rich interaction and complex collaboration are based on the social construction of simple elements. Boundary object are simple but as seeds they have the potential to be reused, combined with other artifacts and evolved to better support situated problems. Nevertheless, more complex elements are over-specialized and are less prone to differentiation and appropriation.

An HMS implementation can be seen as a collection of seeds; therefore the SER process occurs not merely on a macro level, i.e. system evolution, but within the finer grained components of individual boundary objects that can be evolved in parallel influencing one another. Specifically, the HMS breaks down the initial system into smaller seeds, and therefore each can be inspected, adapted and evolved continuously, blurring the distinction between design time and use time. The evolutionary growth phase and reseeding phase are tightly coupled together within the same system and each nugget can be seen as reflecting the SER process. Section 9.7 provides concrete examples to show that the system is open and flexible enough for reseeding and meta-design in use time.

4.2.8 Integrated HMS Model

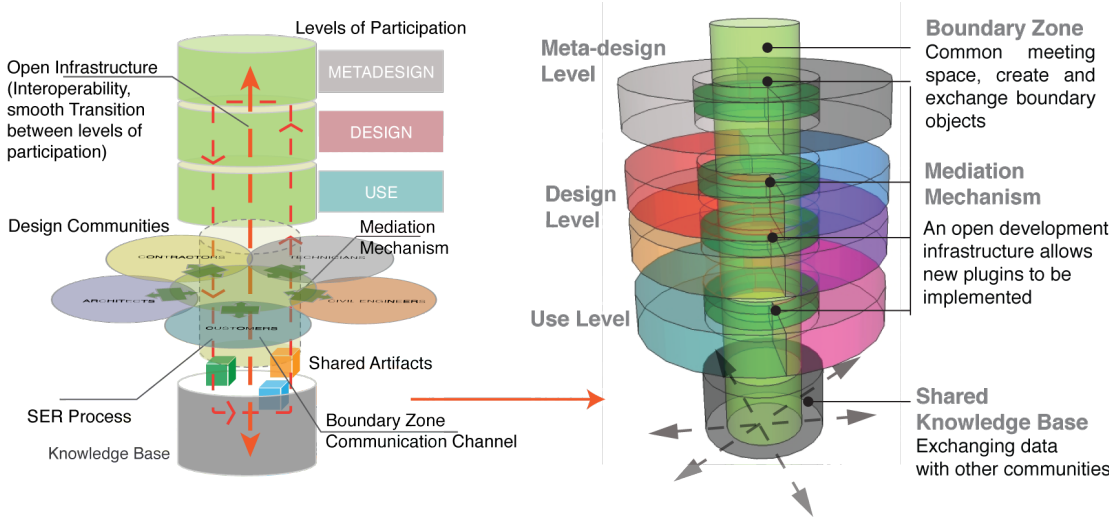


Figure 31 The HMS model integrating different aspects

The HMS model emphasizes the interplay between different CoPs either as individuals or as members of specific CoPs and CoIs; between CoPs and design artifacts; and between different design activities. As such, it provides the means to allow CoPs to construct communication and other collaboration aspects over time. In this case, the HMS is also a process model, as its entities such as boundary objects, boundary zones and environments are not static, but describe sets of interactions. Figure 31 illustrates the result of bringing

together the different attributes of the HMS (on the left) and framing them into a core integrated model (on the right).

Table 2 Framing theoretical concepts

Theory (Concept)	**HMS conceptual model (Model)**
Design communities (Wenger 1998; Fischer 2004) SSW – workshops (Costabile et al. 2007b; Barricelli et al. 2009b; Costabile et al. 2006a) Habitable environments (Alexander et al. 1977; Borchers 2001)	Habitable environments (virtual)
Boundary objects (Star and Griesemer 1989). (Evolving artifacts, Social artifacts) Software readymades (Mørch 2001)	Boundary objects (Continuously evolving, under social construction; means to support appropriation/bricolage)
Boundary zone (Konkola 2001; Mørch and Andersen 2009) Boundary crossing (Engeström 1999; Engeström 2001)	Boundary Zone (Communication channel)
Localization (Ardito et al. 2011; Barricelli 2010; Esselink 2000) Communication gaps (Snow 1993; Petre and Green 1993) Visual languages (Costabile et al. 2007b; Barricelli et al. 2009b; Costabile et al. 2006a)	Mediation mechanism
EUD – tailorability (Mørch 1997) Costabile et al. 2003b; Lieberman et al. 2006) Meta-design (Fischer and Scharff 2000b; Fischer 2010; Fischer and Giaccardi 2006; Fischer et al. 2004a) SSW – three design levels (Costabile et al. 2007b; Costabile et al. 2006a) Collaborative design levels (Popovic 1996) Cultures of participation (Jenkins 2006; Fischer 2009a)	Different levels of participation Different levels of tailoring
Meta-design – open ended software environment (Fischer et al. 2004a; Fischer and Giaccardi 2006) Appropriation (Bentley and Dourish 1995; Wertsch 1998; Pipek 2005) Bricolage (Lévi-Strauss 1968; Ciborra 2002)	Open infrastructure
Underdesign (Fischer et al. 2001) Co-evolution (Dorst and Cross 2001; (Nielsen 1993; Bourguin et al. 2001) Design-in-use (Henderson and Kyng 1991) Socio-technical environment (Fischer and Herrmann 2011) Evolutionary application development (Mørch 2011a)	SER model

Like SSW methodology, the HMS model supports three different levels of participation. The levels of participation in the HMS model however are different modes of interaction rather than different places. Whereas in the SSW participants are tied to their predefined role and accordingly to a specific workshop, in the HMS model participants can act as meta-designers,

designers or users, deciding moment-to-moment how to represent and interact with artifacts. In the HMS model, different CoPs exist at the same level at the same time and in a peer relationship, although they work on different issues. Each CoP can be accessed according to the three modes, allowing the potential for self-evolution of every community. The SSW hierarchical network is still possible in the HMS, but it is the result of an emergent social organization rather than system predefined.

Different CoPs have their own environment, which follows their working habits and isolate the possibility of interference with other CoPs' work. The HMS model envisions a mediation mechanism that allows the same boundary object to be represented differently within individual environments. CoPs have open access to the mediation mechanism, and are empowered to evolve it over time.

The central boundary zone in the HMS model, serves as a communication channel, within which design communities can create and exchange boundary objects. All other CoPs can access and use these boundary objects as well as jointly construct new boundary objects to be stored in the shared knowledge base. Boundary objects are utilized to mediate communication among the different communities. They are software artifacts and can be seen as social *application units*, as they are continuously under social construction, can be evolved over time and encourage appropriation.

The open infrastructure emphasizes the openness and transparency of the system, exposing rationales and design logic of the system to support users to inspect, appropriate and tinker with it. The transparency of the environments and design modes ensure a smooth transition between levels of participation. As discussed in Section 2.4.3, appropriation and bricolage are situated interventions, which require "*open*" and "*flexible*" systems that can be hacked and used in unexpected ways.

Additionally, the open infrastructure is not only about the internal HMS, but also about interoperability, being able to exchange and utilize external resources. The open infrastructure is essential to support the SER. The SER model presents a collaborative design process and highlights the underdesign principle, providing opportunities for evolution and design space to create solutions at use time rather than anticipating all the solutions at design time. Table 2 outlines the connections between the various theoretical concepts and the HMS model characteristics in detail.

In summary, the concepts of boundary zones, boundary objects, environments, the mediation mechanism and levels of participation, open infrastructure and the SER process model provide a design context that users can build upon rather than a comprehensive fully equipped finished system. Notably, this model reflects an opportunistic bottom up approach to leverage situated action and the breaking down and restructuring of roles, social structure and design levels.

4.3 HMS Novelty and Contributions

This section describes some of the novel approaches and contributions brought by the HMS model, in particular the contribution to SSW methodology, and how the HMS model fosters social creativity and appropriation. The HMS model brings together a number of theoretical concepts into practical models and guidelines for building concrete systems, as will be shown in Chapter 5 and Chapter 6.

Contributions to the SSW

The HMS model expands SSW methodology in new directions by generalizing some of the SSW features, allowing users to contribute with more variation and design in-use.

Table 3 SSW and HMS features

SSW	HMS
Communication follows hierarchical orders	Introducing a central communication channel, serving as a boundary zone that allows the exchange and management of boundary objects. All the CoPs can access the boundary zone, and send their requests as well as dispatches to other CoPs.
Three fixed levels - meta-design, design and use level	An open under-development infrastructure, thus further levels or new CoPs involved in the collaborative design process could be added to the network
Distinct separation among roles such as developers, managers, designers, users	The boundaries between the roles are blurred, and new highly dynamic roles emerge
No mediation mechanism	Introducing mediation mechanism into the model to localize exchange boundary objects in a meaningful way
Domain-oriented workshops without considering device and culture differences	Habitable software environments considering not only end users' role, but also their cultures and devices in use.
Annotation tools as a way of communication among CoPs, which has to follow a hierarchical order	Boundary objects as building blocks allowing CoPs to create their own boundary objects and situated solutions and enhancing communication among CoPs
End users influence on the deeper system power is very limited; there is a big gap between surface and the real application.	Making mediation mechanism accessible to end users a to empower them to create appropriate information representations in the context
Predefined social structure	Meta-design, design and user levels are emergent social structure. Flexible degrees of involvement in the design process with allow the tendency to shift control from developers to users as co-developers.

Table 3 provides an overview of the SSW features and their HMS equivalent. For instance, when I started working on the HMS model, the three different levels of participation of the SSW were considered to be three different environments. As I developed the HMS prototype I came to realize how the technical implementation enabled a greater flexibility than I expected and I started exploring design and meta-design as modes of interaction instead of different

places as modeled in the SSW. It is still possible to assign different modes of interactions to different environments as in the SSW, but it becomes a consequence of an emergent social structure rather than the default assumption. The breakdown of the three levels of participation destructs the concept of roles, whose boundaries become blurred. Users can therefore switch between different levels and different perspectives and new roles can emerge and change dynamically on a social level.

The introduction of the boundary zone resulted in further exploration of the exchange and management of boundary objects. Consequently, the annotation tools of the SSW became the same class as the objects (e.g. interface elements) that they annotate, since the model deals with them as normal boundary objects, instead of as a different class of objects devoted to communication.

The flexible exchange of boundary objects raised the question of how to make sense of them in different contexts, which led to developing the concept of an adaptable – rather than adaptive – mediation mechanism.

Extending Boundary Objects

As mentioned in Section 2.3, the boundary object concept needs to be refined to satisfy emergent information and new situations arising during the collaboration process. HMS boundary objects extend the boundary object concept in respect of a social aspect and a situated aspect. For what concerns the social aspect, boundary objects are emergent and defined by how they are used: they are a medium, shaped and reshaped by social interaction among different design communities. From a situated perspective, they embody the meta-design concept as an infrastructure for unexpected use and creating situated solutions for emergent issues. In many ways it is the situated perspective that makes the social emergence perspective possible within the context of a technical implementation, placing boundary objects firmly both as sub-processes of the SER model and being subjected themselves to the SER process.

How collaborative software architectures support the construction of boundary objects will be further discussed in Chapter 6.

Fostering Social Creativity

The HMS model is designed to enhance social creativity through the following considerations:

- Encouraging diversity and user-driven innovation, in the sense of bringing together design communities from different disciplines as well as involving end users in the design process;
- Allowing independence (Surowiecki 2005) since the architecture of the HMS model is globally interconnected and locally controlled (Kapor 2006);
- The collaboration of CoPs is decentralized, hence design communities are able to specialize and draw on local knowledge (Anderson 2006);

- Providing means to allow CoPs to be in control of their design problems, to create convivial tools (Illich 1973) as well as evolving them over time in the use context;
- Supporting knowledge aggregation, making knowledge available to all the communities, thus turning individual contributions into collections;
- Evolving a complex system with a bottom up approach and from meta-designed boundary objects, thus encouraging appropriation and bricolage.

Innovation and creativity support in the HMS model is twofold: on one hand it guides users to design innovative computational environments, and on the other hand the open computational environments support social creativity.

Infrastructures for Bricolage and Appropriation

The appropriation infrastructure described by (Stevens et al. 2009) aims to provide extra communication channels between developers and end users as well as between end users themselves to support tailoring and appropriation activities.

The HMS model extends the domain of the tailoring mechanisms to the communication channels, building a fully social and reconfigurable system model. The HMS model goes beyond the concept of software reuse and seeks to provide inhabited spaces, tailorable via a meta-reflective system, co-evolving with CoPs in the social context.

Notably, in the HMS tailoring is also applied to the processes that underlie social organization.

4.4 Exploring Implementation Options

Any implementation of the HMS model should support the basic HMS concepts outlined in this chapter. Levels of participation, environments, mediation mechanism and support for boundary objects creation and tailoring are the main criteria to qualify as an expression of the HMS model. In this section some opportunities in the CSCW landscape could be applied to the HMS model.

4.4.1 Collaborative Virtual Environments

In a Collaborative Virtual Environment (CVE) the participants can experience a shared virtual reality within which they can see each other as 3D graphical representations called avatars. It is through their avatars that the participants communicate and interact in real-time with each other and with the virtual world, ideally replicating on their day-to-day social dynamics. The virtual environment provides a context for the sharing and manipulation of digital constructs that are relevant to the communication.

CVEs aim to tackle some real-time issues of CSCW applications by providing a real world metaphor for social support and intuitive use of the system that leverages our day-to-day spatial model of social interaction. (Benford 1994; Snowdon 1995; Benford et al. 1998) present a comprehensive overview of the major aspects of research in CVEs.

A case for the use of CVEs as opposed to CSCW media-spaces has been illustrated in (Hindmarsh et al., 1998). The differences between the two fields seem to be significant enough to consider some of the issues of CVEs separately from the rest of CSCW (Benford 1994).

Some of the distinctive characteristics of CVEs are summarized below:

- Explicit User Representation: in CVEs every user is explicitly represented for the benefit of the other participants. The embodiment, or avatar, shows the other participants both the focus of action and the focus of attention of the user (Dourish and Bellotti 1992; Benford 1994; Benford et al. 1995; Dourish 1997). The focus of action can be represented as actions performed by the avatar on objects in the world. The focus of attention can be inferred by the avatar's position and head orientation.
- First Person View: the user usually observes the world as if seen through the eyes of his avatar. This is necessary to keep the expectations of the other participants on what the user sees congruent with the user's real perceptions (Fraser et al. 2000).
- Spatial Metaphor: objects and avatars exist within a world, a 3D spatial metaphor used for the virtual environment. The space is where the action takes place: it is used for socialization, negotiation of awareness, navigation between worlds and as a place where applications can be situated.
- Rich Interaction: the participants should be able to communicate in a number of ways, covering a large range of I/O devices and interaction techniques. Another important aspect of avatars is that they support mutual awareness.

CVEs rose to popularity with platforms such as Second Life, attracting to 21.3 million user by 2010 (LSL 2003):

"Residents can explore the world (known as the grid), meet other residents, socialize, participate in individual and group activities, and create and trade property and services with one another. [..] Built into the software is a three-dimensional modeling tool based around simple geometric shapes that allows residents to build virtual objects. There is a procedural scripting language, Linden Scripting Language, which can be used to add interactivity to objects. Sculpted prims, mesh, textures for clothing or other objects, and animations and gestures can be created using external software and imported." (LSL 2003)

Users can easily author the 3D world with direct manipulation of interfaces and can rise to the role of meta-designers by scripting behaviors in the objects they create by using the Linden Scripting Language, an event-driven programming language (LSL 2003).

While Second Life would provide an excellent starting point to experiment with the implementation of HMS concepts and support EUD, the underlying platform is not open source yet, making it impossible to implement many of the HMS concepts. Developing our

own 3D shared virtual environment platform – or adapting an existing one - is a huge task that requires mastery of many different technical domains.

4.4.2 Company Intranets and Collaboration Platforms

Company Intranets were one of the first alternate uses for world wide web technology back in the early 1990s, and were widely used as bulletin boards with some user-publishing features (Jones 2006). Intranets are about sharing resources, communication and collaboration. The idea of bringing web experiences in-house is appealing; however integration with existing systems is rather difficult and platforms are often not open to programming.

A few of the most flexible intranets are based on wikis. For instance, TWiki is a flexible, powerful, and easy to use enterprise wiki, enterprise collaboration platform, and web application platform (Thoeny 1998). However, they tend to be over complex, not open to scripting, and use mostly server-side technologies such as Java.

4.4.3 Excel Spreadsheets

Spreadsheets have been very successful and widely used in supporting end users to program their own applications (Hutchins et al. 1986; Lewis and Olson 1987; Nardi and Miller 1990, 1991; Nardi 1993). The main reason for their success is that the languages primitives are the application-level primitives with which users are already familiar (Nardi 1993). The spreadsheet is expressive of a great deal of rich domain knowledge, since its complexity lies in the relationship between entities in the domain itself, rather than in the programming needed to create the formulas that model the relationships.

Spreadsheet programs like Microsoft Excel have fulfilled these properties:

- End-users can model and program in Excel and expect the spreadsheet to perform the computations in real time or near real time, independently of whether the model is completed.
- Notably, there is no crisp distinction between an application design-time and runtime in Excel, since both views are integrated into one.
- Excel eliminates the steep learning curve of traditional programming languages (Lewis and Olson 1987) by providing a greatly simplified control structure. The key to providing program control for end users is to keep control constructs simple (Nardi 1993).
- Excel does not crash; instead cell values keep being computed. Some cells may provide nonsensical values, which however is decided by the end-user adjusting the output of the cell value (Anslow and Riehle 2008).

Spreadsheets as a EUD platform have many distinctive characteristics and they are a compelling case of a successful platform for use, design and meta-design. The recent widespread adoption of the Google Documents platform and its shared spreadsheet

documents opens up new possibilities for a shared work environment subjected to EUD and an evolutionary approach.

However, spreadsheets are based on a tables and numbers paradigm. It is therefore hard to extend them in new directions, and in particular to use their language to express new communication channels, domain concepts and new representations. The complexity in creating a shared programmable spreadsheet platform – and the difficulty in finding an open source alternative – is also a barrier to an otherwise potentially effective research direction.

4.4.4 Wikis

In contrast to spreadsheets, wikis are not constrained to a specific computational paradigm or a specific visual display (Anslow and Riehle 2008):

- Wikis encourage a culture of participation, since they enable users to share and develop knowledge from a wide range of domains. Wikis provide bottom-up knowledge production environments giving everybody a voice and providing benefit to those who do the work (Grudin 1989; Grudin 2006). Cunningham suggests that wikis are useful tools for building CoPs (Leuf and Cunningham 2001).
- Wikis have an open structure and foster social creativity. The ease of accessing a wiki in a browser and its openness can be seen as two of the main success factors of wikis. Wikis paved the way to a writable web (Désilets et al. 2005).
- Wikis allow incremental knowledge creation and enhancement. A wiki can be an ideal platform and format to collaboratively build up knowledge or perform design activities among diverse design teams (Schadewitz and Zakaria 2009).
- Wikis have been shown to express emergent organization and a degree of meta-design via communities using them to discuss how the wiki itself should evolve, making the wiki both the product and the medium of communication.
- The distinction between design time and use time is blurred. Wikis are an excellent starting place to seed future opportunities for learning and growth (Gordon 2006).

Note that to support the meta-design level's tailoring, end-user programming is necessary. As such, wiki engines fit these assumptions.

- A wiki page is always in a consistent state, i.e. "the markup always parses," even the results of a page may not make much sense. It is always under development.
- The wiki page can serve as a computation place. The traditional distinction between edit and view mode, or "design-time" and "run-time" is blended by the latest breed of wiki WYSIWIG editors.
- Like spreadsheets, wiki pages do not crash. If a wiki page does not make sense, the wiki engine parses what it can interpret and falls back to text display (Anslow and Riehle 2008).

Because of this research aim, wikis, which are by nature not domain-specific, were chosen. They are thus better for enabling end users to carry out collaborative design at varying granularity and for supporting different modes. The software solution focuses on embodying the HMS model in all its integrated parts rather than through a single element, such as boundary objects. To address the boundary-object concept alone, many other alternatives (e.g. mindmap) might have been used as a starting point.

4.5 Conclusions

The HMS model presented in this chapter is a meta-design conceptual model that addresses social creativity in the context of collaborative design. It allows participants to exchange boundary objects and use them to shape their environments. The HMS defines the environments and builds the medium of communication out of these very same boundary objects, thus allowing flexible reconfiguration of the basic collaboration processes.

The medium is defined as an open mesh of boundary objects amenable to varying gradual levels of tailorability and integration with their surrounding environments.

Additionally, the HMS model presents a number of processes and structural features that provide sufficient detail to act as guidelines for setting up a technical platform (Section 4.2). Chapter 5 introduces MikiWiki, a working wiki-based implementation of the HMS concepts. Chapter 6 reasons both on the HMS and on the MikiWiki implementation to extrapolate a detailed reference architecture. This architecture can then be used as the blueprint for a collaborative platform that will support situated, fine-grained co-evolution.

Chapter 5

The HMS model has been implemented in order to test it in the field and observe the presence of these postulated qualities. This chapter first introduces several related concepts: existing wikis, missing wiki features, and why wikis can be a good starting point to prototype the HMS model. By prototyping the HMS model, it provides opportunities to evaluate it and to gain more insights in how to enable collaborative design-in-use as well how to better bring creativity and collaborative design together.

5. MikiWiki

In order to evaluate a meta-design model and provide concrete guidelines for implementing it, I implemented MikiWiki (Zhu et al. 2011b) as the HMS model prototype. ‘MikiWiki’ stands for 'meta-wiki' and indicates the reflective properties of the system, e.g. adaptable to different situations and being able to evolve.

Wikis are a collection of pages that can be edited by anyone, at any time and from anywhere. They are a popular format for sharing knowledge in both academia and professional domains (Leuf and Cunningham 2001).

I used a wiki architecture as a base, since the wiki architecture matches many conceptual aspects of the HMS model. MikiWiki leverages some features of a regular wiki - namely collaboration, rich context, openness and dynamic links. Beside normal wiki functionality, MikiWiki extends wikis with development functionality, using a macro-like system to change and evolve the system along with collaboration practice.

MikiWiki aims to demonstrate an open socio-technical system based on the HMS model, with which users are empowered to perform collaborative design as well as to tailor their environments, communication, coordination, and all aspects of collaboration in a more fluid way according to situated problems.

In MikiWiki, three different levels (meta-design, design and use) can be accessed by users. Some pages can contain code, and thus advanced users can change the behaviors of existing artifacts or create new ones from within the wiki, an example can be seen in Section 5.5.

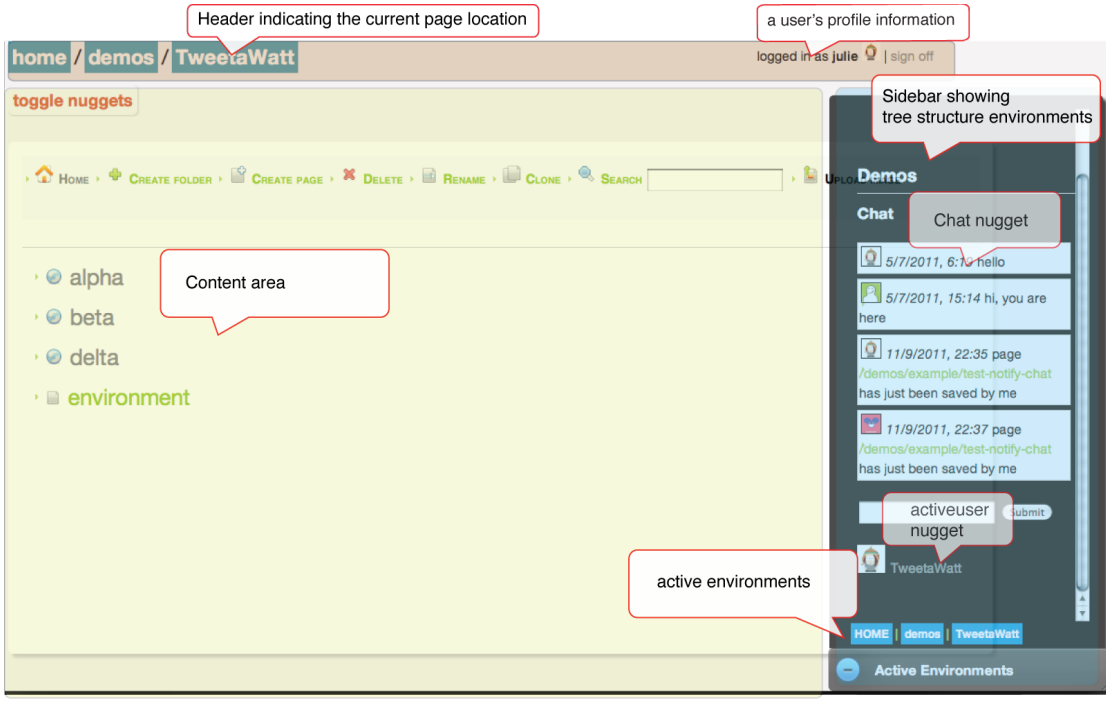


Figure 32 MikiWiki

A MikiWiki page consists of four parts, a header to indicate the current page path and a user's profile information, a content area and a sidebar showing the active environments or nuggets that are made available to the whole environment (Figure 32). Habitable environments examples are discussed in Section 5.3.2 and Section 5.3.3.

MikiWiki provides a common collaboration context across the system and opportunities for design communities to build domain-oriented environments where they can work while being aware of the activities of others. Beyond providing tools for text content production like traditional wikis, MikiWiki allows all the stakeholders to collaborate in practicing design, to modify and create new software artifacts, and to continuously evolve the whole wiki system.

5.1 Mapping the Hive-Mind Space model to MikiWiki

Table 4 depicts how the HMS model derived from various theories and concepts, how its characteristics correspond to MikiWiki system features.

A habitable environment can be seen as a folder, but with an environment page - as it will be explained in Section 5.3. In the environment page, users can specify certain behaviors and attributes that apply to all pages in the environment.

Boundary objects can be coarsely mapped to nuggets in MikiWiki. However, what could be seen conceptually as a single boundary object, could be composed of several nuggets on a technical level. An aggregated *videopage* nugget (composed by three nuggets) example can be seen in Section 5.4.

Table 4 Mapping between the HMS model and MikiWiki

Theory (Concept)	**HMS conceptual model (Model)**	**MikiWiki (System features)**
Design communities SSW – workshops Habitable environments	Habitable environments	Folders, Environment Page, Lookup mechanism
Boundary Objects (Evolving artifacts, Social artifacts) Software readymades	Boundary objects	Nuggets (Social application units)
Boundary Zone Boundary Crossing	Boundary Zone (Communication channel)	Accessible pages, open environments (folders accessible by design communities)
Communication gaps Localization Visual languages	Mediation mechanism	Format page, environments and Lookup mechanism
EUD – tailorability Meta-design SSW – three design levels Collaborative design levels Cultures of participation	Different levels of participation Different levels of tailoring	Meta-design level: design environments, creating format page with JavaScript editor Design level: use design environment, browsing, editing visualization pages, data pages and format pages with JavaScript editor or rich-text editor Use level: browse visualization pages, creating visualization pages with rich-text editor
Meta-design – open ended software environment Appropriation Bricolage	Open infrastructure	End-user development approach to allow client-side programming and programming by examples Enabling flexible switching between different design levels Extensibility to the existing Web ecosystem
Underdesign Co-evolution Design-in-use Socio-technical environment Evolutionary application development	SER model	Providing just enough features to be useful, and at the same time leaving code short and simple to be quickly understood and modifiable so that the set of features can be easily extended.

The whole externalized design process of MikiWiki is presented in table 6: stage 1 (Concepts), stage 2 (Model) and stage 3 (System features). It extends Mørch's model of externalized design of user interfaces with an intermediary conceptual model stage (Mørch 2011b).

Accessible open environments in MikiWiki can be seen as the communication channel. The Mediation mechanism and supporting different levels of participation and tailoring are reflected in MikiWiki. If nuggets in MikiWiki are the analog of boundary objects in the HMS model, then the collection of nuggets and shared MikiWiki pages can be thought of as the analog to the knowledge base.

This is not an exact one-to-one mapping, as many theoretical concepts, such as boundary objects, communication channel and so on cannot be reduced to a simple software system component. The following section will describe how MikiWiki reflects on the HMS model features in detail.

5.2 Nuggets

MacLean suggested that a more incremental approach is desirable to allow end users to express their customization requirements as much as possible using skills they already possess, and to equate increases in customization power with proportionate increases in the level of expertise required (MacLean et al. 1990). I used a component-based software development approach to incrementally implement MikiWiki.

5.2.1 Nuggets as Building Blocks

In analogy to Lego construction kits, providing simple parts with which a user can create complex artifacts (Resnick et al. 2005), nuggets are the building blocks of MikiWiki within and between stakeholders. Nuggets, as the basic components of MikiWiki, are independent from each other and can be used to create new tools or services (Figure 33).

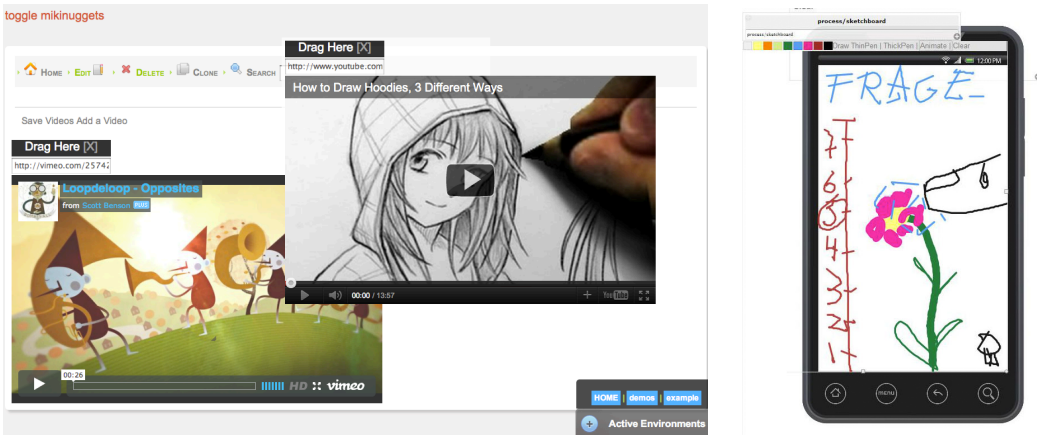


Figure 33 The *videonote* nugget (left) and the *doodle* nugget (right)

Nuggets are concrete boundary objects and embody meta-design principles. They are designed to enable meta-design in use time, as each aspect of a nugget as a wiki page can be accessed, tailored and socially evolved over time, rather than remain predefined and unchangeable. For example, Figure 34 illustrates a *wall* nugget embedded in a MikiWiki page with font color in black against a transparent background. If one wants to use this nugget in a page with a dark background color, one would not be able to read text entries and need to change the font color to white. Being able to access the wall nugget scripting code, one can not only modify it easily to solve the color problem but also add new behaviors, e.g. enabling it to be draggeable or resizable.

5.2.2 Integration of Multiple Perspectives

Similar to the 'multiple representations' of application units (Section 2.2.4) each nugget has three different intertwined perspectives, respectively visualization, format and data representations. Figure 34 presents a *wall* nugget from three perspectives. The *wall* nugget was designed after the Facebook wall, by which users can post their status messages and comment on their friends' status entries. The data page contains the wall data in JSON format; the format page defines how to represent the JSON data in JavaScript; and the visualization page embeds the macro-like code that expands to the final visible nugget. All the representations are wiki pages and thus can be easily edited.

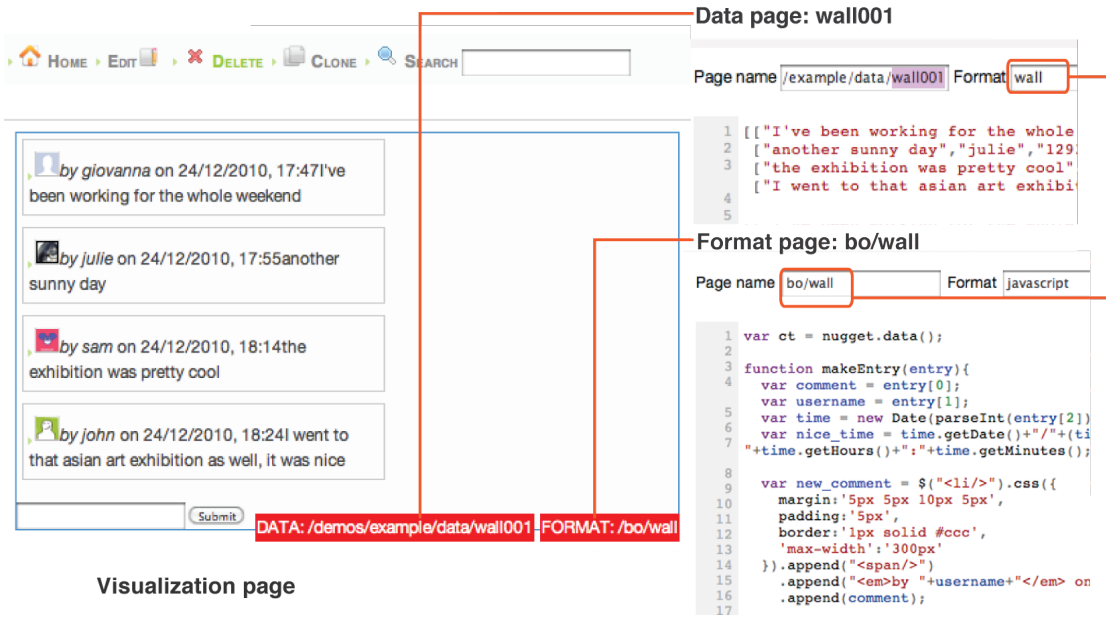


Figure 34 Three aspects of a *wall* nugget

To a certain extend, these perspectives could be associated with different roles. For instance meta-designers could access format pages and data pages to modify nuggets behavior, designers and users mainly use and access the visualization pages. In some cases, designers might want to access the HTML or Cascading Style Sheets (CSS) parts of the format pages to adjust visualization pages "look and feel".

As a comparison with application units, the visualization page can be seen as a *presentation* object and the format page as *implementation* code. The *rationale* can be documented within the format page as comments, which allows the designers to reflect upon and to support others to understand the application (Mørch 1996). The *literate programming* techniques introduced by (Knuth 1983) could be used to mesh *implementation* and *rationale*. Nuggets open up a further perspective – namely, *data representation* for users to inspect, tinker and extend.

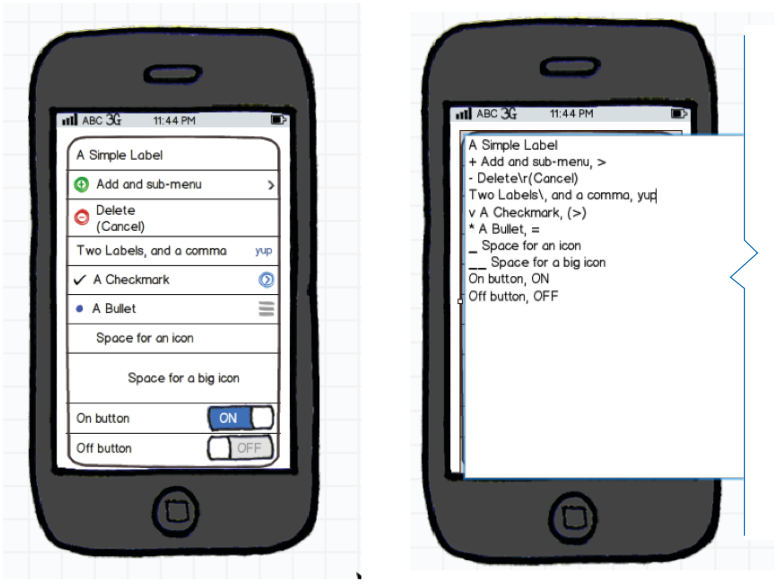


Figure 35 A component and its underlying data in balsamiq

The inspiration for the data pages in MikiWiki was the user interface of (Balsamiq 2008), a popular professional web application used to produce web interface layouts. Figure 35 shows how Balsamiq handles data and presentation. Visual components that allow variations can be double-clicked, to access the textual data that underlies the displayed information. Every component uses its own markup language to assign different meaning to different parts of text. For example, in the context of an iPhone screen component, a capital “ON” or “OFF”, is rendered as a switch button, while a “*v*” at the beginning of a new line is interpreted as a checkmark.

This concept was further extended by allowing users to access the code that interprets the data and maps it to a visual interface. This mapping code resides in the *format page.*

At the design and use level, users can utilize different nuggets to externalize and represent their ideas as needed at different design phases. Therefore nuggets can intertwine the various viewpoints of different users as well as bridge various design phases. Figure 36 demonstrates users designing a mobile application using various nuggets - i.e. *note*, *sync-imagenote*, different *toolbox*, *canvas* nuggets.

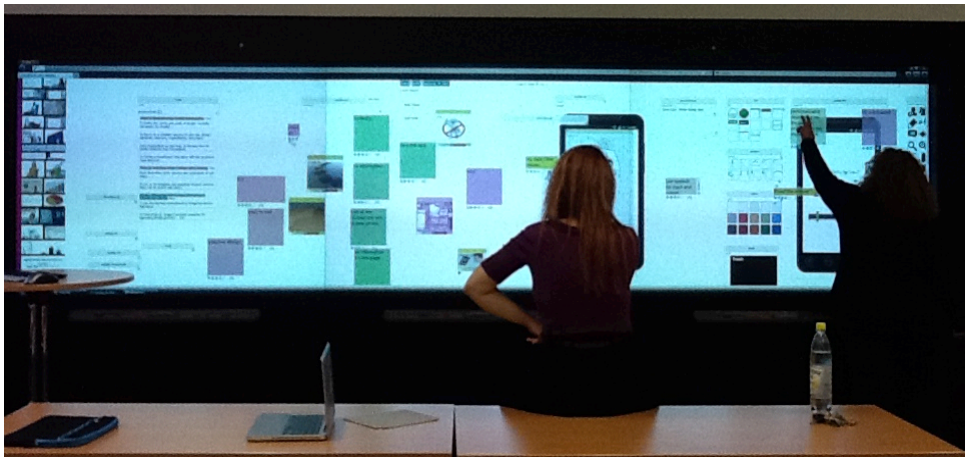


Figure 36 Designing a mobile application with various nuggets

The novelty of nuggets is the following:

1) One distinctive feature of nuggets is that they are independent from each other, rather than integrated with each other by means of input/output ports as in most CBSD systems, for instance in the work of (Stiemerling 2000) and (Wulf et al. 2008). In this case, nuggets and application units are similar, as they explore a bricolage approach to allow users to define interaction among "standalone" components in situ (see Section 2.4.4).

2) In addition, nuggets extend application units towards web application environments. In contrast to conventional (pre-web) applications, nuggets are distributed and can be modified as well as evolved by collaborative interaction.

5.2.3 Taxonomy

Chapter 3 discussed how and why certain attributes were chosen as boundary object building blocks. Based on those building blocks, a set of initial nuggets have implemented for instance rank, comment, todo, annotations, doodle, notification, online presence, chat, video embedding, access control and so on. The initial set of nuggets has substantially extended and new nuggets have been created over time along with design studies and experiments. To support collaborative design, nuggets address it from different aspects - for instance communication, coordination, history tracking, enhancing awareness and so on, as well as different modes, co-located and distributed activities, and asynchronous and synchronous approaches (Table 5).

However, this categorization is not fixed; rather, it depends on the usage context and takes account of users' situated action, for instance using a *todo* nugget as a message board to post questions or answers and discuss them. Thus, a more accurate clustering needs to take account of usage context and users' situated actions. More examples of users appropriating nuggets and evolving them to their design activities will be discussed in design studies.

Since nuggets do not have fixed use context, users are encouraged to further appropriate the system and assign new meanings to them according to different cases. Nuggets therefore act as both a mechanism and interface for supporting the creation and evolution of software artifacts beyond their initial form. They are a medium made of captured knowledge, as CoPs incrementally construct knowledge via nuggets during collaboration and communication. More detailed information about nuggets is included in the Appendix A.

Table 5 Nuggets for supporting collaboration from different aspects

Content	**Communication**	**Awareness (asynchronous)**
video - include YouTube or Vimeo videos **file** - include pdf or ppt files **css**- include external css **website** - include external website **expand** – include another MikiWiki page **hide** – hide certain pages or folders	**comment** - anonymous **wall** - username + time **chat** - real-time **skype –** call, video conference	**lasteditor** - show last editor of page **notify** - to people **notify-chat** – notify changes as chat entries **tooltip** - show page or folder content **showTag** – show tags associated to each page **folder** – show a folder's content & associated meta-data information
Coordination	**Annotations**	**Awareness (real-time)**
todo - edit items **task-table** – advanced todo list	**doodle** – drawing **note** – create PostIt notes **sync-note** - real-time annotation **imagenote** - images as notes **sync-imagenote** - real-time creating images notes **panel** - pages as notes **videonote** - videos as notes **flickr** - text to Flickr images **vote** - multiple voting **opinion** - vote with username and time **tag** – keywords	**profile**:mike - show profile picture of a user **user** – show users profile information and Skype call **activeuser** - name, pic, pages **activeuser**:name - show name only **activeuser**:page - show who's on which page **activeuser**:pic - show only thumbnail **activeuser**:who-is-here - show pics of people on this page
Localization	**Authentication**	**History**
translation-menu - a set of languages to choose from **autotranslate**:it to specify a certain language to translate (en, fr, de, it, zh, jp..)	**allow**:julie,designer,mike+r,accountant+r - allow access only to some users. +r stands for read-only. **allow**:all+readonly - only the admin can edit this	**text-diff**: show differences between two versions **history**: show versions

5.2.4 Nuggets in Use

Figure 37 shows a screenshot of a page containing a description of a set of role-playing game's characters and three nuggets - namely, note, tag and notify nuggets. The <<*data-page-name* as *nugget-name*>> and <<*nugget-name*:*parameters*>> syntax demonstrate the way to include nuggets.

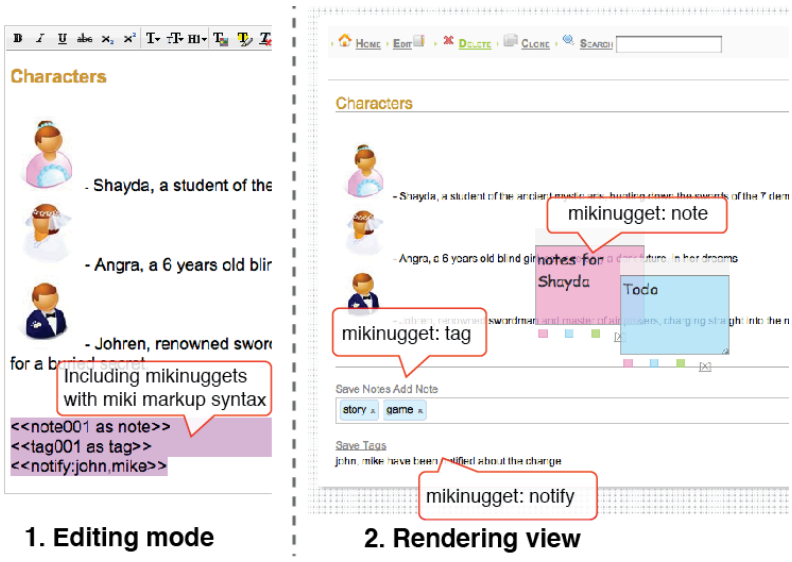


Figure 37 Embedding multiple nuggets

For instance, to include a note nugget, one uses double angular brackets and to specify <<note001 as note>>, "note001" is the *note* nugget data page name, and "note" is the *note* nugget format page name (Figure 38). The format page provides instructions of how to display videonote001 data page. Chapter 7 will explain this in more detail. Another way is to specify parameters as shown in Figure 38. To set permission for a MikiWiki page or for an environment, one can use the *allow* nugget, <<allow:designer>> to indicate that only users with a designer role can access this information.

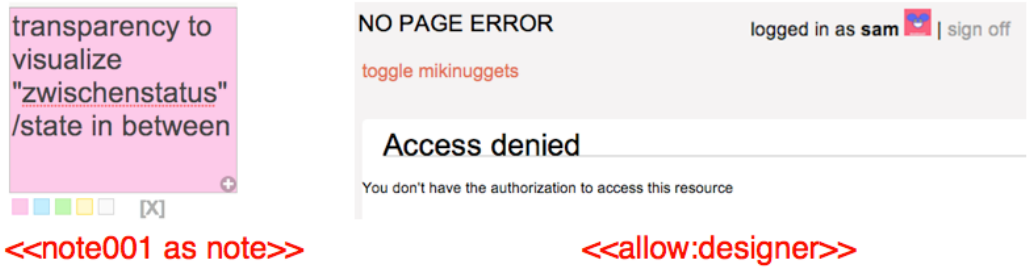


Figure 38 Basic syntax to include nuggets

Figure 37 illustrates how to mix multiple nuggets. In this case, a player can create notes, add textual annotations and place them at any location on the page. Notes' color (pink, blue, green) can indicate the importance of the annotation or annotation types according to player's need.

The *tag* nugget allows players to assign keywords to pages. Similar with embedding the *note* nugget, the syntax is <<tag001 as tag>>. The "tag" format page specifies how to visualize the tag001 data page. The auto complete tags can be predefined in a glossary file. Players can create their own tags.

The notify nugget allows players to pass user names as parameters to specify who should be notified via emails whenever the MikiWiki page is modified. <<notify:john,mike>> indicates that Mike and John will be notified by email when this page is modified. Nuggets enable individuals or CoPs to take control of their communication experiences.

5.2.5 Integrating Runtime and Use Time

The separation between user interface and application (the surface and the deep) restricts end users to simple manipulation of surface features, while the deeper system remains only accessible to developers (Dourish 1995). However, developers often do not know all the ways in which the system might be used by different end users over time (Saul and David 1994). Hence some lower-level details of system behavior should be available for customization at the user interface.

Nuggets are inspired and similar to application units (Mørch and Mehandjiev 2000). They are a means to provide a smooth transition between runtime and use time and support different levels of tailoring. The ease of accessing the deeper system and getting involved in different levels of participation facilitates collaborative tailoring as a whole.

By utilizing these nuggets, non-programmers can easily start using and remixing existing objects. Advanced users can create new nuggets and change system behaviors without moving outside of the use context, or modifying server side code, while experienced designers can create new nuggets if necessary.

In all cases, feedback is immediate, as the system is always running. With different building blocks, users can explore their design ideas, trying to combine different nuggets together. Regardless of whether users start working with or without strong ideas, nuggets provide them a basis and opportunities to explore as well as to build ideas while exploring.

5.2.6 Communication Channel

Communication channels are described in (Stevens et al. 2009) as various means to support communication between users as well as between users and developers (see Section 2.4.3).

With MikiWiki, the system itself acts as a communication channel, as all the communication, negotiation, development and social interaction are supported by the wiki platform. All communication is mediated and supported by tools, technologies, and platforms in the sense of via a “medium”.

More specifically, all the accessible environments, folders, nuggets, and common shared MikiWiki pages can be seen as a communication channel. Collaborative tailoring and evolving nuggets and all aspects of MikiWiki is the activity occurring within the communication channel.

5.3 Habitable Environments

In accordance with the HMS model, a flexible mechanism was designed to allow design teams to partition and locally configure communication. Environments are used as a way to associate specific behavior to a large set of pages.

5.3.1 Designing Habitable Environments

An environment is a folder containing an *environment* page. Any folder in MikiWiki can be promoted to an environment. Figure 39 demonstrates how to promote a folder "*neo*" to an environment by adding an *environment* page (full path – "home/demos/neo/**environment**") within the "*neo"* folder. In this environment page, one can specify some characteristics, i.e. setting the background color in pink and including a *rank* nugget, therefore all the other MikiWiki pages within this "*neo*" environment (a promoted folder) has a pink background and a *rank* nugget.

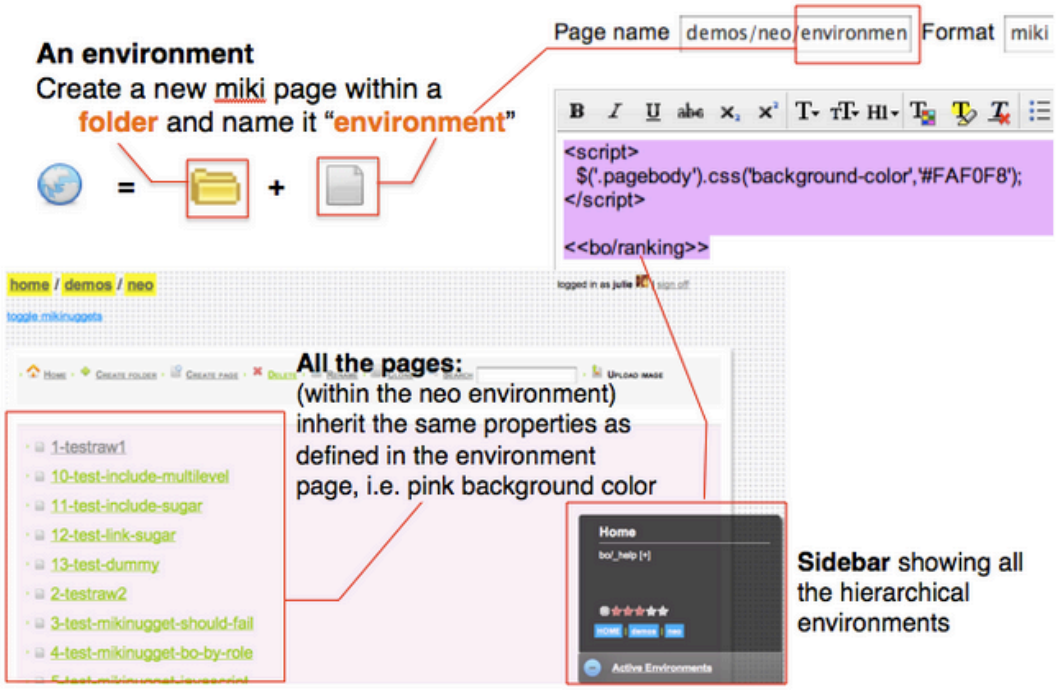


Figure 39 Creating an environment

Environments do not impose a predefined structure on all design communities, but allow the sharing of specific features among selected members. For example, access control in MikiWiki is not an inherent property of all environments, but it can be achieved by including an "*allow*" nugget in the environment settings page: all the pages within this environment therefore inherit the access control property. If an environment loads an *activeuser* nugget, all users that are accessing that environment become reciprocally aware of each other's presence as MikiWiki starts tracking user activities within the environment and displays which users are browsing that environment.

The sidebar shows the content of all the active environments: the current environment and all the upper level environments enclosing it. Since a sub-environment inherits all the characteristics of its upper level environments, the nuggets of the current environment and its parent environments are activated.

The simplified syntax of MikiWiki allows users to edit and create their own MikiWiki pages. For instance, frequent operations such as page linking and embedding use the following simplified square-bracket syntax **[[page-to-link]]** and **[[include:page-to-include]]**, rather than the more complex and semantically rich angular-bracket syntax

```
<<include:{'url':'page-to-link'}>>
```
and;

```
<link:{'url':'page-to-link'}>>
```
.

Pages can be organized hierarchically as 'folders' and 'documents.' A document can be text or a resource such as a picture or a PDF. Folders are used mainly to partition the documents into logical areas and for navigation, which can be further promoted to environments to have more tailoring possibilities.

5.3.2 Habitable Environments for Mediation

CoPs with different cultural backgrounds use different systems of signs, languages, devices and representations (Snow 1993) and may have different perceptions as well as interpretations even towards the same image. Communication is therefore needed to reach a common understanding about the messages they exchange, which is emphasized in the SSW (see Section 2.2.5.1).

Environments are also a mechanism to negotiate the awareness of divergent viewpoints regarding an object of interest by presenting it in a meaningful way to different users.

MikiWiki supports the mediation mechanism of the HMS model by allowing nuggets to be represented differently within individual environments.

Figure 40 shows three different environments, respectively alpha, beta and delta. Each environment has its own *twitterEnergy* nugget, which defines how to visualize the data coming from Tweetawatt (Fried 2011) in that specific environment. Hence, the same energy consumption data is visualized differently as a bar chart, a table and a spline chart.

The mediation mechanism works by allowing several CoPs to access the same data page, while locally overriding the definition of the associated format page within their own environments. The same data is accessed and modified by the two CoPs, while the way of representing and interacting with the data can be very different.

Making the mediation mechanism accessible to end-users empowers them to create appropriate information representations, designing their own environment in context and

optimizing their workflow at different design stages. The mediation mechanism is not a new concept, yet open mediation mechanisms for inspection and evolution in the design process have not been explored. Since the mediation mechanism is defined within a wiki page, it is open to modifications. Instead of gathering data about the end users' culture, role and device, and automatically customizing information for them, the open mediation mechanism goes beyond the "perfect personalization" dilemma (Pariser 2011) and enables the environment designers as well as end users to decide directly what is meaningful information and how to represent it opportunistically.

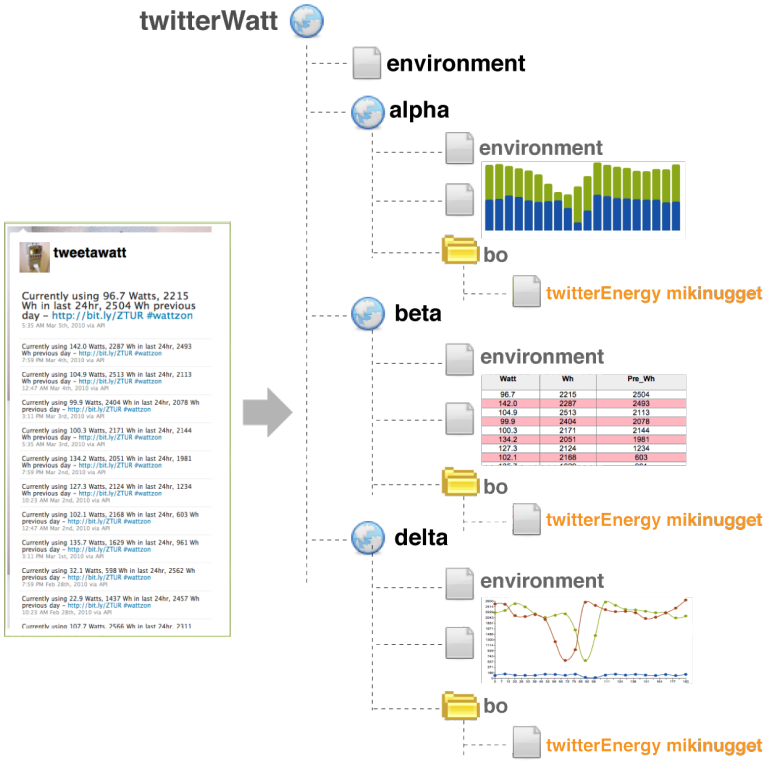


Figure 40 Three different power usage visualizations in three different environments

Additionally, environments are a mechanism to negotiate the awareness of divergent viewpoints, representing diverse points of view. Being aware of those differences eventually enhances the mutual understanding and supports CoPs' collaboration.

MikiWiki supports the mediation mechanism of the HMS model by allowing nuggets to be represented differently within individual environments.

5.3.3 Habitable Environments Inheritance

Some folders can be elected to be environments. An environment is a folder containing an *environment* page. This page contains behavior that is extended to all the pages of that environment. All folders and pages contained within the environment are therefore influenced by that behavior.

Environments can be nested. A MikiWiki for a university could have a University Environment, a Department Environment and a Class Environment. Each environment can add or inhibit those behaviors and characteristics that are useful to its inhabitants (e.g. chats, forums, and profiling).

Environment mechanisms allow users to locally control and build their rules in terms of their context and work practices. Sharing the same nuggets can globally interconnect environments. Nuggets however can be represented differently within individual environments. Each environment can redefine the rendering rules of a boundary object, to adapt the materialization process to the characteristics of the environment and its users.

Figure 41 shows the *foo* environment as a top environment. Within the "*foo/environment*" page, an *autotranslate* nugget is included. <<autotranslate:ja>> specifies to translate miki pages within this environment in Japanese. Therefore, all the pages, folders and sub-environments will be translated into Japanese. Within the sub-environment *プロジェクトの (project)* all the pages are in Japanese. This environment can also have its own characteristics, for instance, different background color, font size, chat box in the sidebar, and so on.

Figure 41 Environment localization

Supporting intercultural collaboration and communication is a scenario that could benefit from this feature. Chat data could be stored and shared in the boundary zone, rather than being

relegated to individual environments. As such the chat entries would be localized differently in different environments, while each CoP could chat in their own language. It could be used to support kids learning languages and making international friends.

5.4 Emergent Patterns

In MikiWiki, usage patterns can be observed and turned into templates, i.e. recurrent markup, HTML or JavaScript code can be abstracted to build a template for future reuse. Empowering users to abstract their own usage patterns and build their own templates enhances the efficiency of the system, and knowledge can be captured, shared, reused and used to build more complex templates or create new nuggets. The process of observing patterns, creating templates, reusing and extending them is shown in Figure 42. To this extent, templates can be exploited to capture emergent behaviors.

For example, HTML templates have existed for quite a long time, and they are a nice compromise between static pages and putting HTML inside of a program, which makes it inaccessible to a designer. To change a static HTML page dynamically, using templates is a good way to go (Lerner 2004). A template is a generic outline for the content of a page. Everything is considered to be static, except for variables, whose values are filled in at runtime.

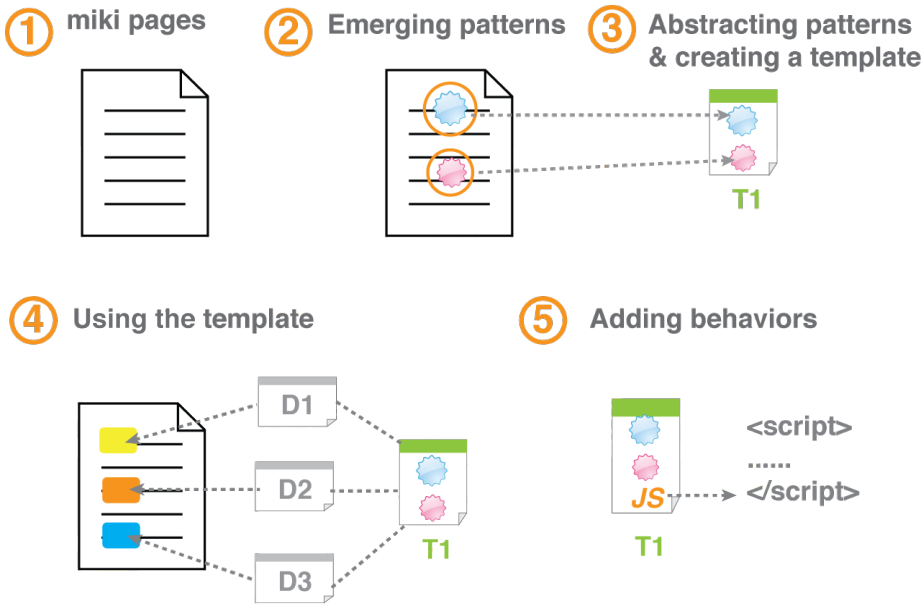


Figure 42 A process of observing patterns and creating templates

In some cases a specific combination of nuggets can be effective for design activities. For example, a combination of a *video*, *ranking* and *comment* nuggets can be viewed as a new simplified *YouTube* nugget. If this nugget assemblage task has to be repeated frequently, it is more efficient to exploit the MikiWiki templating mechanism. For example, consider the template in Figure 43, where a new *videopage* nugget is created and it is assigned the *template* format.

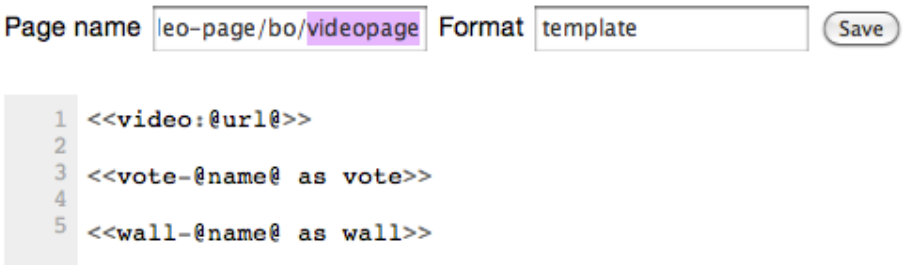


Figure 43 Using a template to create an aggregated nugget

Three nuggets, namely, the *video*, the *vote* and the *wall* nuggets are included within the *videopage* (Figure 43, Figure 44). Figure 45 shows variables (*video* URL, the *vote* nugget data page name and the *wall* nugget data page name) which are placed in between "@ @" signs, whose values can be filled in.

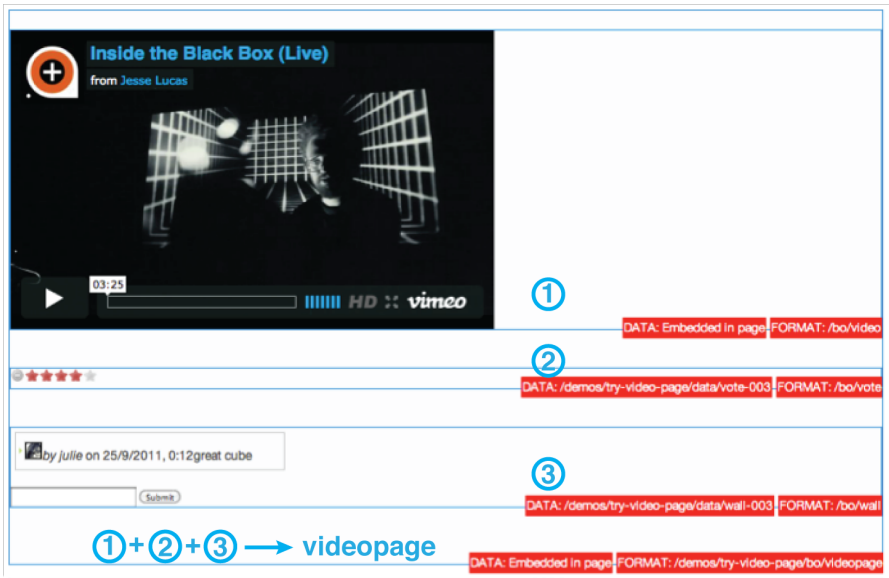


Figure 44 Videopage nugget composed of three nuggets

Typically, meta-designers provide nuggets, (e.g. video, vote and wall), and might modify them according to the usage context for designers. Designers are able to put together, remix and combine nuggets to create something new.

Figure 45 Specifying a video URL and a data page name to the *videopage* nugget

The final results - specialized MikiWiki pages - can then be made available to users.

Creating templates plays an important role in the Aristotele project, which will be discussed in more detail in Chapter 8.

5.5 Levels of Participation

The three levels of design follow SSW methodology, meta-design, design and use levels (Costabile et al. 2006a) , respectively. As an example Figure 46 shows the three levels of participation in designing and using the *videopage* nugget,

At the meta-design level: meta-designers provide nuggets and examples of how to use them. In this case, nuggets are the *video*, the *wall* and the *vote* nuggets.

At the design level: designers main activity is involved in creating an aggregated nugget via combining three nuggets provided by meta-designers. As its format is "*template*", designers can use MikiWiki special markup language rather than involved in scripting. With the new nugget, videopage, designers can parse the video URL and specify data page name accordingly.

At the use level: users can simply rank and comment on videos created by designers.

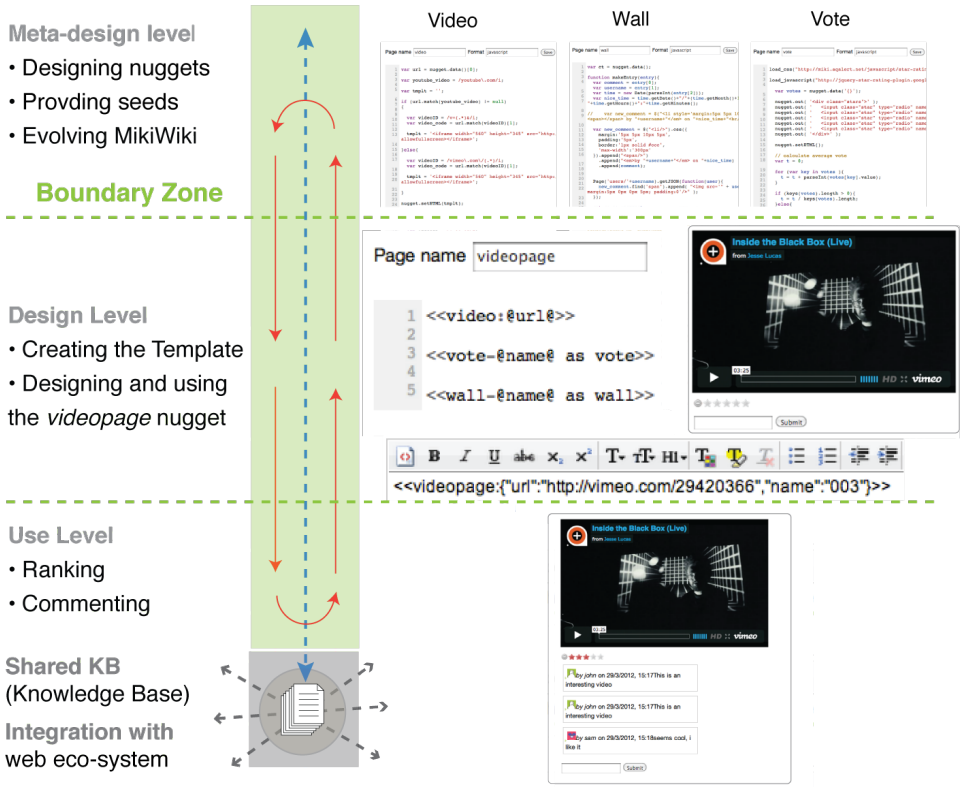


Figure 46 Collaboratively designing a videopage nugget

Notably, all design levels could be flat and accessible by everyone within the same environment, as is demonstrated in the energy feedback system design (see Chapter 8). The levels of participation are not predefined, but emergent in terms of users' skills and roles, which are highly dynamic. For instance, creating the *videonugget* nugget activity could be done by meta-designers' based on the requirements expressed by the designers and end users or observation of their appropriations and recurring usage patterns over time. It could be users' activity by simply putting three different nuggets together in a MikiWiki page.

Before designing MikiWiki, I perceived the three levels of participation (meta-design, design and use) as being different 'places' within the HMS and with different users. During the implementation I realized that they were not as much places as modes of work. Users could decide to create different environments to carry out meta-design or design activities, but this is not necessary. As demonstrated in the design studies, all the design activities can happen in the same environment.

5.5.1 Levels of Tailorability

To better support different levels of participation and tailoring activities, MikiWiki provides two different content editors, a rich text editor (a WYSIWYG editor that allows novice users to easily create text context) and a JavaScript editor (aimed at expert users who are able to program). The rich text editor allows users to switch to the source code mode, so that they can write HTML code.

Apart from the design activities, Table 6 shows different levels of possible tailoring activities (Mørch 1997). In the context of component-based EUD approach, *customization* is to modify the parameters of existing components; *integration* is to create or modify assemblies of components; and *extension* is to create new components by coding (Mørch et al. 2004a).

<table>
<tr><th>Complexity / Adaptation</th><th>Customization</th><th>Integration</th><th>Extension</th></tr>
<tr><td>Advanced users, web developers</td><td rowspan="3">Modify parameters of existing nuggets via data page or format page;
Embed and combine different nuggets;
Specify parameters for nuggets, e.g. video URL, linking external CSS, files</td><td>Modify nuggets format page;
Combine different nuggets' format pages</td><td>Program (on the server side);
Script;
Create new nuggets</td></tr>
<tr><td>Designers</td><td>Program by nuggets examples;</td><td>Script;
Create new templates
Write HTML, CSS code</td></tr>
<tr><td>Users</td><td>Program by nuggets examples;</td><td>Create new templates</td></tr>
</table>

Table 6 Levels of tailoring adapt from (Spahn 2008)

The meta-design, design and use modes coexist within the same environment and users can move between them, achieving higher levels of system tailorability. Therefore, MikiWiki is both a development environment and a collaborative design environment. Tailoring activities are intertwined with design and use activities rather than merely for the sake of tailoring. Notably, tailoring is an activity that requires certain familiarity with the system. For instance in the Creativity Barometer study (Chapter 9), users and designers were not involved in tailoring the system, partially because design sessions were short and partially because meta-designer did all the tailoring work in between the design sessions.

5.5.2 Levels of Creativity

Meta-design features underpinned by the HMS model play an important role in supporting different levels of creativity: creative design from meta-designers and creativity in use from designers as well as users triggers further creative meta-design – that is, creativity from different layers, situated creativity, individual creativity as well as social creativity.

Meta-design level: constructing design environments is an activity occurring at the meta-design level, in that the meta-designer sets up the initial design environment for the design session and constantly evolves it opportunistically to cope with emergent socio-technical issues without needing to change server-side code. The opportunistic developers are those who excel at taking existing code and components and combining them in some way to accomplish a task (Brandt et al. 2008). They do not invent brand new solutions, classes or frameworks; rather, they tinker with existing components in creative ways to solve their task. Meta-design creativity can be viewed as bricolage, situated creativity in action.

Design level: design environments support creativity at the design and the use level, in that participants continuously adapt nuggets to form a design space in order to perform their design tasks at that moment.

Use level: participants use the tailored design space at different phases to externalize their thoughts immediately.

As a whole, social creativity is supported through collaborative design; intertwined creation processes; and reciprocal interplaying both between users, and users and software artifacts.

5.6 Open Infrastructure: Harnessing the Existing Web Ecosystem

Open infrastructure is addressed in many aspects. The focus on EUD allows client-side programming and programming by example within MikiWiki.

The open infrastructure is supported by the extensibility of MikiWiki, which can harness the existing "*web ecosystem*". By "*web ecosystem*" I mean the entirety of web applications, services and APIs that are made available to web users and application developers. Many of these applications are considered so widespread that many other applications and work processes rely on them, for instance Dropbox, Google search, Google applications, YouTube, twitter and Facebook. The Web is a rich resource of information that can help the design process. Development tools in general and end-user development tools specifically should make use of these resources by providing seamless connection mechanisms and helping to find relevant resources based on the current design state (Repenning and Ioannidou 2006b).

The openness and adaptability of MikiWiki is the most central characteristic of meta-design, which can be achieved not only by flexibly combining small components but also the smooth integration with the Web ecosystem. For example, in the Aristotele project (Chapter 8), designers embedded various resources (e.g. Google Docs) to utilize synchronous editing

possibility, Dropbox, YouTube, websites, external slides, PDF files, and so on. This was to prepare the knowledge base, which was independent from MikiWiki yet could be closely related and easily connected to a network. Additionally, tailorability is needed to allow integration of external resources in the system. CSCW research is currently shifting towards the integration of different existing tools into comprehensive CSCW systems (Sandkuhl et al. 1998) rather than reinventing the wheel.

Embedding Multimedia

Nuggets are provided to easily embed videos from YouTube and Vimeo. Figure 47 shows an example of using the *videonote* nugget to embed a set of videos by passing the video URL. Video notes can be resized, rearranged, played simultaneously or in a particular sequence to convey certain concepts. The final result can be more powerful than static images. There are many ways to experiment, play with videos and be inspired by them.

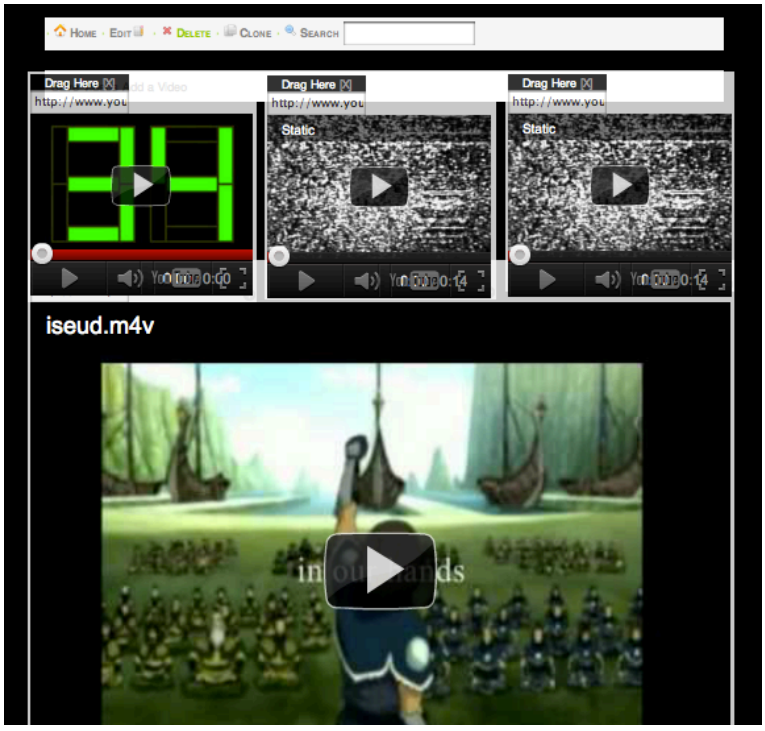


Figure 47 Integrating YouTube videos with the *videonote* nugget

External Files Linking

Google Docs, external websites, PDF files, slides and different file formats can be embedded via utilizing different nuggets. As such, users can easily collect resources, and store and share them with other members. This keeps MikiWiki lightweight and flexible, as resources are independent from MikiWiki. Combining with the *tag* nugget, users can better organize and retrieve resources. Additionally, MikiWiki takes advantage of the strength of other platforms (for instance, embedding Google Docs to extend the synchronous editing aspect in MikiWiki). Figure 48 shows utilizing the Dropbox public folder generated URL to link an image with the *imagenote* nugget.

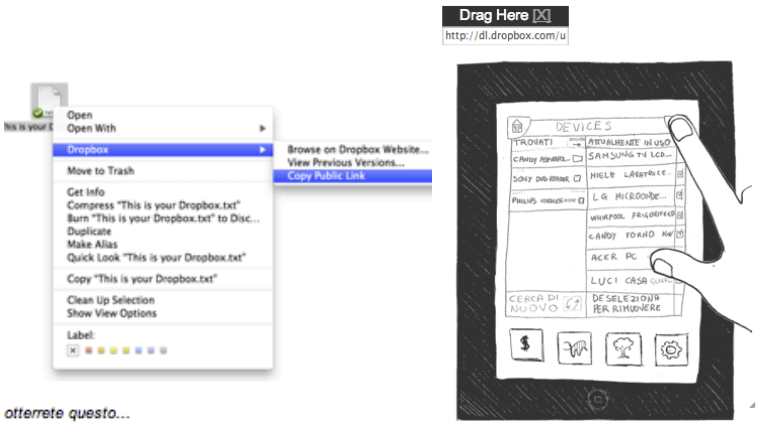


Figure 48 Linking an image from Dropbox public folder

We have embedded an ongoing web portal in MikiWiki, thus end users used the colored *note* and *doodle* nuggets to comment on the system. Additionally, users associate different colors with different meanings or priorities.

Google and Flickr Image Search

MikiWiki provides nuggets to search images from Google and Flickr with keywords, then choose and use them within the system. Figure 49 illustrates a moodboard created via the *sync-imagenote* nugget.

Notably, users can specify an image URL rather than keywords to retrieve a specific image. On the other hand, the randomness of search results generated by Google and Flickr brings positive inputs, stimulates free association and encourages situated creativity. Many users commented on this in the creativity barometer mobile version design (see Chapter 10).

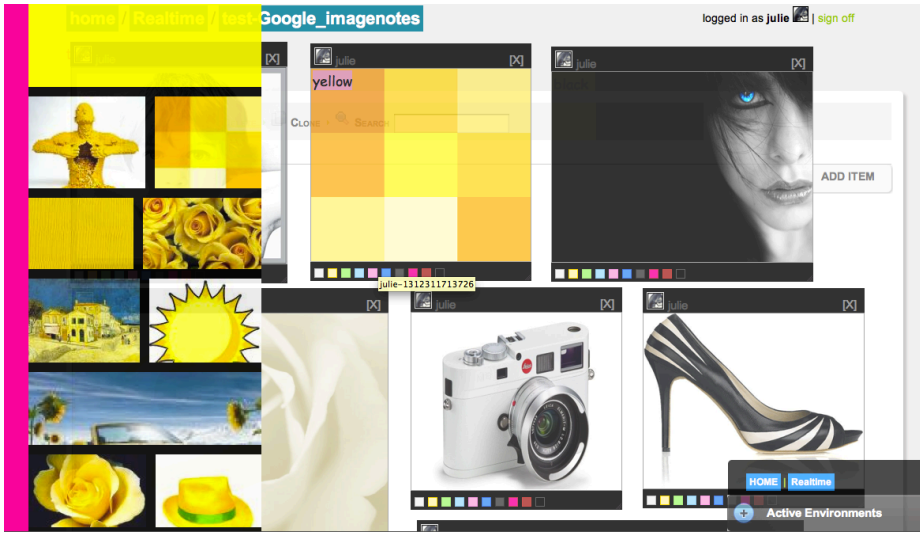


Figure 49 Images from Google and Flickr

Loading Cascading Style Sheets (CSS) within and outside MikiWiki

Loading external CSSs can be very useful to prototype web applications. In particular, experimenting with different emotions or moods, one can apply rendering instructions for

individual MikiWiki pages, overriding locally or for environments as well as all the sub-environments as shown in this case (Figure 50).

Users can create their own CSS within MikiWiki, which empowers them to decide how to render the content - for instance, viewing compact documents with small fonts, or specifying larger fonts for increased legibility. CSS can be easily accessed and reused. This possibility allows designers to refine a preferred “look and feel” as well as alternates for target groups or media according to practice. As such, it makes it easier to manage style in the project or environment basis.

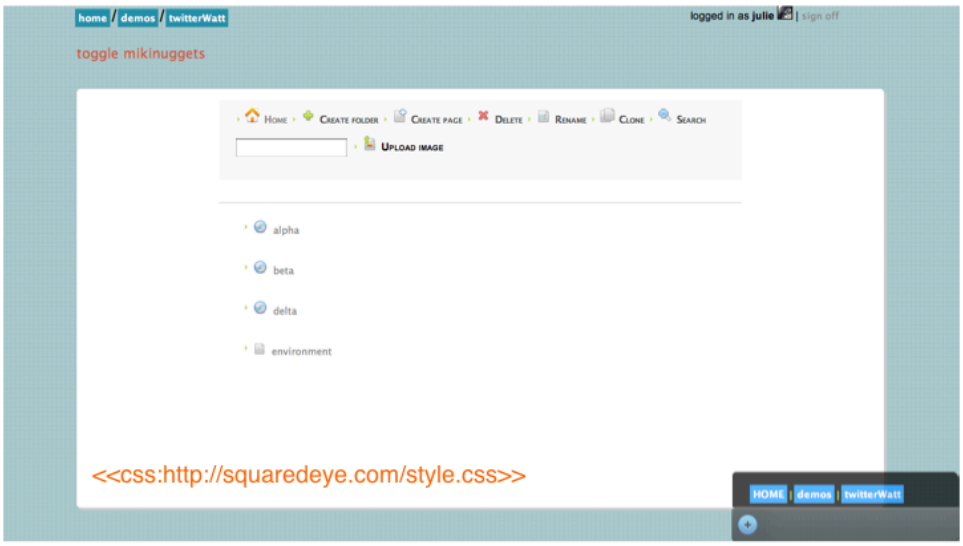


Figure 50 Styling environment with an external CSS

Dynamically Loading External JavaScript

Loading external JavaScript files at runtime reduces application startup time and improves the application performance. More importantly, it further opens up the system allowing inspection and tinkering, and easing the difficulties of creating new nuggets, as users can load other JavaScript libraries dynamically.

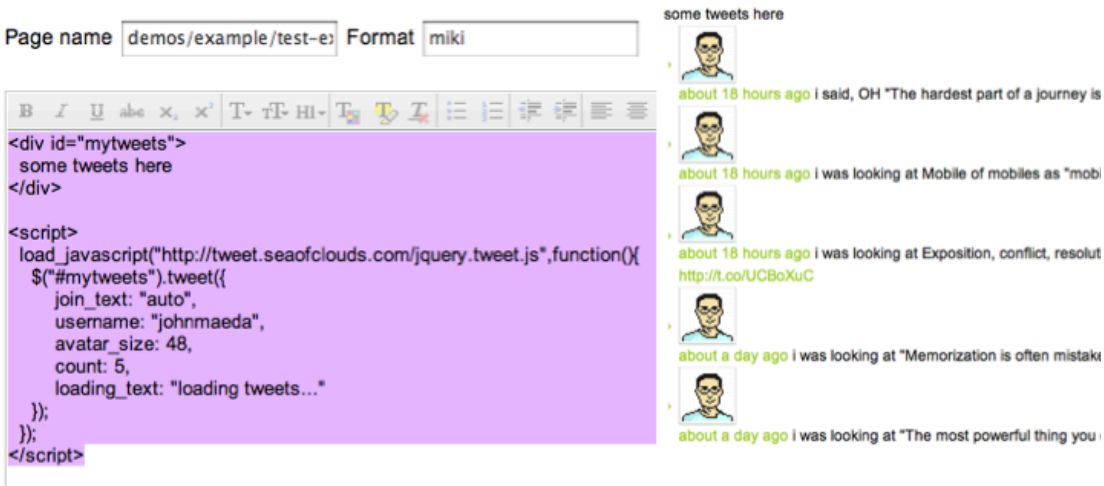


Figure 51 Loading an external JavaScript library to display tweets

Figure 51 demonstrates the use of an external JavaScript library to display John Maeda's tweets in MikiWiki in real time.

Furthermore, it is increasingly common to create compelling web applications that integrate images, text, multimedia and other data from various web sites in an innovative manner. MikiWiki provides a playground whose open infrastructure empowers users themselves to create their own content or integrate specific resources in the environment when it is necessary and in use time.

5.7 SER Model

The underdesign concept is a key concept embodied in MikiWiki. MikiWiki provides an infrastructure with generic features to support collaboration, e.g. communication, coordination, history inspection. Thus it can be easily applied and adapted for various application domains. MikiWiki has been applied and extended for various design cases - respectively, prototyping web applications synchronously and asynchronously, collaborative writing, and role-playing games.

Collaborative and communication features in MikiWiki are not in-built in the system, but are made available as underdesigned "nuggets" on top of the system. Hence, they are seeds encouraging appropriation and modification. Furthermore, the way nuggets are designed and used aligns with the concept of underdesign. Nuggets can act as exemplary seeds, representing a small portion of infinite possibilities that can be explored by users by the means of appropriation during the collaborative design process.

Nuggets are designed with just enough features to be useful, and at the same time leaving code short and simple to be quickly understood and modifiable so that initial features can be easily extended for different contexts. This is in contrast to traditional software engineering approaches that strive for completeness, formal purity and flexibility - by trying to anticipate change - at the expense of short, readable and easily modifiable code.

Figure 52 presents a rather simple and underdesigned RPG environment, in which players used the *sync-imagenote* nugget, typed keywords, searched images associated to their characters in the embedded Google Image Search service, and used them directly to build narratives. As players do not have to be preloaded characters, or predetermined story setting, the game can be more open, supporting improvisation. Players use notes as speech bubbles. By moving the yellow note to indicate who has the right to move the characters or modify the world at any one point, players take turns to build narratives.

Figure 52 Using the *sync-imagenote* nugget to play an RPG

What is interesting about using wikis, and especially using MikiWiki, is that players can start with social and soft rules (e.g. rules about social negotiation; what is appropriate to say or not to; whether the content is relevant and so on), and these facets could be established merely by agreement among players. In this case, all the rules are based on social conventions.

Nevertheless, over time players and programmers are able to observe conventions and formalize them into code, consequently incrementally enhancing the RPG environment to be more standardized. For instance, in Figure 52 the note to indicate which player is playing can be formalized as a timer, and the speech bubble image as an in-place editing text area with a speech bubble shape.

MikiWiki does not intend to provide a completed product or a set of well-defined functions. In contrast, the ongoing design process only makes sense and unfolds itself over time based on situated contexts as well as the development of mutual understanding among stakeholders. In the collaborative design process, stakeholders follow their inclinations and design instinctively to pursue their evolving goals to couple with wicked problems (Rittel and Webber 1984).

With respect to well-designed and closed systems or possible application domains, MikiWiki has its own advantages:

Productivity Applications. Applications such as Google Docs, GoogleTalk and Gmail already offer a high level of sharing and collaboration within an organization, between friends and with the larger public. These applications however lack support for customization, which is often achieved only by using specialized browser plug-ins. The use of some of the techniques employed by MikiWiki would allow building of a custom features and leverage the

existing communication mechanisms to share them with other users. Similarly, web widgets have been discussed in Section 3.4.4.

Online Spreadsheets. Google Spreadsheet and other online applications allow users to work on the same spreadsheets. Google Spreadsheet went so far as allow the scripting of new formulas in JavaScript. It is possible that these formulas and macros could be shared and maintained collaboratively across an organization using mechanisms similar to the ones employed by MikiWiki.

Collaborative Creative Writing. MikiWiki can be used as a brainstorming tool and platform to build dedicated tools for defining basic story principles and structure, and leave participants generating ideas and developing plots.

Online Games. Games have been open to user customizations (known as 'mod') for a long time. SecondLife is a great example of how to provide extensive customization and scripting from within the environment itself (LSL 2003). As in MikiWiki, online gaming platforms could benefit from an environment for social customization and sharing of programmable artifacts and characters.

Collaborative Software Design (Henderson and Kyng 1991). Software design and design process documentation can be easily integrated in MikiWiki. Beginner, advanced and expert users are encouraged to participate in software design, discuss it, share code and test application prototypes. Github (social project sharing) (GitHub 2008), Cloud9 (team based online development environment) (Cloud9 2011) and jsFiddle (online JavaScript prototyping) (jsFiddle 2010) are all projects working in this direction.

Social Networking Platforms. Currently, the most popular social networking platforms, such as Facebook and LinkedIn, support the development of third party integrated applications, yet are lacking an integrated development environment and custom services live on third party servers called via callback APIs. MikiWiki-inspired techniques would work on client-side extension mechanisms and integrated shareable code environments, relying on the existing social features.

5.8 Conclusions

In short, MikiWiki is a prototype developed according to the HMS model. It explores objectives, techniques and processes and underpins meta-design characteristics. Thus, users can collaboratively find incremental solutions that bring creativity and technology together in a way that is highly adaptive to local conditions.

The mapping might not be exactly one-to-one; for instance the communication channel cannot be reduced to a simple software artifact. Rather, it consists of creation processes, social interaction and communication, which are abstract, dynamic and emergent, and are enabled by the nugget infrastructure.

The next chapter will introduce how MikiWiki is implemented to reflect the HMS model characteristics from a technical perspective, providing a reference architecture for future implementations.

Chapter 6

This chapter describes the HMS architecture, a reference architecture for the HMS Model described in Chapter 5, and it details the implementation of MikiWiki, the wiki-based architecture prototype described in the previous chapter. Utilizing an end-user development approach, MikiWiki provides meta-design artifacts as seeds for evolutionary growth, supports different levels of participation and enable continuous design-in-use. This chapter is divided into three parts: the first part outlines the characteristics of the HMS conceptual architecture and the rationale behind it; the second part explains the reasons I chose the wiki model as a starting point for an implementation; the third part explains the implemented MikiWiki architecture in detail, how the HMS architecture attributes are embodied in the MikiWiki architecture and how different features are accessed from a technical perspective.

6. Architecture and Implementation

6.1 Architectural Principles

The HMS features described in Chapter 4 should be translated into concrete system specifications, which can be implemented and executed. The architecture detailed in this chapter reflects on the HMS model characteristics to argument the flexibility, openness and tailorability of the infrastructure for collaborative design in use and the means for tinkering as well as evolving communication between users and between users and developers.

The main principle of the HMS architecture is to empower end-users to carry on design in use, putting them in charge of their design problems by enhancing their tailoring capabilities and creating situational applications (Riehle and In 2006) for better collaboration and problem-solving.

The server-side architecture should be lightweight and simple in order to shift design power to the client-side, for instance to manage the interaction or communication between users, to create personally meaningful design environment.

The HMS architecture is based on a combination of:

1) Client-side tailoring and end-user programming

2) A component-based approach for connecting design time and use time, based on the idea of building a configurable 'infrastructure' working with general building blocks, components and component assemblies. Some of these components can themselves become social objects and may be seen as 'boundary objects' between the meta-designers, designer, and users (Ehn 2008).

6.1.1 Client-side Web Development

Client-side Web development is an emerging trend, since it provides the possibility to empower end users to create applications by merely using a Web browser (Ingalls et al. 2008).

Agile development approach engages users in participation, but only in the software development stage rather than use time. Allowing users to access development facilities via the same interface used for the application environment facilitates a tighter integration between use-time and development-time.

The wide use of AJAX techniques contributes a fast interaction in the Web-browser, facilitating easy and fast collaboration.

6.1.2 Basic Services and Flexible Mechanisms Separation

The rationale behind the HMS architecture design is to empower end-users continuing design in use. This has to be well balanced between the notion of change and the concept of stability.

The distinction between tailoring and use thus rely on the understood and intended variability of artifacts and their patterns of use. Certain aspects of these artifacts we, as users, regard as stable; others we regard as more or less constantly changing. This relative stability of certain aspects is exactly what allows us to consider tailoring of an artifact (Henderson and Kyng 1991).

To design systems that can be tailored through construction, it is important that the elements and mechanisms, which support construction, form a coherent assemblage and support a coherent tailoring activity. In short, construction must be designed.

Defining the initial architecture is a key step in design, one that may have long-term effect in terms of the future maintainability and extensibility of the system. The basic architectural concept in MikiWiki was the wiki page, which evolved into the nugget concept, as described in Chapter 6, while all the definition of all the other architectural services co-evolved organically in time with the use of MikiWiki, adding only as much complexity as it was needed at the time.

In the HMS architecture, *nuggets* are the rough analogue of boundary objects in the HMS conceptual model. Nuggets are the basic unit of sharing, providing the basic properties that allow the emergence and composition of boundary objects. The concept of nugget in use has been widely described in Chapter 5.

The HMS architecture provides end users with a simple API. Instead of seeking to make the API complete or conceptually pure, the HMS architecture developed only the basic and generic API that end-users needed at the time and built it to be immediately usable.

Some functionality cannot be removed from the server side: specifically, all the basic facilities that handle the sharing of information as the server acts as a repository and single aggregation point for the boundary objects. An abstract system is created for sharing nuggets, without encumbering the server side with the specifics of what is being shared, therefore keeping it very simple.

The semantics of what is being shared are expressed on the client side, where eventual nuggets are executed. A nugget can be embedded within another nugget in order to create sharable remixable components. Nuggets in the HMS architecture are explicitly designed to reflect the HMS boundary objects, articulating social, evolutionary and bricolage aspects. These can be achieved by supporting different levels of tailoring, integration of multi-perspectives, bridging runtime and use time and empowering users to access, inspect and influence the system.

6.1.3 Levels of Tailorability

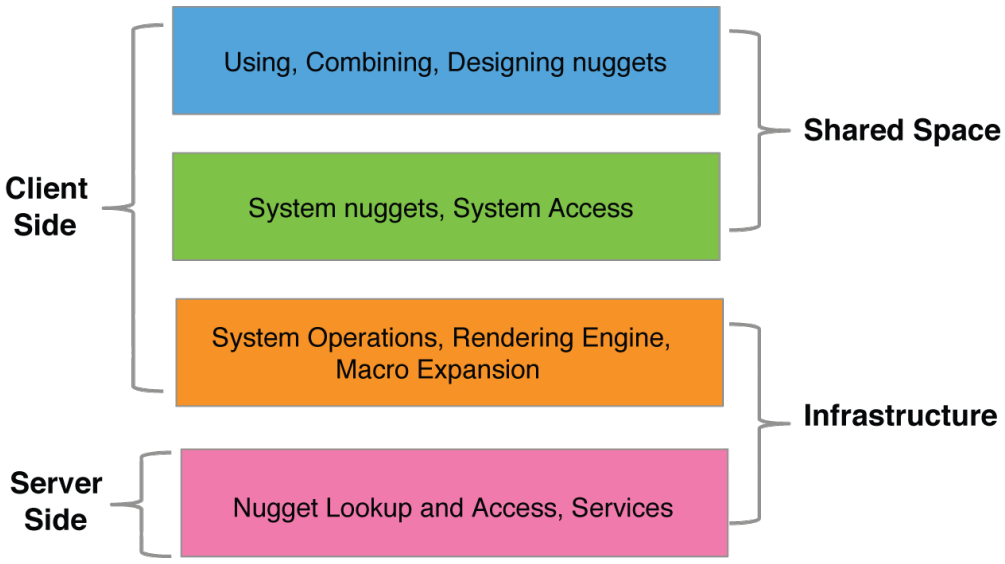


Figure 53 Architecture levels

As a concept demonstrator, the HMS architecture aims to explore some key characteristics of meta-design, i.e. design infrastructure, tailorability and EUD. Therefore, the system architecture mainly addresses the tailoring of the client side, experimenting with empowering

end users to evolve the behavior of the system according to the characteristics of their collaboration.

Issues of security and scalability are not explicitly addressed since they are out of our research focus and I mean to keep the architecture tidy, lightweight and fully focused on collaborative tailoring and open-ended evolution.

Figure 53 shows the four main levels of the HMS architecture:

1) **The Server-side Infrastructure**: this part of the HMS architecture defines the basic concepts of nugget and environment as well as al the basic services that require centralized coordination. Even the meta-designer participants cannot modify this layer of the architecture, as it requires access to the server.

2) **The Client-side Infrastructure**: this layer is executed on the client side and it provides the basic user interface, navigation structure, editing interfaces, the nugget rendering engine and the APIs needed to connect to the serve-side services. Advanced users taking over the role of meta-designers can access this layer and override some of the basic behavior of the system, although this layer is not explicitly designed to be easily accessible and sharable.

3) **The System Shared Space**: this layer resides on the client side and provides users with some universal nuggets that can be generally useful. This layer includes the *link* and *include* nuggets, the *allow* nugget – handling basic authentication – and a number of utility nuggets that are provided as seeds to be cloned and overridden. Users cannot modify system nuggets, as this could permanently break the system, but they can override them with their own implementations. As the architecture evolves more and more interface and navigation features should be moved from the Client-side Infrastructure to the System Shared Space, to make it explicitly visible and available for tinkering.

4) **The Inhabited Shared Space**: this level is technically similar to the System Shared Space, but allows users either a group or as individuals to use and compose nuggets, design their own and evolve existing ones. Within the Inhabited Shared Space users can create their own environments and shape their artifacts and communication processes. Within this layer there can be rules and access limitations, but they are not imposed by the system, but they are socially handled by users and by the norms that they created and imposed within their environments.

6.2 Architectural Overview

The HMS architecture is built around the basic concept of nugget. A nugget is the basic unit of sharing and is augmented with a number of services designed to offer users the opportunity to develop processes and artifacts that can be seen as boundary objects. Nuggets can act as boundary objects, integrating the different perspectives and capturing knowledge of CoPs.

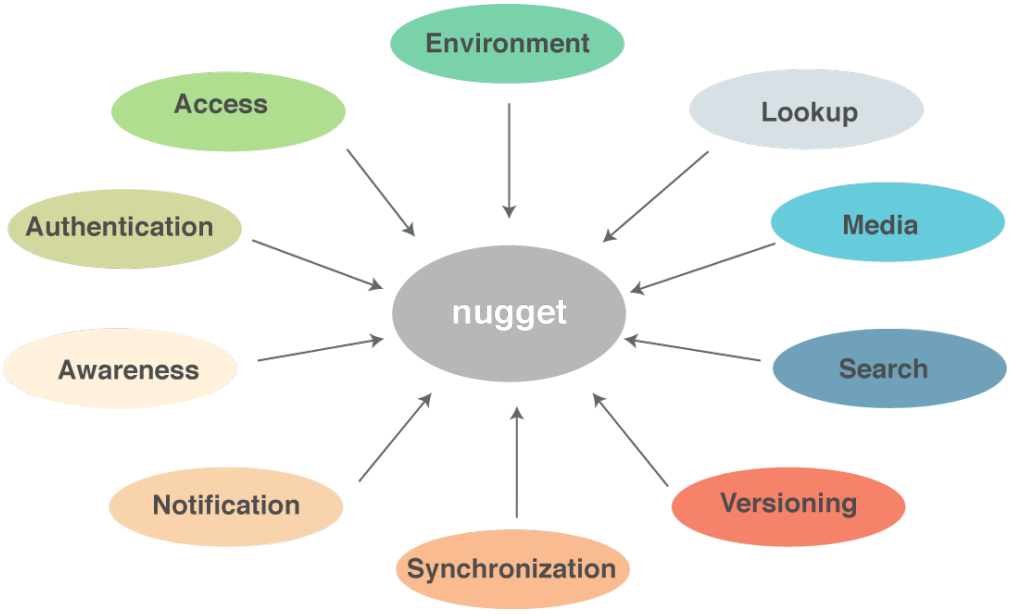


Figure 54 Possible operations on nuggets

Figure 54 depicts the actions that can be performed on the nuggets by the services supported by the HMS architecture.

Developing New Nuggets

Nuggets are used to express basic user interface elements, shared artifacts and communication tools. For example, an icon representing a user that is modifying a page could be a nugget, the navigation menu could be another nugget and a shared text yet another nugget. Nuggets can be used to encapsulate simple changes, talk about them and share them. Nuggets often start as simple text, small changes to the system, or simple embedded scripts and get encapsulated only at a later stage when they acquire a social meaning and need to be shared.

1) Nugget Macro Language

Users should be able to instantiate nuggets by the means of some kind of simple component-based macro language. This language could be either a textual macro language or a visual drag-and-drop interface. In the implementation of MikiWiki I opted for a simple textual macro mechanism based on the use of angular and square brackets to embed nuggets within the text of a wiki page.

2) Nugget Templating Language

Users should be able to abstract simple patterns based on the aggregation of simple nuggets or on the variation of unrelated dimensions of configuration. This can be done by specifying the nuggets belonging to the aggregate and explicitly marking the variation points. In MikiWiki this is achieved via the simple templating language interpreted by the *template* format nugget, as described in Chapter 5.

3) Nugget Definition Language

Finally, it is necessary to have a client-side language that can be used to define new nuggets and modify existing ones. Ideally this would be the same language used to define the client-side infrastructure, allowing de-facto deep and full access to the client-side architecture. At the time of the thesis I used JavaScript as the nugget definition language for MikiWiki, and I also experimented successfully with CoffeeScript (Ashkenas 2011a), a higher level language that translates to JavaScript. Since there are a number of high level languages, including visual flow languages, that have produce JavaScript as target code, it is possible to reason on the use of more specialized client-side languages to describe nuggets for specific domains (Ashkenas 2011b).

4) Nugget Metaprotocol API

The Nugget definition language should have access to a basic API that provides a metaprotocol service on top of nuggets. I should be able to programmatically create and inspect a nugget, asking questions about its data, format and other metadata and their run-time placement and state within the system. This API is necessary to provide run-time nugget interoperability and manipulation.

Nugget Access

A basic protocol to access nuggets is needed. According to Henderson et al., software systems should make authoring as easy as possible. Support means of creation is the support of information access: finding, negotiation, making available (Henderson and Kyng 1991) .

Nuggets become a core part of the definition of the HMS architecture, since they can be added, deleted, modified and manipulated, shared, by members in the design community. Create, read, update and delete (CRUD operations – as defined by (Martin 1983)) are four basic functions one can operate on nuggets, allowing users to:

- Create new nuggets
- Read the data of existing nuggets
- Read the metadata of existing nuggets: format, update time and identity of the user that made the update
- Update and tailor existing nuggets
- Update the metadata of a nugget
- Delete existing nuggets

Media

Modern web systems rely heavily on images, video, PDF documents and other media. We must have a way to add them to the system and manipulate them as nuggets. We should enable the upload of different media contents, for instance, different text based documents,

images, audio files with different formats and video files with various formats to create a rich knowledge base, sharing a wide range of information.

Environment

The Environment service provides users with a way to structure space and create new possibilities for enabling parallel social and informational organization. It allows users to incrementally define their own space, content and behavior to match their work practices.

Environments should be treated as nuggets, being themselves a piece of shared knowledge, whose purpose is to organize other pieces of knowledge. In the MikiWiki implementation of the HMS architectures, environments are configured via an *environment* page, making them effectively into nuggets and negotiable boundary objects.

Versioning

The HMS architecture should support users to explore, try things out, and yet backtrack when unsuccessful. This means that the available tools must be trustworthy so that users are comfortable trying things (Resnick et al. 2005). For instance, the possibility of creating an undo capability is a strong requirement for many tasks. Implementing Undo can be quite difficult however (Myers and Kosbie 1996), so many research systems leave it off. Versioning allows users to go back to previous version, thus they are able to make changes without fear of losing content. The rich histories that are required to support undo can be useful for users to reflect on the collaborative creation process and reason on the interactions between different design communities.

Authentication

Collaboration requires access to information. Access-control policies establish the ground rules under which various users may access shared information objects. This allows users to define groups and impose nugget-level read and write access restrictions on their own resources and nuggets based on user lists and affiliations.

Asynchronous Notification

Coordination is fundamental issue in all kinds of collaborative activities. Therefore, if synchronous awareness supports coordination, the provision of awareness information to support coordination should be just as important in asynchronous collaborative systems (Bellotti and Rogers 1997; Dourish 1997).

Notifications provide a means to support asynchronous awareness. A notification service allows users to react to changes in the system, even when the system is not the primary focus of their attention. Notification can range from sending an email, an SMS or a twitter notification. An HMS architecture should tie the system to users' lives by notifying them when objects of interest are accessed or changed, and provide them with basic information to enable them to act on that.

Realtime Awareness

The essential role of awareness is to make one's activity visible to others. In co-present collaborative settings, activities are coordinated between individuals through their awareness of each other's action (Dourish 1997). Knowing what other users are doing offers hints for socialization, for asking questions, to check what they are doing. The system should be able to capture the focus of users attention and the changes they make to the system and have the potential to make it available to other users.

As in the case of notification this can inspire socialization and collaboration, but it also offers a layer of implicit background information, allowing users to keep track of one another without resorting to explicit communication.

Referencing and Lookup Mechanism

The HMS architecture should support either a URI (Universal Resource Identifier) (Berners-Lee 2005) or another naming system that allows the identification of specific resources. This is needed in order to support the concept of linking and therefore providing unique names for nuggets allows them to interact and to be composed into more complex structures.

It is necessary to customize specific resources within the system, so that users can personalize their local environments by overriding a nugget with new behavior. In order to implement the overriding system, the HMS naming architecture should support a partial naming reference system, based on pattern matching between the names used to reference nuggets and the URIs used to name the nuggets.

A nugget lookup mechanism should resolve the identity of an individual nugget by retrieving the nugget whose name correctly matches the partial name provided.

This system has been implemented in MikiWiki by trying to match the partial name reference to a set of nuggets, starting from the environment nearest to the reference and moving outwards to the enclosing environments.

The lookup system was first evolved in MikiWiki out of the practical difficulty of having to specify a full path for every nugget every time we referred to it. Additionally, renaming one of the enclosing environments would break the absolute path name of the enclosed nugget.

This problem was solved by allowing initially partial relative names, and later a full lookup mechanism based on hierarchical scoping: whenever a resource is referenced, the HMS architecture looks it up starting from the immediate environment, then proceeding to the containing environments. The implication is two-fold: one is that locally defined resources override globally defined resources; the second one is that information should be close to where it is used (Randall et al. 2007).

Synchronous Communication

While not strictly necessary, and not always welcome (users may like to work undisturbed), synchronous communication can remove many ambiguities and provide for a much richer interaction.

Successful negotiation on issues related to organization, planning, and control requires provision of an effective system for communication among the individuals involved. For this reason, human-to-human communication is one of the key features needed for CSCW (Chapanis 1975).

In the absence of either multimedia conferencing support or audio communication channels, successful collaboration can still be conducted through the use of text-based interaction systems, known variously as chat applications or chat rooms. Text-based chat applications can provide private channels for a subset of collaborators to hold side conversations outside the purview of the main proceedings. As chat applications become more sophisticated, they can provide convenient means to distribute documents, data, and images related to a collaborative session.

Versioning and synchronization policies define the ground rules under which different versions of the same object may be combined into a single, consistent copy (Mills 2003).

Search

Finding content and making individual perspectives on this content visible is still a problem in collaboration support systems (Mathes 2004), especially if non-textual content is concerned (Prilla 2007). Being able to retrieve information, explore existing knowledge base. It could rely on an explicit nugget tagging mechanism.

The Tag service allows users to assign keywords to pages without relying on a controlled vocabulary. One of the major advantages of tagging is its universal applicability to different content type, whether a page contains an image, a video, an audio clip or merely text-based content. Tagging systems have the potential to improve search, provide recommendations, and optimize personal organization while introducing new modalities of social communication and opportunities for data mining (Marlow et al. 2006). Note that tagging supports collaborative activities, which can be used to contextualize content individually, but also contributes to a group's perspective and awareness.

Tagging can be used as a way to organize and describe content in a bottom-up fashion. It enriches the means to structure and contextualize content and it keeps the cognitive effort needed for contextualization low (Grudin 2006) and therefore reflects the playful and user-oriented approach of Web 2.0 mechanisms.

6.3 Architecture and Process Overview

This section puts together the nuggets with all the services I have discussed in the previous section to provide an overview of an HMS client-server architecture. The abstract HMS architecture is generic and it does not necessary imply a wiki implementation. The same architectural reference could be used to support multi-user virtual worlds or document sharing within a corporate intranet, as detailed in Chapter 4.

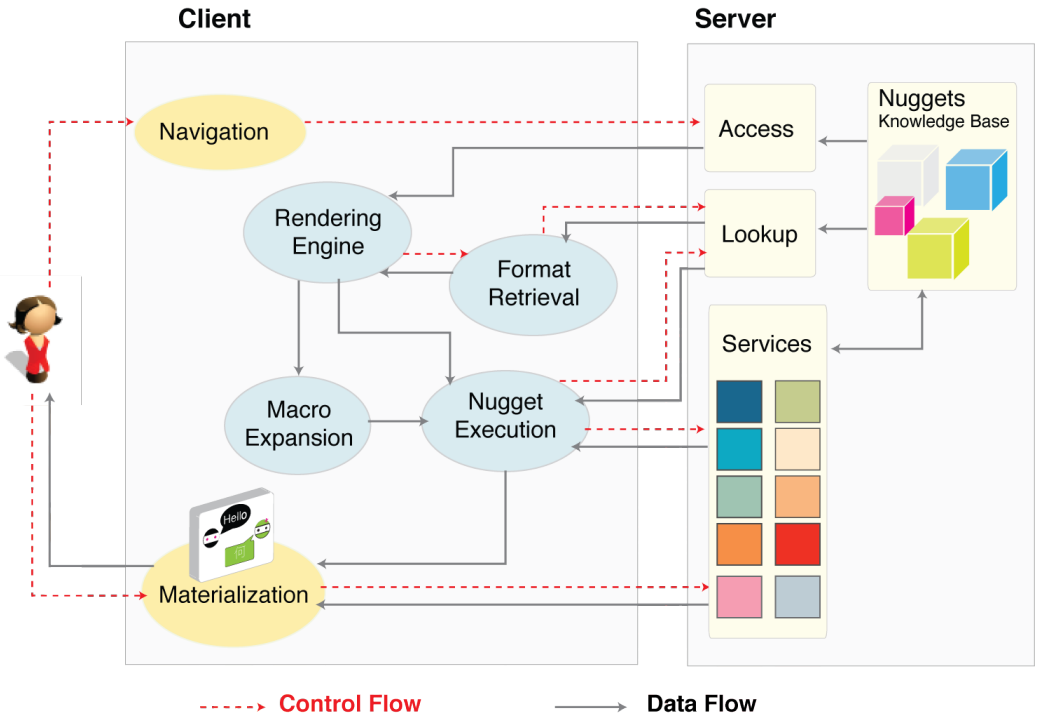


Figure 55 Abstract HMS architecture

Figure 55 illustrates the abstract HMS architecture, which integrates all the services on the server side, with the rendering and navigation services handled on the client side.

A client should provide users with a visualization of the shared elements. The client should provide navigation capabilities and the possibility of addressing specific nuggets and environments (which are also nuggets). The client should have the means to retrieve nuggets from the server via either a full or partial reference.

The client has a renderer that displays the nuggets based on a format that describes how to render the retrieved data. Calls are made to the server until all information necessary to perform the rendering has been retrieved.

Finally the nuggets are checked for macro expansion and executed on the client side, potentially leading to more calls to the server to utilize services such as synchronization and history and awareness. The client side should support interpreters for the nugget macros, templates and language definitions.

The final result of the nugget executions is the nugget materialization, providing a visualization (and new modes of interaction) for the users.

6.4 HMS Architecture Implementation

The approach and motivation to use a wiki model as the basis to prototype the HMS model are based on the reasons described in Chapter 5 and Chapter 6.

6.4.1 Extending Wiki

Wikis are generally self-organizing websites, where everyone can edit existing pages and create new content any time. The interesting thing here is that the initial pages can be seen as seeds for other users to read, edit and contribute, and therefore errors can be found and corrected in this process. As can be seen in the most successful wikis such as Wikipedia (Wales and Sanger 2010) or TvTropes (Tropes 2004), after several evolutionary cycles articles usually become accurate and complete.

The open editable structure, pages as basic units of sharing, existing documented architecture models and implementation of traditional wikis, make them a good starting point for prototyping the HMS model.

Typical wikis lack support to create new domain-oriented tools within the wiki itself. Existing wiki engines only allow users to enter passive content and do not allow users to customize wiki pages. Therefore wikis cannot be used to host or author rich dynamic and interactive content (Krahn et al. 2009).

Application Wikis and extensible wikis

On the other hand, application wikis enhance wiki systems with lightweight programming features that aid in making data structure and processes explicit. Using these features, end-users can program a wiki to better support their collaboration (Anslow and Riehle 2008), for instance SnipSnap (SnipSnap 2003) and XWiki(Xwiki 2003). This approach, however, is limited since language constructs are often domain specific (Anslow and Riehle 2008) and users are constrained to write code in markup languages rather than to interact and inspect objects (Ingalls et al. 2008; Krahn et al. 2009).

Many wiki engines today support simple computational tasks in an ad-hoc fashion. MediaWiki offers parameterized templates (mediaWiki 2002) and TWiki offers macros (Thoeny 1998). Some wiki engines have been specifically set up to support general computational tasks, for example LivelyWiki (Krahn et al. 2009) and XWiki (Xwiki 2003).

Current solutions for end-user programmable web applications are either designed for limited use cases or are still too general for most non-software engineers and too much oriented towards a software development methodology - e.g. Lively Wiki and its extension Lively Fabrik according to (Razavi 2010).

It is worth mentioning here that LivelyWiki is a close cousin of MikiWiki. It makes use of JavaScript as implementation language and enables to build web client-side applications. LivelyWiki emphasis is on the client-side development in order to support end users programming. Nevertheless, it is different from the initial goal of this research, since meta-design addresses underdesign, starting from something familiar (wikis), and simple while becoming comprehensive over time rather than vice versa. Other differences are that LivelyWiki does not support synchronous collaboration and is focused more on the tailoring of shared artifacts than the tailoring of communication. LivelyWiki emphasizes direct manipulation techniques to design websites within a software engineering context, whereas I focus on designing communication with a minimalist open development approach.

In short, in order to evolve current wiki engines into general platforms for empowering end-collaborative design, wiki markup needs to be extended to allow for the expression and execution of programs at run-time.

Towards a Reflective Meta-Wiki

In this section I will explain how the HMS architecture extends wikis with typed pages (Correia, Ferreira et al. 2009), allowing users to link structured text and data pages to specific templates and layouts. This mechanism allows users to incrementally evolve part of the wiki text from informal text to structured contents.

Exploring the ability for users to redesign the platform as well as the content reflects on the meta-design concept, which is the focus on extending wiki editing beyond text into other media - one of the more limiting aspects of most wiki platforms.

A meta-reflective wiki aims at supporting meta-design and end-user tailoring. The HMS architecture system is open and it goes beyond templates, allowing advanced users to process structured content by defining their own page types using JavaScript.

This points towards a more radical approach to open-source, where the code is further opened by actively encouraging engagement at the user level. The nesting of 'nuggets' within pages helps confine any mis-coding by novices that could otherwise have negative impact on the platforms operation at a system level.

6.4.2 Technology

MikiWiki is implemented as a Ruby (Ruby 1996) web application written on top of the Sinatra framework (Sinatra 2008) on the server side and as HTML and JavaScript on client side, making ample use of the JQuery framework (jQuery 2006) and its plug-ins. JQuery framework is easy to use and very powerful to manipulate DOM elements.

JavaScript was chosen as the nugget creation language for several reasons: the code is interpreted on the client's side, and various libraries are available for building cross-platform and cross-browser JavaScript applications. JavaScript can be utilized for rapidly prototyping

mashups that require client-side processing, ranging from data aggregation, alternate UIs and alternate views of data and real time monitoring (Wong and Hong 2008). In this case, MikiWiki becomes a programmatic interface and a medium for providing services.

As MikiWiki runs in a browser, users can access MikiWiki anywhere and anytime from a multitude of devices, to modify or create all type of contents as well as share with other communities.

There is a growing popularity of frameworks such as node.js (Node.js 2009), enabling JavaScript on the server side, which I hope to leverage in the future to customize centralized systems. Although JavaScript supports dynamic scripting, it is seldom used to empower end users to influence and even change the behavior of Web programs.

Client – Server Responsibilities

In MikiWiki, the server side supports the minimal amount of features and services that maintain basic functionality. The server side handles all the tasks and rules related to the page and environment infrastructure, the basic navigation framework, authorization and notification services, while the client handles the rendering and the management of the interaction with the users as well as the way nuggets interact with one another and their calls to server side services.

Some functionality cannot be removed from the server side: all the basic facilities that handle the sharing of information, since the server acts as a repository and single aggregation point for wiki pages and web interfaces do not support peer to peer communication between users yet. Whenever a feature does not necessary have to be run on the server side, it can be expressed as JavaScript code within a wiki page. This allows the feature to be available for inspection and tinkering by the users, who might decide to customize it for their environment. Centralized services can be accessed by the client-side features via AJAX calls.

Loading and Rendering Mechanism

The server-side architecture strives to be simple and minimalistic. The server does not process any semantic information relative to the content of the pages or the meaning of the data that is being served. The associations among visualization page, data page and format page are controlled and authorized by users, while the server only keeps track of these relationships rather than their semantics.

Once the pages are loaded on the client side and the embedded nuggets are expanded, then the page content is interpreted according to the format of the data page.

Embedded nuggets are executed within the browser as JavaScript code. When a resource is requested during the rendering of a nugget, static text or JSON content and related dynamic JavaScript are retrieved, passed to the rendering engine, rendered in HTML format and displayed in the user's browser.

Figure 56 illustrates a lifecycle for fetching and rendering a page in MikiWiki. A user accesses MikiWiki by loading the URL of a page within the Web browser. This request gets sent to the MikiWiki Server.

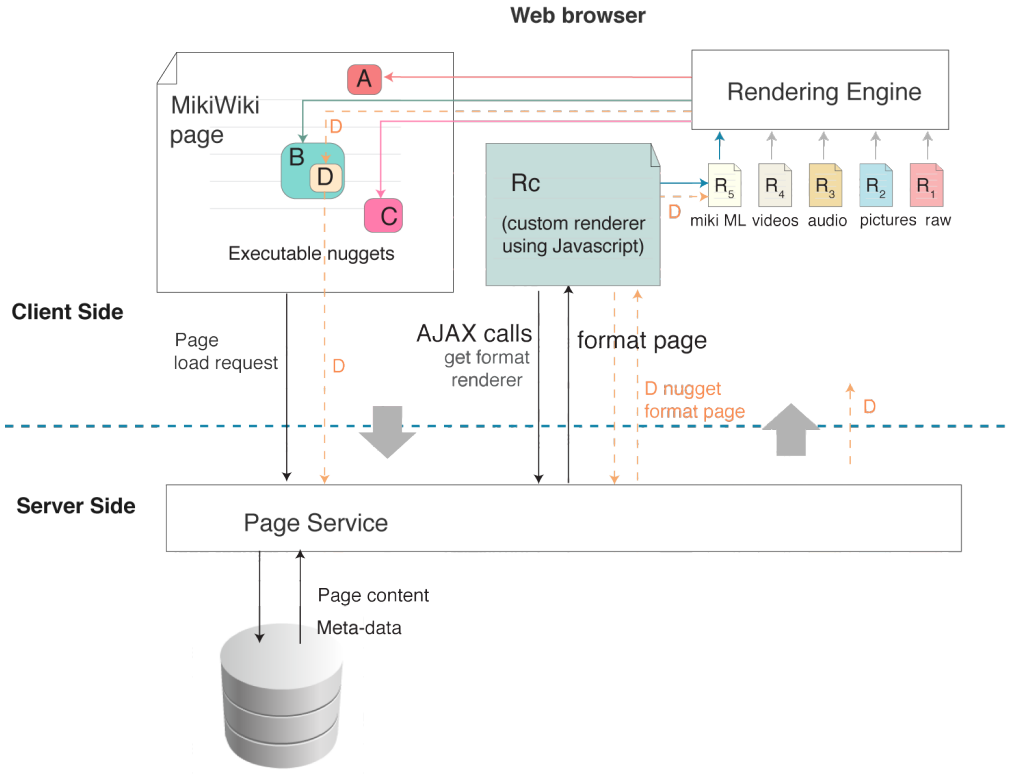


Figure 56 Rendering life cycle

A page load request is routed to the Wiki Page Service, within the Services layer. The Wiki Page Service takes care of fetching the page content and metadata, and answers queries related to a page and its context: for instance whether this page has child pages or a parent page, whether it is an environment, etc.

The information returned to the browser might not be simple HTML, since it possibly contains nuggets, which are placed in hidden DIV tags. All the nuggets must be interpreted, expanded and rendered to become visible HTML within the page.

Once a page is loaded, the MikiWiki Rendering Engine is activated. Every nugget in the loaded page gets passed to the Rendering Engine to be executed and materialized according to visible HTML code.

The Rendering Engine firstly checks the format metadata of the nugget to see whether the predefined rendering strategies can handle it. The basic rendering strategies support pictures, videos, plain text (to be used when editing), MikiWiki Markup Language, templates and JavaScript.

If the format does not fall into any of these rendering categories, the Rendering Engine falls back on a custom rendering strategy. The name of the format is then interpreted as

corresponding to a MikiWiki page, containing detailed rendering instructions in the form of a MikiWiki template or JavaScript code.

The Rendering Engine fetches the JavaScript contained by the format page via an AJAX call and executes it in the browser. The JavaScript may further make its own AJAX calls to the MikiWiki services in order to retrieve all the information needed to perform its rendering for instance, fetching further pages containing data in a JSON format, querying the environments or getting user profile information, checking file versions and so on.

Finally, the JavaScript produces the HTML code and makes it visible. If the returned HTML code contains further boundary objects, the expansion step is executed recursively until all nested nuggets are fully expanded and rendered in the Web browser.

6.4.3 Layered Architecture

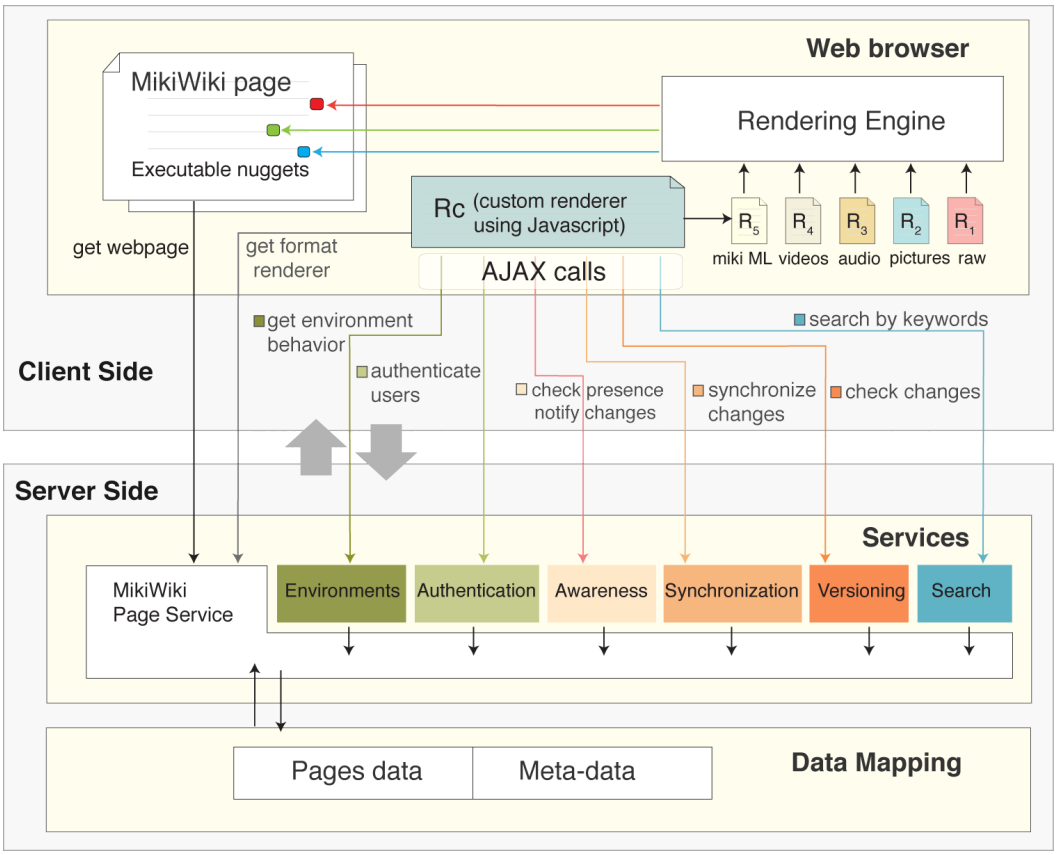


Figure 57 MikiWiki architecture

Figure 57 shows the MikiWiki layered architecture. The server provides some additional functionality to user generated pages on the client side by exposing an AJAX interface to authentication, synchronization, awareness, versioning and other services. They are the same services from the Hive Mind Architecture. I added some additional services like Tagging and Registration that do not appear in the conceptual architecture, but that we need in the implementation.

The service layer provides MikiWiki page CRUD functionality (Create, Read, Update, Delete), environment assessment information, authentication of the user, enhancing awareness, notifying changes, synchronizing communication, tracking versions and searching, tagging and uploading media functionalities.

Services provide users with some essential functionality expected in modern collaboration systems, focusing on the essential aspects of communication, configuration, coordination, information access, interaction and usability (Mills 2003).

6.5 Nugget Organization

Although each nugget can be accessed via its unique URI, we need to organize them in such a way that an end-user can navigate them and explore the space they create.

6.5.1 Organizational structure

The HMS does not specify the shapes of the boundary zone, boundary objects or environments. It specifies 'bordering' and 'containment' relationships, but not the nature of the space or its navigation. The focus of the HMS model is not on the space metaphor, and thus in MikiWiki I chose the simplest interpretation of space that was consistent with the model. The collaborative space is represented as a tree structure of pages, allowing the organization of artifacts in environments and sub-environments.

There are two main ways of structuring content within MikiWiki: either using the default MikiWiki pages or the *panel* nugget.

MikiWiki Pages as Organizational Structure

MikiWiki extends wikis with typed pages (Correia et al. 2009), which allows users to link structured text and data pages to specific templates and layouts.

In MikiWiki a page represents a user's view at any point in time and it can be associated to the web page corresponding to the MikiWiki page path. Pages are the fundamental unit of the experience as defined by (Curtis 2009).

The MikiWiki system is open and it goes beyond templates, allowing advanced users to process structured content by defining their own page types using templates and JavaScript. This mechanism allows users to incrementally evolve part of the wiki text from informal text to structured contents.

Information in MikiWiki can be organized via pages. MikiWiki pages can act as the materialized representations of boundary objects and every page can easily embed other pages. CoPs can manipulate, create and reuse pages created by other users by referencing their URL, uniquely identifying them.

A page can be either a “Folder page”, used primarily to group and navigate sub-pages, or a “Visualization page”, used to convey users’ content. From within a folder it is possible to create new pages and to upload pictures and other assets.

Panel Nugget as Organizational Structure

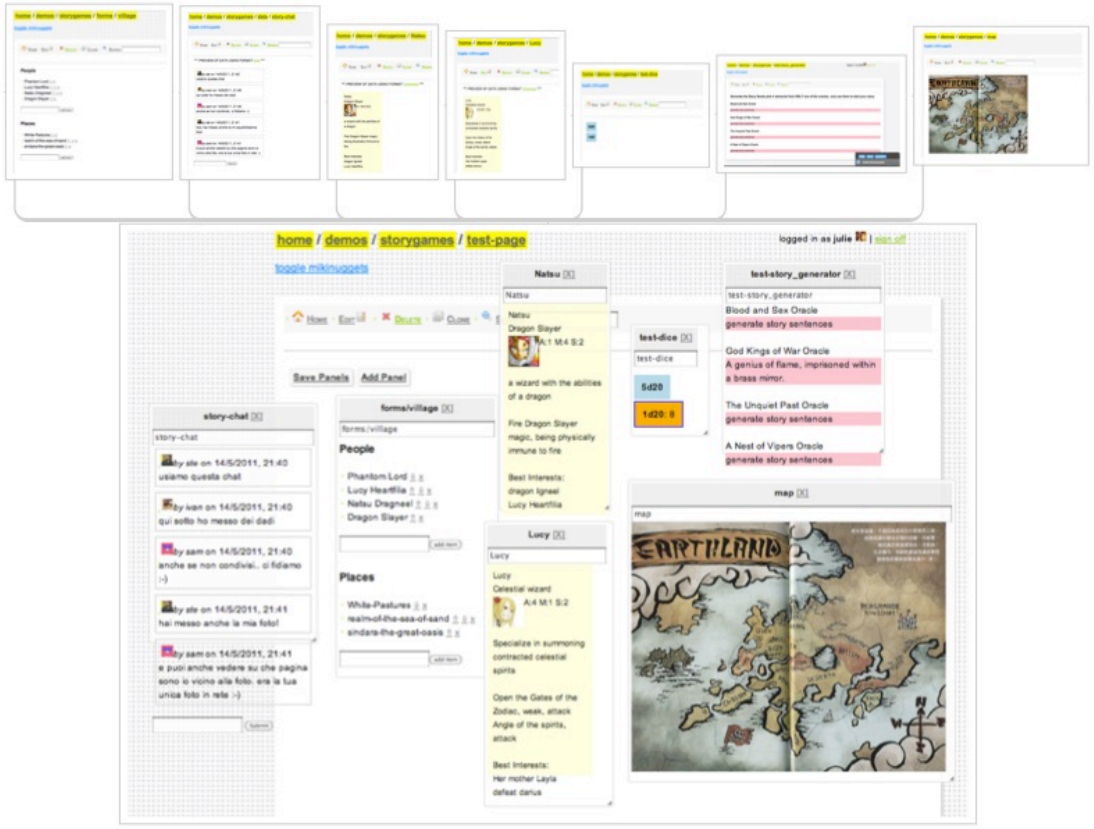


Figure 58 A RPG environment constituted by various MikiWiki pages

Information can be organized with the *panel* nugget. The *panel* nugget allows users to create floating panels and use them to embed MikiWiki pages. It provides users with the flexibility to combine and reuse different MikiWiki pages and to organize them within a workspace.

Figure 58 demonstrates a role-playing game environment is made of a set of MikiWiki pages, representing respectively, a form, a chat, two characters’ sheets, two dice, a story seeds generator and a map. In RPG environment, players can easily create another page panel to include new MikiWiki pages to support playing games. On the other hand, each MikiWiki page has its own individual data, which is independent from the RPG environment. This loosely coupled environment not only allows players to retrieve all useful information from and within MikiWiki instantly, but also provide them freedom to modify individual MikiWiki pages without destroy the environment.

This example illustrates how nuggets can be composed of other nuggets. As the RPG environment is itself another MikiWiki page, other nuggets can further refer it to.

6.5.2 Nuggets: Visualization pages, Data pages, Format pages

MikiWiki pages are access points to the deeper level of the system. They are the building blocks of a complex, multi-level medium that can be constantly tailored by users, thus allowing the system structure and behavior to evolve. MikiWiki supports three basic types of pages: text pages, data pages and format pages. All of them act as boundary objects, in the sense that they can be shared and extended by users.

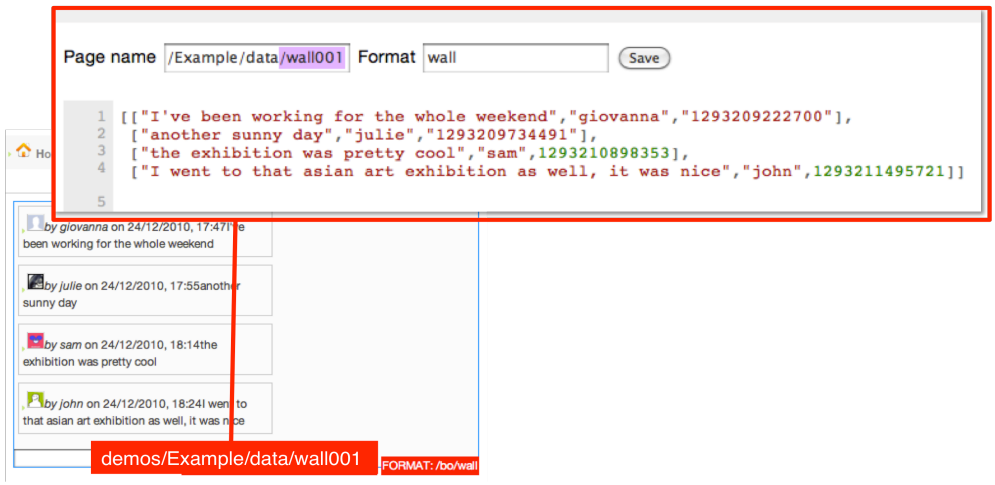


Figure 59 A wall data in JSON format

Visualization pages can embed other visualization pages or data pages. Data pages are typed pages with structured text (either in JSON, XML or any text convention that the user chooses) and they are associated with a specific format page. A format page can be either an HTML template with insertion points or JavaScript code. A format page is basically a set of rules or some code that defines how to render a data page.

Figure 59 shows a rendered visualization page containing a *wall* nugget, in analogy with the Facebook profile wall. Advanced users can choose to highlight the nuggets embedded within a page and make visible a link to the pages containing their data (wall-data in the figure) as well as their behavior (wall in the figure).

These links facilitate and stimulate advanced users to access meta-level functionality and modify or clone existing functionality and tools. Figure 60 and Figure 61 show the intertwined relationships among the visualization page, data page and format page. The format page defines how to visualize the JSON data and the visualization page embeds and displays the rendered result.

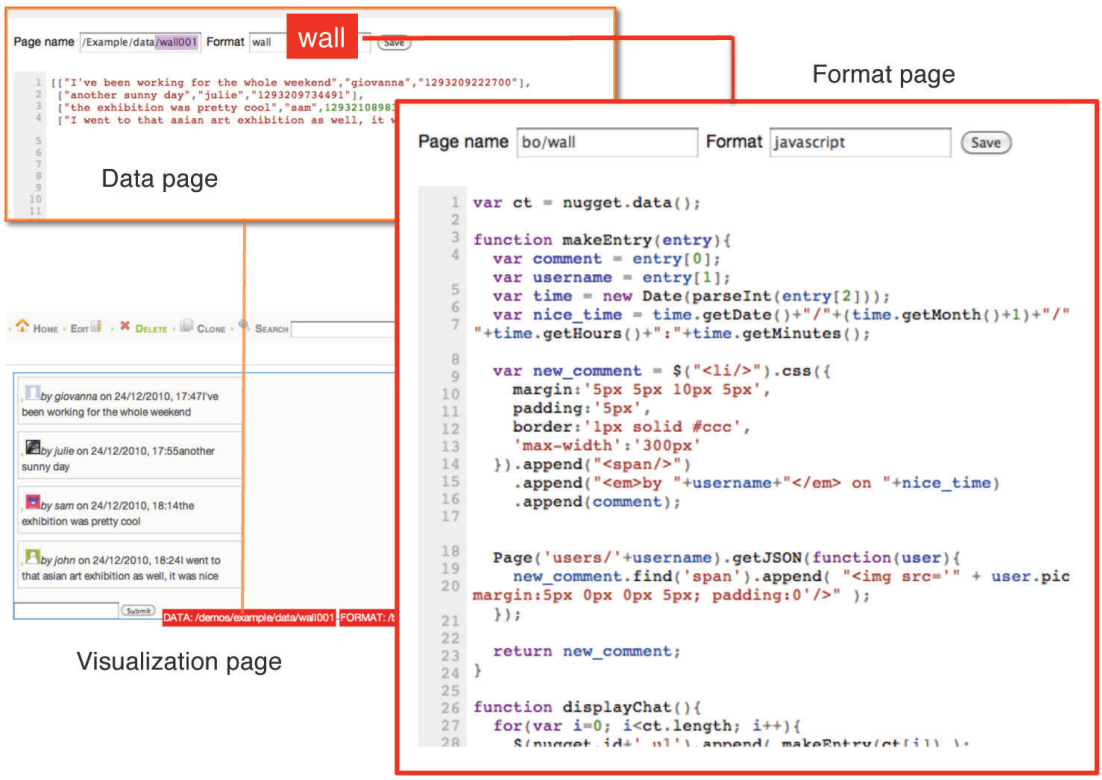


Figure 60 Visualization, data and format page of a wall nugget

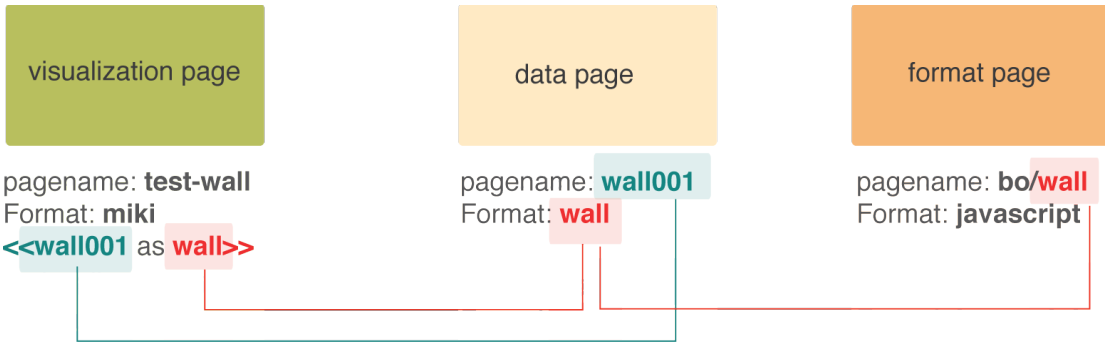


Figure 61 Relationship among visualization page, data page and format page

6.5.3 Environment

A folder can be promoted to an environment, a basic mechanism to manage and organize boundary objects in the HMS (Section 4.2.1). The members of a CoP can tailor an environment to reflect their thinking and workflow. Within this local environment, a CoP can create pages related to the tasks at hand as well as create sub- environments. Also, in MikiWiki, the environment is the context in which nuggets are and can be fetched. The properties of an environment are shared between all its children.

Lookup Mechanism

Instead of providing an absolute path for a wiki page, one can just specify the name of the page. This is particularly used in specifying the format of a page by writing, for example, *chat* rather than *my-environment/bo/chat*. The system finds that specific page via the lookup

mechanism. If the same format page is placed in the root of two environments, data will be displayed in the same way in both environments. However, if the user decides to override that representation in one of the environments, the user can put another format page with the same name within an environment, knowing that it will always be picked up in preference to the global nugget within that environment.

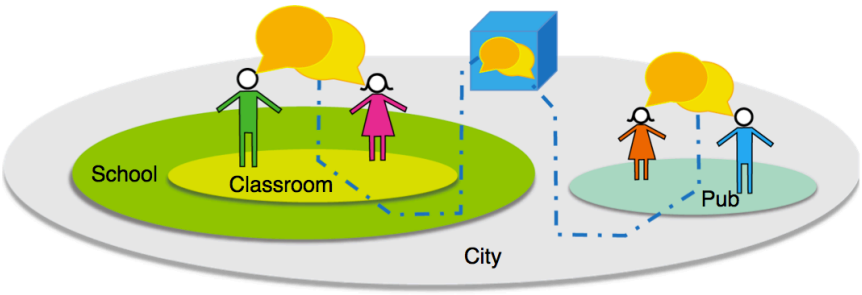


Figure 62 Utilizing the same initial chat nugget

Figure 62 illustrates how a chat nugget can be created in the city environment. Both the school and the pub can use the same chat nugget definition, yet with their own chat content.

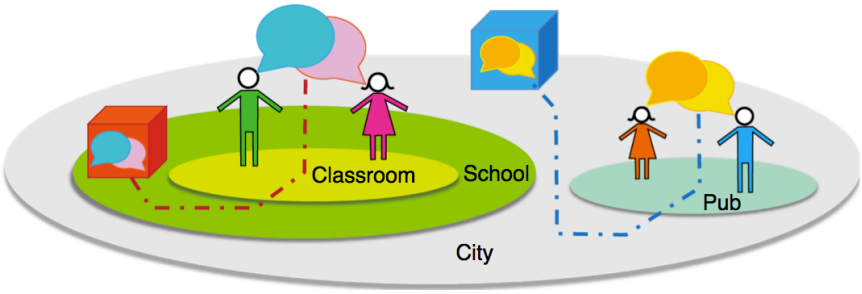


Figure 63 Localizing the chat inside the school

Figure 63 shows that after creating a new chat definition in the school environment, the school will picks up the chat defined within the school environment, which overrides the previous city ones. The new chat, local to the school, could act extra features such as the logging of lessons, or the capability for the teacher to temporary mute discussion, letting only one person speak at a time.

Figure 64 shows how the lookup mechanism works in abstract terms. If you are using a chat from within the alpha-2 environment, how can the *chat* nugget be retrieved if it is not contained inside the environment itself? The system first looks in the alpha-2/bo folder for a nugget named *chat*. If it does not find *chat* there, it looks up to the alpha parent environment, within the alpha/bo folder. If the *chat* nugget is not found in alpha environment, either, the system then continues searching up to the global scope, finally looking in the bo folder of the root environment, where the *chat* nugget is finally found, at which point the *chat* nugget is retrieved and activated. If the *chat* nugget had not been defined in the root bo folder, MikiWiki

would have provided a link to an empty page, offering the possibility of creating the chat nugget.

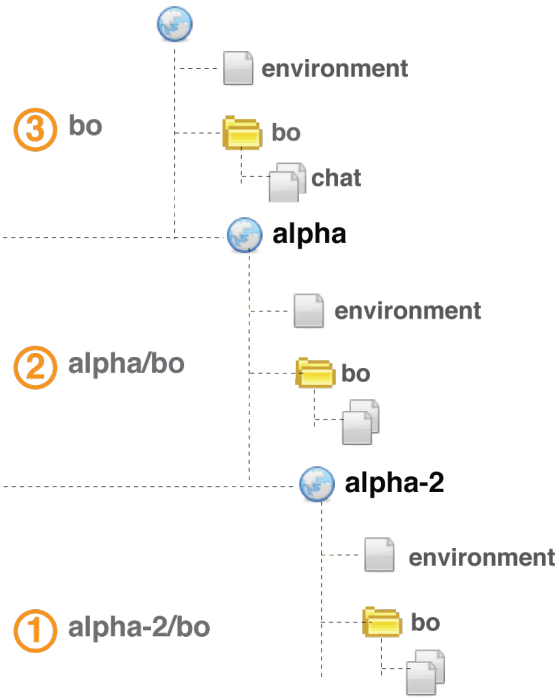


Figure 64 Lookup mechanism

The lookup mechanism is simply a way for a nugget to be looked up by systematically checking a hierarchical set of environments. Notably, the scope lookup ends when the nugget is found in the nearest available location of the chain, even if the same nugget name is used up further up the chain. In other words, the lookup mechanism always fetches the nugget that is more local to the point where it needs to be used.

6.6 Services

Database

The storage of MikiWiki files in plain text format. This enables us to read MikiWiki files with any text-reading tool regardless of the state of computer systems. I did not want the documentation locked in a database format that I could not read without a database server, forcing me to go through complex steps of data migration whenever I made a substantial change to MikiWiki.

The two essential concepts of file systems are files and directories. MikiWiki has a tree structure. The hierarchical structure of the files is by no means to impose a hierarchical social structure rather it provides a very flexible way to allow CoPs to create their own self-organized environments as well as evolve them over time.

Data and Metadata

Pages are stored as a couple of files: the raw data file and the metadata file. The raw data file stores the content of the page, be it a simple textual content, a JSON data representation, javascript code or an image. The metadata page stores information that allows the MikiWiki to know how to handle and access the contents of the raw file.

A typical meta-data page has the following information: page format information, and the owner and creation time or date of last modification.

```
--
format: miki
user: john
update: 2011-07-04 17:57:44.406725 -07:00
```

Authentication

Metadata can be used to check users' rights to access files. In this case, apart from the general meta-data information, namely, page format information, and the owner and creation time or date of last modification, the metadata file includes access control information - i.e. who can access this page and what kind of editing rights certain users can have.

```
---
allow:
  readwrite:
  - designer
  readonly:
  - mike
format: miki
update: 2011-11-01 20:59:49.099338 +01:00
user: admin
```

Search

The metadata file is used by the Search service and by the Tag nugget to store and retrieve tags relevant to search content.

```
---
format: miki
tags: book,arthitecture,design,MikiWiki
update: 2011-11-23 23:08:27.031930 +01:00
user: admin
```

Authoring Nuggets

This section describes how I implemented the nugget languages of the abstract HMS architecture in Section 6.2.

1) Composing nuggets

In MikiWiki, every page is a nugget, and it can be embedded in another page. The simplest way to create a new nugget is to use the default *miki* format, the MikiWiki markup language (MikiML). The *miki* format supports the rendering of basic wiki syntax, including page links and includes, but it also allows the embedding of other nuggets. The double angular brackets syntax is used to embed other nuggets, which can be other textual pages or more exotic custom made formats like video, chat, etc.

For example, by including a *video* nugget, a *vote* nugget, and a *comment* nugget in a MikiWiki page, one can create a *youtube* page, showing a video and allowing people to vote it and comment it, mimicking some of the real YouTube functionality. This page is a nugget, an "aggregated" one that can be embedded in other contexts.

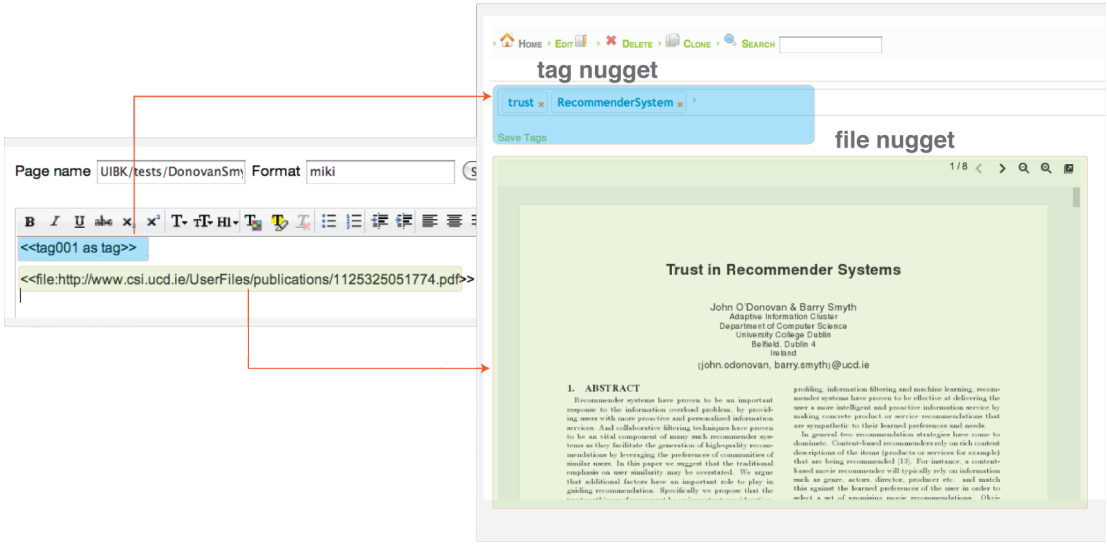


Figure 65 Using MikiML to include a tag nugget and a file nugget

Figure 65 shows a nugget made during the Aristotele experiments detailed in Chapter 9. In this example a new nugget is built by composing a PDF file –imported with the *file* nugget- and a *tags* nugget, used to index it within the system.

MikiWiki directly supports textual composition using the miki format, but it is also possible to create the *panel* format, which allows to import several nuggets and position them within the browser by dragging, as opposed to placing them within text. An example of the use of panels can be seen in Figure 58, in Section 6.5.1.

The parsing of the MikiML format is performed on the client side by JavaScript code, theoretically leaving it open to new syntax and language expansions. This code is encapsulated as a *miki* nugget in the feature, leaving only JavaScript as the primitive nugget format that does not need resolution.

2) Extracting Templates

As (Cypher 1993) argued that computers are good at performing repetitive activates, it is ironic to leave computer users to perform all of the same low-level repetition, instead of computer. Solutions are needed to enable users to create their own custom commands. The techniques for achieving this goal are commonly referred to as “end user programming”, which does not necessarily mean to program per se, rather to achieve effects that can only be achieved through programming. Extracting templates allows users to create a set of pages that share the same structure. Web applications often utilize HTML templates to separate the webpage presentation from its underlying business logic and objects (Tatsubori and Suzumura 2009).

Taking the example from the previous section, one could further make the combination of the PDFs and tags reusable with a less customization effort. Ideally users would give it a new name and parameterize the variable parts. MikiWiki supports this by the means of the *template* format.

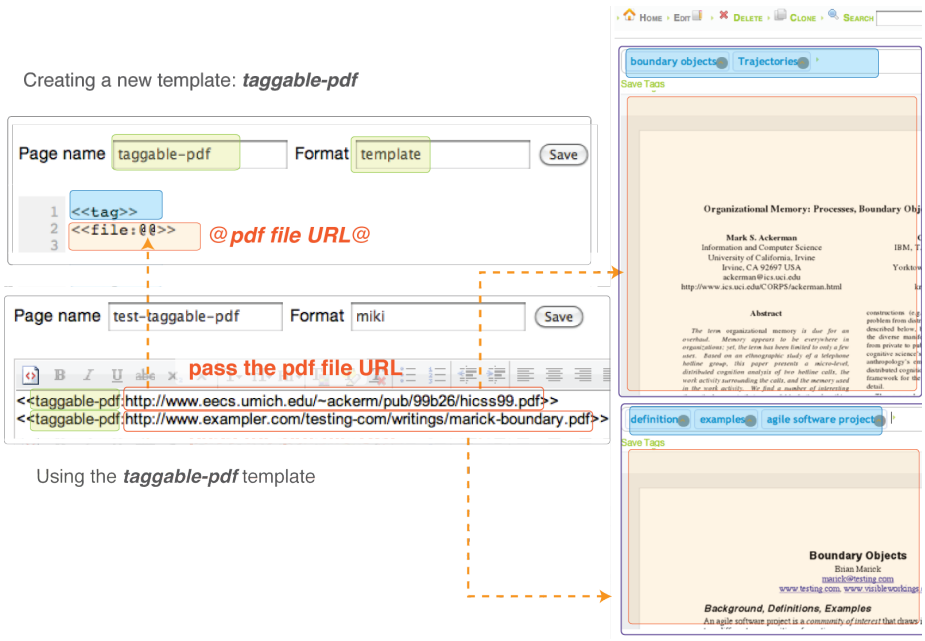


Figure 66 MikiWiki Templating mechanism in action

The *template* format builds on the *miki* format by adding parameters to it. When a nugget with template format is embedded, the run-time system checks if any parameters were passed to the nugget and substitutes them in the text of the template, generating a new custom page on the fly. In the PDF and tags example, the names of the individual PDFs can be removed from the page, put in a separate data page that can be individually shared and embedded. This data page will refer to a new *taggable-pdf* format. The taggable-pdf page will contain the base page template and will be declared as being of *template* format (Figure 66).

Providing the possibility to extract repeated patterns and create templates empowers users to create new combinations of nuggets to use over time and share with other communities as ready-made solutions.

3) Tweaking and Creating Nuggets

A further step is to add new behavior to nuggets to support unforeseen complex interactions. This is done by substituting the template page with a *javascript* page. The JavaScript can be edited directly within MikiWiki with an embedded code editor and it provides access to service libraries and to all the services exposed by the MikiWiki server.

The following code creates the simplest nugget in MikiWiki, using the *javascript* format to print out “hello world” to a MikiWiki page.

Since the *nugget.setHTML()* function outputs HTML in a very specific insertion point determined by the containing page – be it a MikiWiki page or panel - all kind of interactivity allowed by JavaScript can be added to nuggets. Additionally, the MikiWiki architecture provides a number of services that can be exploited to access the data of other pages in the background, perform synchronization, explore data history, etc.

The first time a user encounters the *javascript* format is most likely to be when existing nuggets need to be modified. Tweaking nuggets can be done by opening up the format page of a nugget and directly modifying the JavaScript code or by merging different piece of code together to create a new nugget. More examples of this kind of customization will be introduced in the design studies.

User-generated JavaScript code can be used to create more complex nuggets via exploiting the server-side services. Appendix B explains MikiWiki APIs that can be exploited by advanced users to create more complex visualizations and behaviors, designing more comprehensive nuggets.

6.7 Conclusions

In this chapter, I have described the essential attributes of the HMS architecture. It is needed to note that the HMS model prototype does not necessarily to be a wiki, but it can be any platform that supports the premises for a rich flexible medium of interaction. Based on this research focus and a limited time period, I chose a wiki-like platform as ideal for prototyping the HMS model, exploring its concepts and develop a reference architecture.

First of all I presented the conceptual principles underlying the conceptual architecture. Pages and nuggets are the basic units for designing other pages and nuggets, building the HMS

model and constructing communication over time. Various services should be designed around nuggets in order to support design activities.

After outlining the architectural design I explained the detailed architecture implementation, how each HMS architecture feature is materialized and how APIs can be used for further tinkering and development.

Chapter 7

The following three chapters introduce two design studies, the Energy Feedback System Mockup project (Chapter 7) and the Aristotele project (Chapter 8), and one evaluation study, the Creativity Barometer evaluation (Chapter 9).

The objectives of the two design studies are:

- Conducting a case-based incremental implementation of MikiWiki over an extensive period of time and evolving MikiWiki in practice;
- Investigating how MikiWiki helps solving design tasks in a distributed collaboration context and how MikiWiki complies with the HMS model characteristics.

Over time MikiWiki has become comprehensive and flexible enough to accommodate emergent issues. The Creativity Barometer evaluation (Chapter 9) examines how MikiWiki supports collaborative design in a co-located context and in particular focuses on why and how it fosters creativity.

This chapter presents the first design study involving the design of a mockup for an energy feedback system. It demonstrates how embodied HMS model features in MikiWiki support collaborative design and appropriation based on observation and feedback from participants. Some insights and limitations will be discussed at the end.

7. Design Case Study-1: Energy Feedback System Mockups

This was a collaborative project between the CSLab of the Department of Computer Science at the Università degli Studi di Milano and the L3D of the Department of Computer Science and the Institute of Cognitive Science at the University of Colorado Boulder. Meta-designers, and designers were tasked to collaboratively design an energy feedback mockup environment.

7.1 Context and Goal of the Design Study

The increasing challenges presented by climate change and energy issues urge us to reflect upon the unsustainable lifestyle we all engage in unconsciously on a collective level. It is crucial to find ways to foster sustainable changes in behavior that lead to a reduction in energy consumption. However, the invisibility and abstract nature of energy makes it difficult to be aware of and understand its usage. Indeed, it is difficult for consumers to connect their daily activities to energy consumption and monthly energy bills without more transparent cues to energy use (Burgess and Nye 2008). Extensive research has shown that energy usage can be reduced by providing consumers with feedback on their consumptions, especially continuous or daily feedback correlated with higher saving results than monthly feedback (Abrahamse et al. 2005; Darby 2006).

Researchers at L3D have been actively researching the energy domain. They have worked on meta-design environments to support consumers by simulating and visualizing their energy consumption in real time to understand the impact of their behavior impact, as well as analyzing and sharing individual behavior in communities, with the goal of achieving energy sustainability via collective behavior (Fischer 2011; Dick 2011). As a visiting researcher in L3D for five months, I had great opportunities to learn from them about energy related socio-technical issues. I decided to choose energy as one of the MikiWiki application domains in that it provides me a concrete design context to test and evolve MikiWiki in practice. Additionally, MikiWiki can be used to prototype an energy feedback system as it provides an evolutionary design approach for exploration. It supports distributed participants with different skills in collaborative design, rapidly prototyping and trying out energy feedback system design. Design results can be easily stored, shared and discussed among participants within MikiWiki.

The goal of this design study is to explore how MikiWiki helps in solving design tasks and how MikiWiki complies with the HMS model characteristics as well as to evolve MikiWiki in practice.

7.2 Project Timeline

The project was conducted from 23rd May to 31st June 2011. It can be divided into three phases.

Learning by playing: in the first phase (23rd May-06th June 2011) the meta-designer introduced MikiWiki and its functionality to the designers, encouraging them to create individual environments and to play with nuggets. The designers were shown how to access all three levels of participation, namely *use*, *design* and *meta-design* levels, acquiring a basic understanding of the design rationale behind MikiWiki and how to make use of the different modes.

Designing for design: during the second phase (07th June-17th June 2011) participants investigated existing energy feedback systems and defined the requirements for their energy mockup environment. After gathering the requirements, they started giving shape to the energy mockup environment, adapting and evolving MikiWiki, inquiring the meta-designers for information related to MikiWiki usage.

Blending design and use: during the third phase (18th June-31st June 2011) the designers used the mockup environment to finalize their energy feedback application interface design. During this phase, they needed to evolve their mockup environment to implement their design ideas and cope with problems. In this phase environment design and application design activities were continuously interwoven.

7.3 Participants and Design Tasks

The project was done in cooperation distributed with the first year undergraduate students from the Department of Computer Science at the Università degli Studi di Milano. They carried out the design project with the support of two meta-designers - myself and a web-designer from a web development agency. Table 7 depicts participants' profile information. The participants were divided into roles according to how they used MikiWiki. The role of designer in this case was associated with graphic, interface and interaction design; the role of users was using MikiWiki to communicate with other participants and document design processes as in a normal wiki; and the role of meta-designer was involved in scripting, appropriating nuggets at a code level.

Table 7 Participants' role and profile information

Participant (Gender)	Age	Main Role	Education and Expertise
1(M)	20-25	User and designer	Undergraduate student in computer science without any programming experience, with some knowledge of web design and communication design
2(M)	20-25	User and designer	Undergraduate student in computer science without any programming experience, with some knowledge of web design and communication design
3(M)	20-25	Meta-designer and designer	Undergraduate student in computer science with programming experience
4(F)	20-25	Designer and user	Undergraduate student in computer science without any programming experience, with some work experience and interest in web design and graphic design
5(F)	30-35	Meta-designer	Researcher interested in EUD development with comprehensive web programming knowledge, in particular JavaScript
6(M)	30-35	Meta-designer	Web developer with extensive web programming experience

The design task for the participants in this case study was the following:

1) To design a generic mockup environment related to energy consumption for the iPad tablet platform. Choosing the iPad suggests that the final application has to be designed for mobile devices as consumers can use it to monitor their house energy consumption anytime and anywhere;

2) To use this generic mockup environment to design the final energy feedback application. This application should allow consumers to visualize their energy usage data in a way that can be easily understood and is personally meaningful.

The generic mockup environment can be easily tinkered and evolved for prototyping other energy related applications by updating UI design elements related to the final application or by changing the iPad to another mobile device.

7.4 Data Collection

Questionnaires were used to collect the students' opinion on their collaborative design experience. Some guiding questions were:

- Whether MikiWiki supported their communication, coordination and collaboration as well as whether they were happy with their design results with respect to whether MikiWiki supported different aspects of collaboration;
- Whether students encountered difficulties using MikiWiki and how they solved problems by using and appropriating the system, with particular attention to the students' coping strategies;
- What their design tasks were and how students perceived their role over time, in order to understand whether MikiWIki supports different design activities and levels of participation;
- The most useful parts of MikiWiki and the differences between MikiWiki and other collaboration tools that they used, to identify the novel aspects of MikiWiki;
- Whether the design process encouraged users to become designers.
- Difficulties students experienced in using MikiWiki and how the system could be improved with respect to reseeding and evolutionary growth.

Additionally, MikiWiki can keep a log file recording information for each interaction in terms of: timestamp, user name, environment name, page name and its format (to understand whether a user modifies contents or code) etc. Notably, the log file was not used for a statistical analysis but rather to understand students' activities e.g. whether they encountered problems, when they might need help or when it was necessary to examples to provide support to stimulate their participation. D1 to D4 identify the different designers.

7.5 Observations on HMS Aspects

The following sections will present how MikiWiki complies with the HMS model, and how these aspects have been observed in this design study and exploited to support energy feedback system prototyping and evolutionary design-in-use.

7.5.1 Habitable Environments

At the initial stage, each student created her/his own environment, in which s/he could select nuggets of interest, try out demonstration examples and learn how to use MikiWiki. By creating a personal practice “environment”, students felt free to experiment and explore different scenarios, following their own learning pace without destroying or being influenced by other students’ works. On the other hand, all the individual environments allowed other students to access, so they could inspect one another’s progress. One of the reasons for creating these *personal* environments is because individual learning history and efforts could serve for final evaluation.

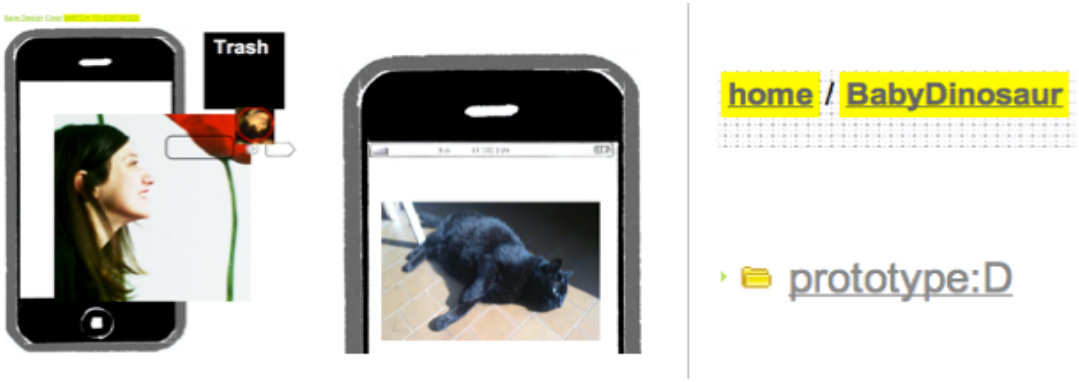


Figure 67 Adding fun elements

Within students’ individual *environments,* they preferred to introduce their personal interests or elements associated with their everyday life. For instance, In D4’s *personal environment*, she played with the iPhone mockup example by using her own and her cat pictures as design components (in Figure 67).

It is worth mentioning that in the shared public space, students tried to introduce fun elements by naming folders with emoticons “:D” or quirky names such as “BabyDinosaur” (Figure 67). Explicitly using a slightly quirky language and hinting at mischievous behavior could be used in future iterations of MikiWiki to support better emotional engagement with and memorability of the many nuggets available.

This example illustrates that MikiWiki allows users to create environments, organizing private and public social spaces.

7.5.2 Boundary Objects

As mentioned in Section 5.2.3 nuggets are small and simple building blocks that do not have fixed use context. Users are encouraged to appropriate them, assign new meanings to them and utilize them to create situated solutions. Several examples presented below demonstrate these possibilities.

Situated Appropriation

The color of a *note* nugget can be pink, yellow, green or blue, and the meaning of each color is socially assigned by users, rather than being predefined. A note nugget can be used in many different ways. Notably, the designers used the note nugget to create todo lists. Since MikiWiki did not support a calendar or scheduling nugget, the participants overlapped the notes on top of a calendar image (Figure 68). The color of the notes implied different levels of priority; for example the pink note indicates the exam time (24th June 2011). Students invented their own notation to indicate authorship and attribution: (Simo) indicates that Simone is the note creator while, as mediated from Twitter notation (Dorsey et al. 2006), @upiii means that the message should be read by upiii (Francesca's handle). "xD" is simply an emoticon signifying a smile.

Figure 68 Tasks calendar

Combining Nuggets in Unexpected Ways

Energy data was made available within MikiWiki from an external remote source, which is explained In Section 7.5.5. By utilizing an existing JavaScript charting library, energy data could be visualized in real time (Figure 69) in a MikiWiki page. The green line shows Watt usage, while the blue line shows Amperes.

I gave this example to students as a demonstration of the possibility of providing timely feedback and the consequent increased awareness of invisible power usage data, rather than expecting them to use this MikiWiki page as part of their design.

Figure 69 Combining two MikiWiki pages

Nevertheless, designers took advantage of the *panel* nugget (Section 7.5.5), embedding the energy consumption visualization in a floating panel and positioning it on top of their application interface to mock up their real-time energy consumption feedback idea (Figure 69). With a little help from meta-designers in removing the *panel* nugget dragbar, the final interface achieved a seamless integration between the iPad canvas and the energy visualization MikiWiki page.

Repurposing the Chat Nugget

Since students were often not online at the same time, they appropriated the *chat* nugget as an asynchronous notification channel rather than a synchronous communication tool (Figure 70). They used it to post messages, pointed out the most recent important changes, and passed links via chat. When others logged in, they could see the notes from the previous users and be aware of the news. During the second phase, designers removed the *chat* nugget, as they found that the popping up of the chat entries distracted their design flow. As one of the designers commented:

"*è importante che mentre si lavora non appaiano i banner della chat per non distrarre designer o developer che sia*." [D4]

["While you work, it's important that to no overlay sidebar comes to the front, distracting developers and designers."]

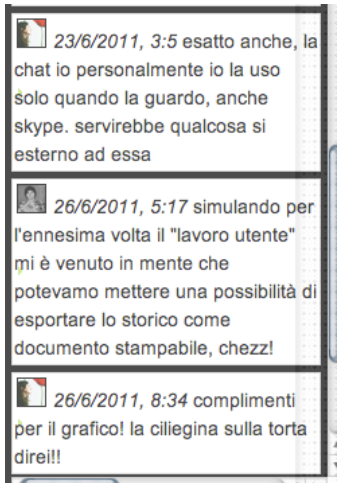


Figure 70 Using chat as a message board

Nevertheless, they included the *chat* nugget in the sidebar again at the last phase, when they needed to communicate intensively before the deadline. This example also reveals that there is not one ideal environment, but rather that features should be changed, enabled and disabled according to the situated needs of the users.

Bypassing the System

One of the issues students encountered was that they could not arrange the *toolbox*, the iPad *canvas* and the *trash* nuggets within the actual screen display size, since the iPad canvas image they created was too large to fit in most screens.

Figure 71 Mockup environment display view

Noticing this issue, the meta-designers created a smaller *canvas* nugget, which resized the iPad image size and the design elements dropped on it, allowing everything to fit within the actual window. Meta-designers notified designers of this example and assumed they would follow up.

Nevertheless, designers did not follow the example the meta-designers provided; rather they used the web browser key combination [ctrl][-] to zoom out of the mockup design environment

and make it fit (Figure 71). The reason could be that this was easier either at that moment or because of learning time (Dix 2007). In this case, it might be partially because they had already designed many MikiWiki pages for the big iPad *canvas* nugget and they preferred to find an easier and more efficient way to solve this problem than to reposition the design elements. One design implication here is to explicitly remove features that can be better accomplished by the socio-technical system (at a social level) in which MikiWiki is embedded.

7.5.3 Different Levels of Participation

Figure 72 shows the three levels of participation in relation to the iPad mockup environment.

Figure 72 Three different levels of participation

1) At the meta-design level, two meta-designers guided designers in using MikiWiki and provided different nuggets for designers to design the mockup environment, according to the different design phases.

2) At the design level (upper level), i) four designers created a mockup design environment. The creation process is collaborative and iterative together with the meta-designers. In this case, the mockup design environment is composed of three nuggets, namely *toolbox*, *canvas*, and *trash* written in JavaScript. The *energy-chart* nugget was added at a later stage. Designers performed some meta-design activities opportunistically, when they saw the opportunity for immediate advantage and without asking for permission from the meta-

designers. This is revealed from the both log file and new format pages they created. They analyzed and reasoned on the navigation and interactions of the final application.

At the design level (bottom level), ii) designers used the mockup environment to create an iPad mockup by simply dragging and dropping components and sharing their results with their team. In addition, other users could create their own *toolbox* nugget with customized design components, while the iPad *canvas* nugget could be easily replaced by a new mobile device or platform for another design case. At this level users and designers could vote, annotate and discuss the mockups. Users could create notes to ask for new features and modifications or to annotate the application interfaces with suggestions and comments.

3) At the use level, end users could use the application, customize their personal preferences, (e.g. devices, service provider, saving mode, and so on), and choose and refine consumption visualizations. This level is, however, outside the scope of the work presented here.

Dynamic Roles – Boundary Crossing

However, the levels of participation were not predefined, but emergent in terms of users' skills and roles, which were highly dynamic.

Students, D1 considered his role as both a designer and a mixed role.

"*[Il mio ruolo era] Principalmente designer ma anche ruoli misti*"

"[My role was] mainly designer, but also a mixed role"]

D2 perceived his role as a user and a mixed role; D3 considered his role as a meta-designer; and D4 considered her role as a designer. It is interesting to note how the role of users (especially the perceived role) changed fluidly according to the tasks, without ever requiring a separate environment designed specifically for that role.

In MikiWiki, all the design activities happened in the same space. The roles were not so clear-cut, as the same participant could move between a '*user'* or a '*designer'* role at the same time, and occasionally designers switched to a '*meta-designer*' role, to "get it done quick as long as it works" by making opportunistic changes to the tools. The designed mockups are wiki pages, acting as boundary objects containing other boundary objects, exchanged, shared and shaped within the design team.

7.5.4 Open Infrastructure

The open infrastructure of the HMS (Section 4.2.6) not only seeks to provide a more transparent structure but also allows it to be evolved and easily connected with external resources/communities.

In this design study, designers highly valued that they could immediately see the result of the code they wrote, despite two of them not having any programming experience.

"*La possibilità di condividere i vari step del progetto con gli altri collaboratori e la possibilità di fare modifiche anche importanti da codice*" [D4]

["I liked the possibility of sharing with the other people all the steps making up the project, as well as the possibility of applying changes – even important ones – via code."]

The following two examples present the advantages of utilizing external web services and connecting other devices.

Leveraging Existing Services

When creating their own custom toolbox, the designers had to upload their own images. However uploading a large amount of images can be a repetitive and tedious work. Thus, they extended the system with other existing platforms to circumvent the uploading process.

Lately, a fast and effortless ways to synchronously share files and media is to drag them to a public Dropbox folder located on the desktop (Houston and Ferdowsi 2007). Each file within the Dropbox public folder is automatically associated to a publicly accessible URL.

Figure 73 Utilizing Dropbox to integrate sketches

In order to use existing images stored in Dropbox rather than uploading individual images again into MikiWiki, designers took advantage of the *imagenote* nugget. An *imagenote* nugget takes an image URL as an input and embeds the image within a draggable resizable note.

Students sketched their design ideas on their desktop applications and put them in the Dropbox public folder. From Dropbox they copied the sketches' public links to the image notes, making them directly available to MikiWiki.

Talking to MikiWiki from the Outside

In order to get real-time energy feedback, researchers at L3D connected Kill-A-Watt monitors (Wikipedia 2008) with XBee adapter kits (Wikipedia 2005) to built Watt-watcher kits for monitoring energy consumption in the home (Fried 2011). Via a watt-watcher kit a computer can listen for a signal and compute the current power usage energy.

By utilizing a Watt-watcher, I worked as a meta-designer to modify a small external application (written in Python), which received the energy consumption data from a domestic house in Boulder (Colorado) via XBee and republished it in near real-time to MikiWiki, without modifying its infrastructure. MikiWiki acted as a bridge and integration point for an external system, namely XBee data stream integration.

The Python service simply logged in MikiWiki as a user and updated a specific page twice a second, with a small string of JSON data containing the energy usage readings, so that the students in Milan could use real data for their design. In this example, MikiWiki reveals its potential to be augmented with physical devices and as a platform for sharing open data.

Notably, "*open infrastructure*" was initially though to be able to allow communication with other design communities outside the HMS model or utilization of different web services in MikiWiki. But through this concrete implementation, it also means the flexibility to be augmented by physical artifacts or extended to the physical world.

7.5.5 SER Model

Underdesign of the HMS model is reflected in several aspects in this design study, for instance, modifying and creating new nuggets, providing means to allow students to create situated solutions and to shape their communication or working environments over time.

All designers appreciated the flexibility of MikiWiki and the possibility of being able to change things on the fly. The designers perceived the difference between MikiWiki – which was defined as being *alternative* - and conventional systems, where products have *final versions* and cannot be changed. As they pointed out:

"*L'esperienza del fare il progetto in questo modo alternativo". [D3]*

["The experience of doing a project in such an alternative way".]

"la parte parallela allo sviluppo (calendario, note, avvisi); la possibilità di modificare il layout senza mai avere necessità di renderlo "definitivo " [D2]

"[I liked] the part parallel to development (calendar, notes, notifications); the possibility of modifying the layout without ever having to settle for a "final" version."*]*

Incremental Implementation - Meta-designing Nuggets

All the nuggets can be accessed and evolved on the client side over time, as they are MikiWiki pages. To this end, not only meta-designers but also designers and users were able

to access them and collaboratively design-in-use. During the collaborative design process, the initial nuggets were extended (e.g. *note*, *imagenote*, *panel, notify* nuggets), and new nuggets were created according to specific requirements of this project and to better support students' design activities. The following examples focus on refining initial nuggets according to emergent social-technical issues.

Chat Nugget

The chat nugget was used to communicate both synchronously and asynchronously. It was used in combination with the *activeuser* nugget, which shows online users. Being aware who is online, users could decide whether to use the chat synchronously, to engage the online user, or asynchronously, leaving messages.

The initial chat can be seen in Figure 74, on the left. The chat entry background color is black while the text color is white. However, this created unforeseen problems when users started copying some text from the chat to paste it into a MikiWiki page, since the text color was the same as the MikiWiki page's background, i.e. white. Another issue was that URLs pasted inside the chat were not recognized as a link and therefore were not made clickable. The only cumbersome workaround was to copy and paste the URL to the browser address box.

Figure 74 Initial chat nugget (left) and evolved one (right)

The updated chat nugget is shown in Figure 74, on the right. One of the participants tweaked the *chat* nugget. Three changes have been made to the initial chat nugget: i) each chat entry's font color is black, allowing users to copy text to wikipages with a white background; ii) the chat entries' background color is changed to white to offer some contrast to the black font; iii) each entry is parsed and if a URL link is found, it is automatically made clickable in order improve contextual navigation.

Toolbox Nugget

Figure 75 shows the initially provided iPhone design toolbox, and the new modified version. In contrast to the initial *toolbox* nugget, besides different graphic representations of the design elements, the updated *toolbox* nugget has tooltips. Without tooltips, it was sometimes hard to understand the meaning of a design element since the relative scale was not preserved in the

toolbox (e.g.: is that keyboard really as big as a checkbox?). By showing an icon's name, designers can more effictively select icons and design the application interface.

Figure 75 Toolbox nugget

Canvas Nugget

The initial *canvas* nugget supported the design of iPhone mockups and had two links: *Save* and *Clear* (Figure 76, on the left). Since participants wanted to design for the iPad platform, they had to make a few changes, evolving the *canvas* nugget.

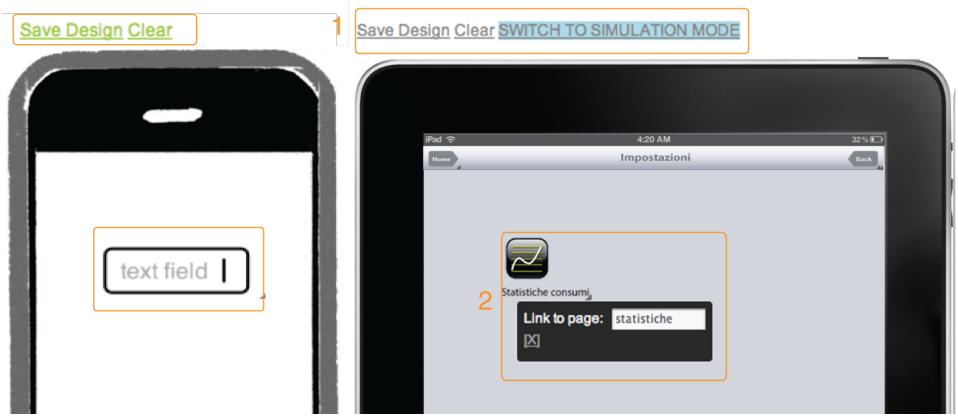


Figure 76 Canvas nugget

Once the mockups of a few pages were in place, the designers wanted to find a way to make them interactive, allowing navigating from one mockup to another. The meta-designers set up the new *canvas* to allow designers to switch between an editing mode and a simulation mode. In the editing mode, by double clicking an icon, one can specify which MikiWiki page links to that design element. In this example, the designers specified that an icon would link to the *statistiche* MikiWiki page. Once done with the editing, they could switch to the simulation mode, where they could click icons and be redirected to other pages with different mockups, simulating the navigation experience of the application mockup.

Panel Nugget

The initial *panel* nugget was used to create draggable panels, which could embed other MikiWiki pages. In Figure 77, the left image shows the initial *panel nugget*, with a grey dragbar and an input textbox allowing a user to specify the MikiWiki page name.

Since designers wanted to create a seamless integration between the energy consumption MikiWiki page and the iPad canvas, it was necessary to remove the dragging handler and the input textbox. The right image shows the refined panel nugget without these elements but displaying the same data, i.e. energy consumption (a MikiWiki page) however using different format page results in different representation. This was achieved by cloning and tweaking the initial *panel* nugget and telling MikiWiki to override the same data page with the new nugget visualization.

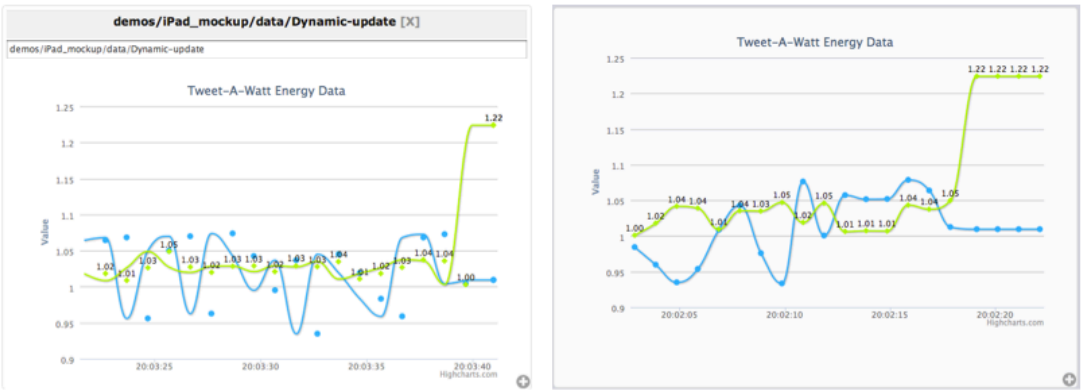


Figure 77 Panel nugget

Evolving Communication

Although MikiWiki supports versioning and notification, it did not provide clear instructions on how to compare different versions or how to notify changes. Hence it became challenging for users to keep updated on what was happening on MikiWiki.

As mentioned above (Section 7.5.2), during the first design phase, the sidebar chat was not only a tool for real time communication (chatting, making jokes, asking questions) but it also became an asynchronous notification channel. Designers posted messages, noted the most recent important changes, and passed links via chat. When others logged in, they could see the information (Figure 78).

In the second design phase, designers created a MikiWiki page, @UPDATE, as an information board where users could leave messages, things to be done, problems, questions, etc. Notably, by using the "@" sign in the pagename, the *@UPDATE* page could be listed before all other subpages in the environment, with much higher visibility than the other pages (Figure 78).

During the last design phase, there were three methods designers used to keep each other posted, the *chat* nugget, the @UPDATE page and the *notify* nugget. By using the *notify* nugget, each time a page was modified, all the designers would get a notification email (Figure 78). The email notification only showed that a page had been saved, modified, deleted or created and which user was responsible, rather than giving more information about what had been specifically changed.

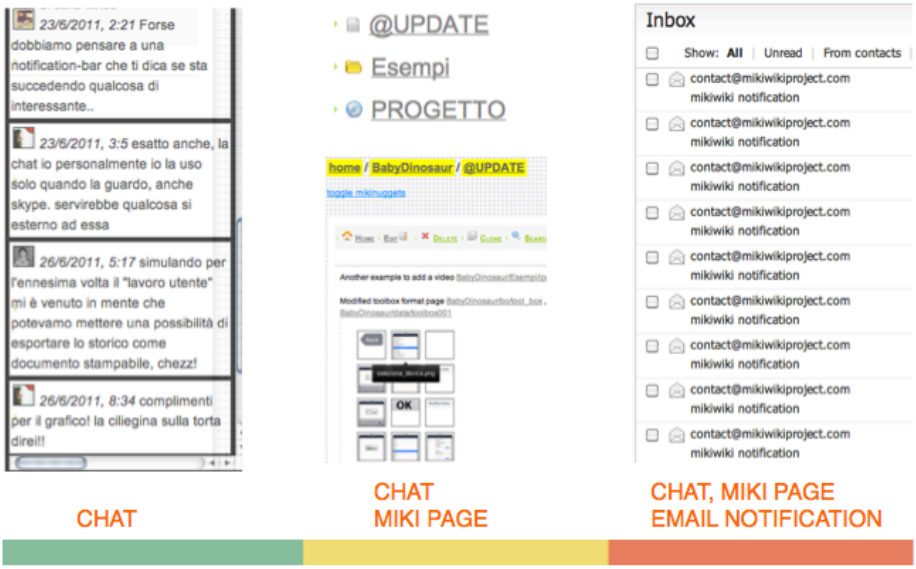


Figure 78 Evolving communication

Multiple channels of notification ensured that important information was disseminated on time and to everyone, according to their own style of communication.

Evolving Environments

The initial iPhone mockup design environment acted as a seed providing the designers with an opportunity to copy, tinker and learn (Figure 79, left image).

This environment provided only the very basic functionality to get started, but enough feature stubs to get the participants involved in completing and extending the functionality both on their own and with the help of meta-designers.

It did not take long to modify the initial canvas nugget's code, changing the backdrop, adding clickable elements, a simulation mode and the possibility of dragging images from any source. Following the same principles as for the iPhone, designers created their energy feedback mockup environment, a toolbox with design elements oriented to the energy domain, and an iPad *canvas* nugget. This allowed users to create interactive mockups prototyping some simple navigation and interface behaviors.

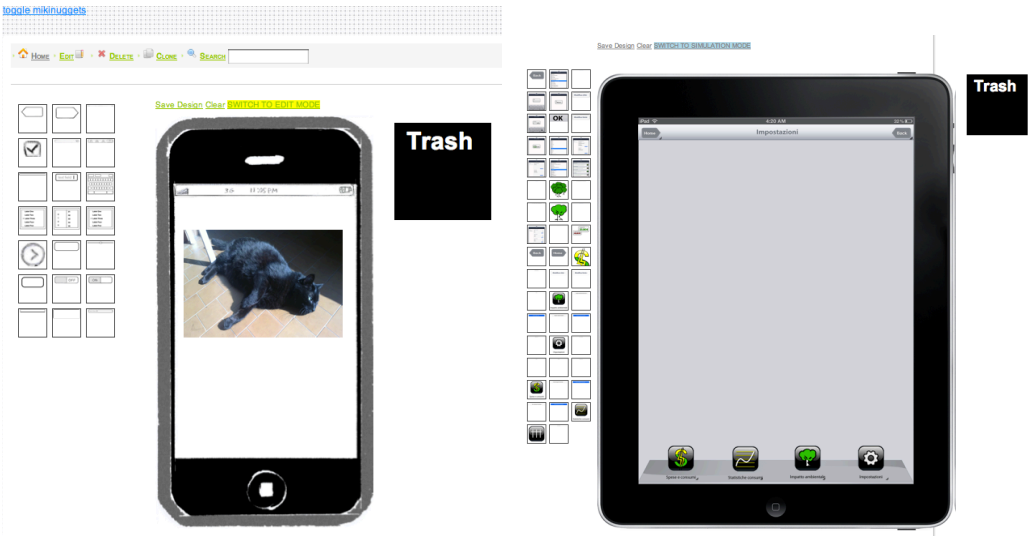


Figure 79 Evolving mockup design environment

7.5.6 Collaborative Design and Experience

All students noted that they were able to communicate, coordinate and share information with one another with the support of MikiWiki. They were all satisfied with their design results.

"La possibilità di poter "sviluppare" quasi tutte le idee che ci venivano in mente." [D1]

["I liked the possibility of developing almost all ideas we could think about."]

"*La flessibilità di MikiWiki è servita per comunicare tra di noi e soddisfare le nostre richieste che esigeva dal punto di vista tecnico, la nostra applicazione.*" [D3]

["The flexibility of MikiWiki was useful to communicate between us as well as to satisfy the technical requests that were needed for our application."]

Students gave positive feedback on being designers, despite which normally required additional workload and efforts, and valued the importance of the design experiences/process.

"*Sia il risultato che l'esperienza [sono importanti], non avevo mai avuto modo di lavorare su un wiki e soprattutto di mettere mano a un codice che mi facesse realmente vedere a video quello che stavo sviluppando*" [D4]

["Both the results and the experience [were important], as I never had a chance to use a wiki and especially I never worked on code that allowed me to really see [the results immediately] on screen of what I was developing."]

Moreover, students were able to make situated changes in MikiWiki to reconfigure the communication tools (Section 7.5.4). They used different nuggets to communicate (Section 7.5.1) and coordinate with one another.

An overview of the main application interfaces can be seen in Figure 80. The application aims to provide different visualization possibilities, namely in statistics, monetary value, and

environmental impact. The preference settings - respectively language, location, device, sharing and comparing with friends, defining energy consumption limits and goal setting - all oriented to offer a more personally meaningful energy feedback application.

Energy consumption home page

Real-time energy statistics feedback

Environmental impacts

Energy consumption expenses information

Energy service providers

Devices' energy consumption

Application instructions

Preference setting (Language, location, device, friends' comparison, energy consumption limits/goal setting)

Comparing energy consumption with friends

Figure 80 Application interfaces

7.5.7 Critical Issues

Learning Curve

The main issue in this project is that to fully exploit MikiWiki, one needs to go through a learning phase, since a general view of the paradigm of content, data and format pages must be explained and demonstrated by showing how to embed and tweak nuggets.

"*Fin da subito non risulta del tutto intuitivo ma col tempo si apprende molto velocemente*." [D2]

["Initially it is not very intuitive, then you can learn it very quickly."]

Once students understood the underlying logic they started tinkering with sample code and reconfiguring nuggets and the environment to create different visualizations. On the other hand, the meta-designers' support was crucial during the design process.

"*Con le dovute informazioni è stato molto semplice da utilizzare*" [D4]

["Once we received the necessary information, it was very easy to use"]

Users also expressed the desire to have had more time to grow familiar with the system.

"*Purtroppo il risultato, la fase progettuale è+ stata viziata da inesperienza e tempi molto stretti di consegna, dilatandola si sarebbero approfonditi molti aspetti interessanti*" [D1]

["Unfortunately for the result was the most important issue, the project phase was influenced by our inexperience and by a very tight deadline. Having more time would have allowed us to go deeper into many interesting aspects."]

Tailoring Activities

Despite the fact that MikiWiki has been improved over time, the main influential tailoring activities, i.e. extension (Section 5.5.1), were carried out by meta-designers based on the requirements from designers. One of the meta-designers was able to program by nuggets examples, e.g. creating a iPad canvas and a toolbox nugget based on a set of iPhone examples. One designer was able to understand and experiment with code, but to fully make use of MikiWiki still requires certain scripting knowledge and learning/exploring time. Students' main tailoring activities were customization.

Users' learning relied heavily on the tutoring by the meta-designers due to tight time schedule and inexperience [D1, D2, D4]. This reveals certain usability issues to be improved, e.g. making use of better interface cues and appropriate examples, and templates to ease the learning curve and scripting difficulties.

Notably, emotional support and immediate feedback played an important role in engaging students. It was observed that immediate support and encouragement from meta-designers motivated students to spend time in design activities and exploring MikiWiki.

7.6 Conclusions

In this specific prototyping of a UI mockup system, MikiWiki differentiates itself from other mockup applications such as MockingBird (MockingBird 2009), Balsamiq (Balsamiq 2008), UIMS (User Interface Management Systems) (Olsen et al. 1984), and GUI builders or construction kits (Fischer and Lemke 1988), as it not only supports rapid prototyping, but also

supports social interaction. MikiWiki as a generic platform can be adapted to other application domains (Section 5.7) and evolved over time.

MikiWiki in this design case demonstrates that it supports collaborative prototyping of an energy feedback system (Section 7.5.6), but also supports as an open, underdesigned meta-design environment (Section 7.5.4 and Section 7.5.5). It can be constantly improved during the design process. Nuggets encourage appropriation, and creativity in use (Section 7.5.2). Participants are able to switch between different roles (Section 7.5.3). All these observations confirm the feasibility of the HMS model and its potential benefits.

One of the most important observations is the role dynamic. As such, different phases of design (meta-design, design and use) represent different modes rather than discrete levels. With respect to the communication channel, it remains as an abstract concept rather than tangible software elements. It can be seen as the process of exchanging and sharing design results, co-learning and thus reaching mutual understanding.

Chapter 8

This chapter presents a design study in which MikiWiki was used jointly by the Universita' degli studi di Milano and Universität Innsbruck for designing experimental environments. These experiments are part of the Aristotele project (ARISTOTELE 2009) and the results will be analysed and used for implementing software recommendation systems. In this chapter I will present how we collaboratively evolved MikiWiki according to the project requirements on both meta-design and design levels as well as how designers appropriated different aspects of MikiWiki to support experimental activities. The design process became very creative over time as mutual understanding increased and users grew more familiar with the strengths and limitations of MikiWiki. Reflections on the design studies will be discussed at the end of this chapter.

8. Design Case Study-2: Aristotele Project

I used MikiWiki together with researchers of the Universita' degli studi di Milano and Universität Innsbruck to prototype a recommendation system and design online experimental environments. These experimental environments were then used by students from both the Department of Information Systems at Universität Innsbruck and the Department of Computer Science at Università degli Studi di Milano for collaborative writing and collaborative modelling.

8.1 Context and Goal of the Design Study

Aristotele is a project designed to study personalised learning and collaborative working environments fostering social creativity and innovation inside organisations. As part of the experiments related to the Aristotele project, this study aimed to test working hypotheses that additional Recommended Information (RI) given to a group improves work performance, quality and creativity.

Two scenarios were given for this task:

1) A company aims to create a new service, the Virtual Private Train Agent. A team of three people is appointed to write a comprehensive documentation for the services that come along with the Virtual Private Train agent.

2) At the McDuck Bank in Manhattan the average time to accept or reject a credit request is one week and sometimes even up to two weeks. Clients often complain about the duration. However, no one in the bank is able to inform the client in which department and phase the credit request is at a given time. The task is therefore to discuss a possible structure (architecture) for such a service with a strong focus on security provisions.

Within the company's collaboration tool, prototyped using MikiWiki, participants have the possibility to synchronously write the service documentation, communicate with others using chat, and browse the knowledge base relevant to this task. The knowledge base is a collection of documents, videos, websites and other resources that are shared within the organization. Additionally, the system provides a tag cloud which enables a quick search for relevant items. Some relevant items have been collected in the knowledge base.

The goal of this design study was not so much on the experiments but on how the designers used MikiWiki to design and evolve the experimental environments as well as how MikiWiki complies with the HMS model characteristics.

8.2 Project Timeline

We worked with the designers in the course of three distinct phases, as seen in Figure 81.

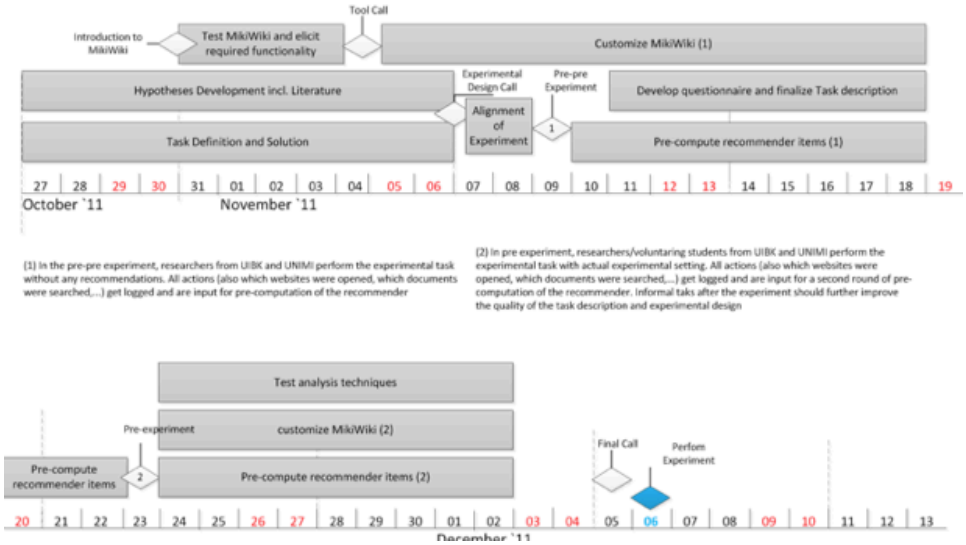


Figure 81 Timeline

During the first phase I introduced MikiWiki and its functionalities to the designers allowing them to define the tasks for the experiments and assess their feasibility within MikiWiki.

During the second phase the designers started adapting and evolving MikiWiki for the experiments, inquiring for information related to MikiWiki usage from the meta-designers, and

requiring further changes if they had reached the limits of what the existing nuggets provided. In the course of this phase the designers conducted a *pre-pre-experiment* on 9th November 2011 to assess issues of scalability and integration with external tools (e.g. Google Docs, YouTube videos, external websites and so on).

The third phase started after the *pre-experiment* on 23rd December 2011, where the final experimental environments were tested. During this phase we carried out further refinement and evolution of the experimental setting with MikiWiki.

8.3 Participants and Design Tasks

In this study there are three types of users:

1) Meta-designers: Two meta-designers provided nuggets for the designers to set up the environments. Based on the requirements expressed by the designers, the meta-designers created and adjusted existing nuggets for designers' acitivites.

2) Designers: Four main designers, together with meta-designers, collaboratively set up and evolved environments to test their research hypothesis.

3) Users: Although everyone is a MikiWiki user, there are two main types of users:

> a) Testers - researchers who help to test experimental environments, perform simple navigation tasks, create files, identify problems related to usability and report difficulties as well as suggest improvements
>
> b) Students - who use experimental environments to collaboratively write the business process model (BPM).

In this design study, meta-designers and designers collaboratively designed experimental environments. It was a co-learning and creation process in which meta-designers and designers reached a shared common understanding. In this process, meta-designers guided designers through using MikiWiki and tried to understand designers' needs, thus providing new nuggets for designers setting up their environments. Designers learned how to use MikiWiki, and evolved it for setting up their experimental environments.

8.4 Data Collection

I used open-ended questionnaires as a qualitative research method to find out what designers thought about MikiWiki, their opinions, their design experiences and the rationale behind their opinions. Given the small numbers of designers involved, open-ended questionnaires seemed more suitable, since we looked for feedback and qualitative understanding rather than quantitative validation (Dawson 2002). Four designers, two female and two male, aged from 25 to 40 years, and comprising PhD students and an associate professor, filled in questionnaires. They are all involved in collaboration, learning and decision support related

research. Two participants are specialized in web technology research, while the other two are specialized in information system research.

The open questions focus on:

- What kinds of design tasks designers do via the support of MikiWiki, to understand whether MikiWiki supports different design activities and levels of participation;
- The main differences between MikiWiki and other collaboration tools that participants have used, exploring the novel aspects of MikiWiki;
- What turns out to be possible that was expected to be difficult or impossible to understand whether MikiWiki is flexible enough to be appropriated accordingly;
- Whether and how MikiWiki supports their creativity.

Similar to the energy project discussed in Chapter 7, the usage of the environments by the users was observed and logged, log files were partially inspected, and based on this plus users' requirements an adaption of MikiWiki was conducted. Labels D1 to D4 identify the different designers.

8.5 Observations on HMS Aspects

This section presents some observations on how MikiWiki reflects aspects of the HMS model aspects and how these aspects supported successful completion of this design project. In addition, co-evolution and co-learning between meta-designers and designers will be briefly discussed, as they were another crucial aspect in the course of the experiment preparation.

8.5.1 Habitable Environments

The HMS *environment* concept plays an important role in this design study, since it allows designers to set different environments for different groups, i.e. groups with RI, groups without RI and environments that can be accessed by meta-designers and designers. It provides individual group privacy and it allows designers to collect group samplings more effectively. Designers can enforce certain social rules in the environments. Designers stated their intent to build environments with a predefined structure and linear workflow as well as to restrict any behaviours irrelevant to the collaborative writing tasks.

They set up 12 environments, named WorkSpace_G1 (WSG1) to WorkSpace_G12 (WSG12). Each environment, (or WSG), was used by three students during the experiments.

The 12 environments are further divided into two groups: WSG1-WSG6 was equipped with the prototyped recommendation system, while WSG7-WSG12 without one.

Figure 82 shows the 12 WSGs, one *TaskDescription* document and a "*sources*" folder, which are independent from individual WSGs, and therefore are shared and can be easily included in an individual WSG.

Students only had access to their own WSG. WSG1-WSG6 were allowed to browse through the "*sources*" folder contents, which can be used by students to support their collaborative writing tasks.

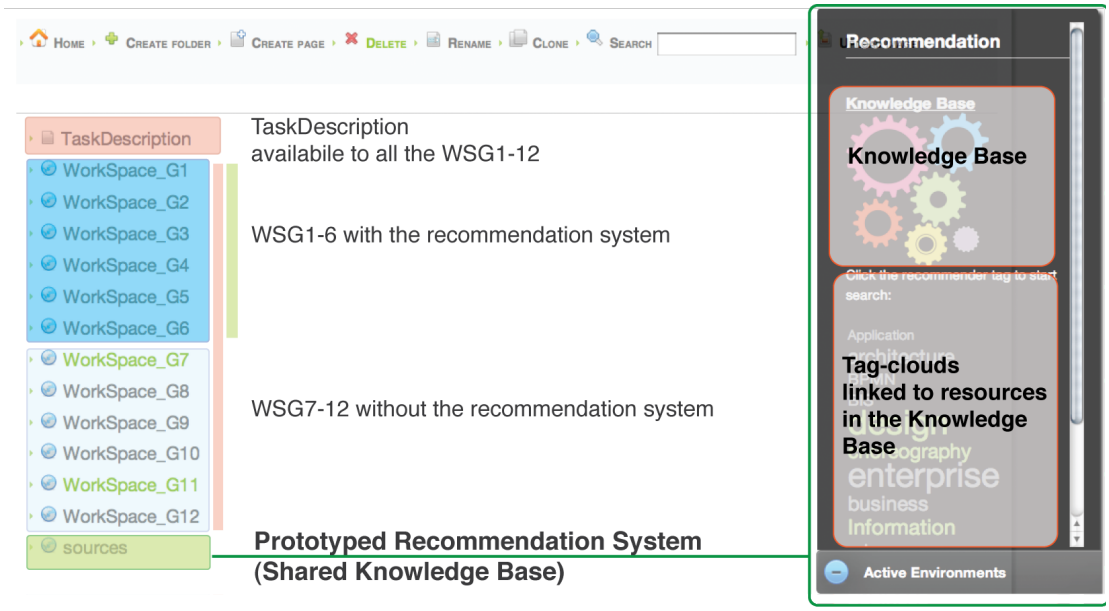


Figure 82 Environments overview in the pre-test phase

Creating and using MikiWiki Environments had the following benefits:

1) Environments support students in filtering out distracting information and put the focus on the task at hand (Figure 83).

2) MikiWiki provides the possibility to personalize environments. Designers could redesign the MikiWiki interface and add new elements, i.e. a shared knowledge base arranged at the top right corner of the MikiWiki page. Another simple example is that some environments can be equipped with a prototyped recommendation system - i.e. the tag cloud - while others are not. Environments are an essential mechanism in this design study to support designers in testing their working hypothesis. As one of the designers commented: *"with MikiWiki it is easy to personalize the environment composed of the exact set of services you want to be available in the environment"* [D1]

3) MikiWiki allows individual environments to share certain common behaviours - for instance, being able to access the shared knowledge base for WSG1-6 and being able to use shared nuggets, such as the "*expand*" nugget used to include "*TaskDescription.*"

4) By setting up environments in MikiWiki, meta-designers and designers can impose different rules (Section 8.5.6) on different environments. For instance, after students log into the system, they will be redirected to a specific WSG.

5) In MikiWiki it is easy to analyse each group's activities, communication and social interaction. For example, each WSG has its own chat data page, which records user's name,

timestamp and chat entries as well as an embedded GooglDoc for writing tasks that can be used for further analysis.

6) MikiWiki allows designers to set up test environments efficiently. For instance, designers specified hiding of certain menus and buttons by adding specific nuggets within the "*UIBK/environment*" page. All the sub-environments WSG1-12 inherit these characteristics (Figure 82).

7) Within each WSG folder, the "*environment*" MikiWiki page and the "*bo*" folder are technical conventions used to create localized properties and policies.

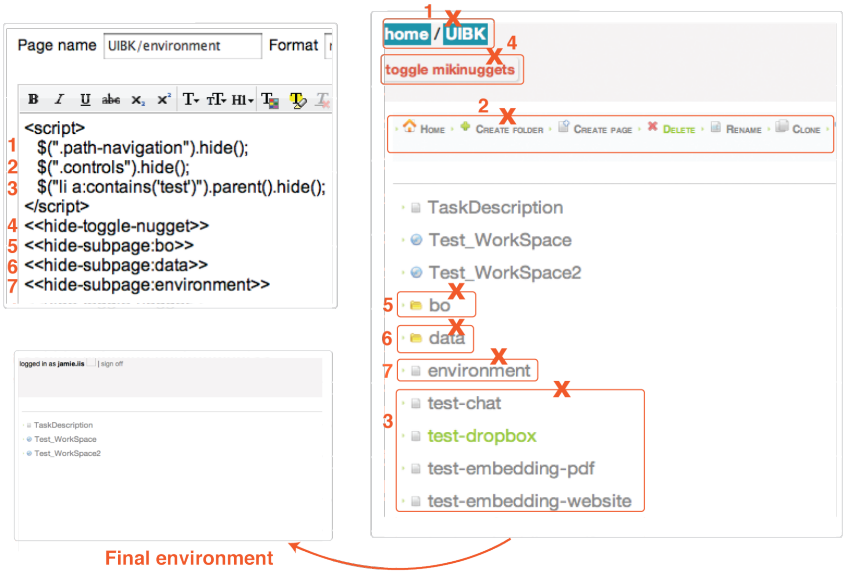


Figure 83 Evolving the experimental environments

Since designers wanted students to focus only on the writing tasks and on searching for digital resources related to their tasks, they aimed to eliminate all unnecessary navigation, information browsing and other actions, e.g. editing or deleting a MikiWiki page.

Figure 83 illustrates how designers inspected the HTML code generated by MikiWiki, identifying all the superfluous elements, utilizing nuggets and manipulating DOM elements to hide all the additional buttons, folders and MikiWiki pages presented in the environment. The final environment is shown at the left bottom corner.

In the energy feedback system study, habitable environments (Section 7.5.1) were used as personal learning places. However, in this design case they are indispensible, since the whole design logic of how to distributed resources, filtering information, impose social rules, as well as the prototyped recommender system were based on the environment mechanism.

8.5.2 Boundary Objects

Repurposing the Todo Nugget

The *todo* nugget allows users to create a simple task list. Each task entry can be moved up and down the list and deleted. Designers appropriated the *todo* nugget as a Q&A tool, using it to post questions and answers. Interestingly, their appropriation included establishing the social protocol of signing every entry with an acronym, to indicate who posted it (Figure 84).

This suggests the importance of creating a sense of ownership in the collaboration process. If the same task had been undertaken as a meta-design task, modifying code to add more functionality, it would have taken much more time and resources. However, a successful appropriation can be taken as a working example of a social practice and later reified into an automated mechanism via meta-design.

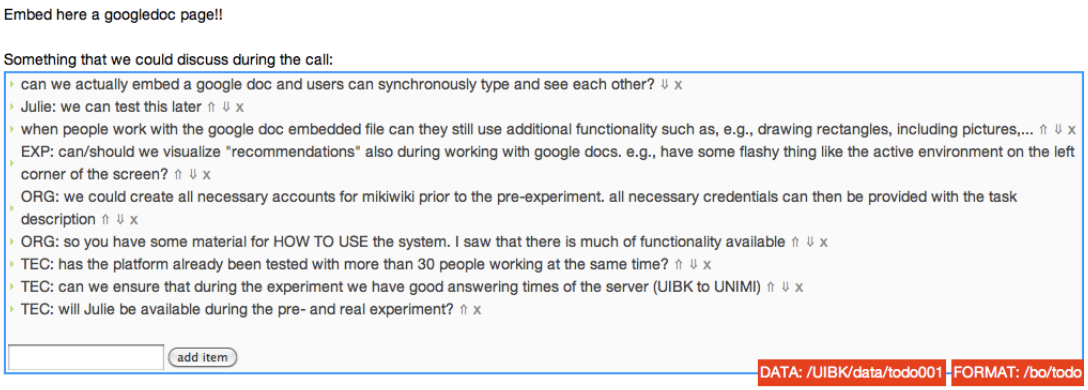


Figure 84 Todo nugget as a Q&A board

Bricolage - ShowTag Nugget

Rather than engineering new nuggets from scratch, MikiWiki provides the possibility of taking the existent nuggets and performing minimum adaptations to change them or repurpose them to the task at hand. A typical example is the *showTag* nugget. This nugget is the combination of the *tooltip* nugget that was previously created and the existing *folder* nugget, which is used to provide an overview of folder sub-pages and also detailing meta-data information, such as creators, creation time, format of a page, and so on.

For this specific use case, the *folder* nugget was evolved to display the tags associated with a sub-page. Figure 85 lists a set of documents and associated tags.

Figure 85 Uploaded files with associated tags

Meta-designers did not code a new nugget from scratch nor create a very sustainable solution; rather they opportunistically merged two nuggets' code to rapidly solve this specific problem. Reusing, combining, tinkering with and evolving nuggets also suggests creativity on the part of meta-designers and demonstrates how meta-designers coped with the designers' emerging requirements via bricolage and opportunistic programming. *"IMHO it supports reuse and this can drive creativity in allowing us to mix existing solutions."* [D1]

Building Blocks

Nuggets are small building blocks. In this design study, the experimental environments were solely made of a set of nuggets, respectively, an *expand* nugget for design tasks, a *chat* nugget for real time communication, a *googledoc* nugget for synchronous collaborative writing, a *KB* nugget combined with a *tooltip* nugget providing students an overview of the knowledge base as well as access to resources (information related to the BPM tasks), and an *include* nugget linking to tag clouds. Tag clouds are used as filtered information, organized by keywords (Section 8.5.3). Notably, some environments are without the *include* nugget in order to compare with the groups with it (to include tag clouds).

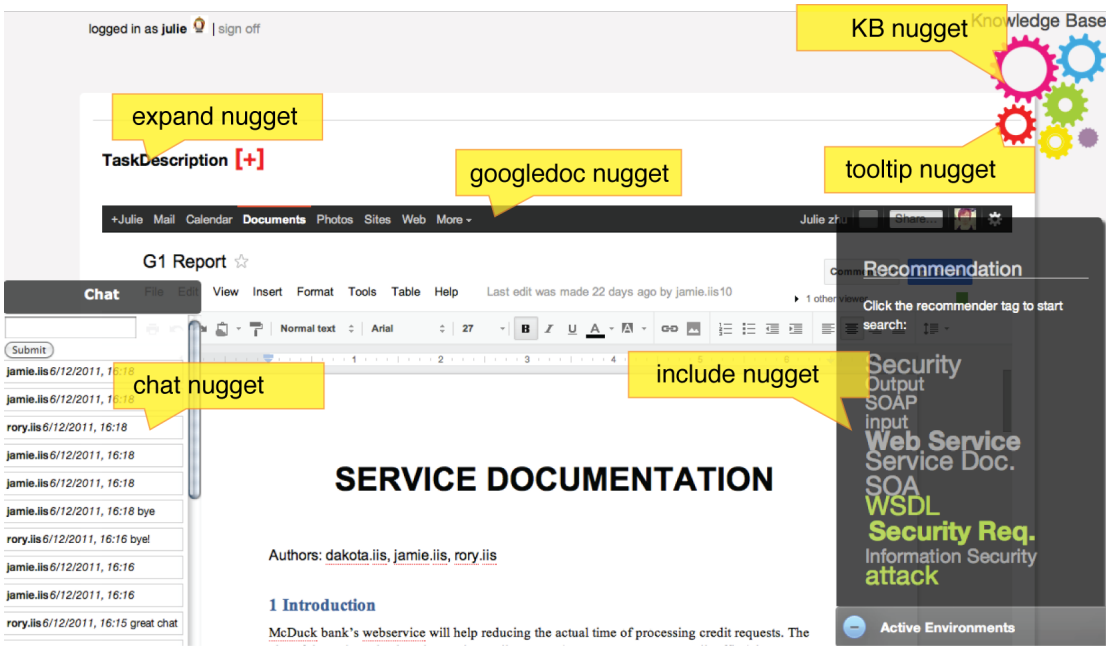


Figure 86 Environment building blocks

One designed commented on the flexibility and the situated integration of environments, *"MikiWiki gives great help in creativity since its flexible environment can be adapted to different scenarios, without limiting user work practice and overloading them with useless tools, or requesting them to fulfil additional tasks. Also, the possibility to include communication tools and remote co-authoring of documents givse great help"* [D4].

Some insights from setting up environments are that nuggets not only shape social interaction among design teams, but also together construct a set of "*design contexts*" for designers and users to further create their own contents. Each nugget has its own data page, and content.

To this end, nuggets together build infrastructures for designing opportunities. As nuggets are independent from each other, lightweight development and continuous adaptations become possible.

8.5.3 Open Infrastructure

Some important observations with respect to open infrastructure are that MikiWiki became (1) a medium to leverage different services and resources, and (2) an open system for appropriation and for inspection.

Leveraging Existing Services

Although MikiWiki supports synchronous communication and it already has several real-time collaboration nuggets, it would take considerable effort to develop a truly synchronous collaborative text editor with a minimal part of the functionality of existing online tools.

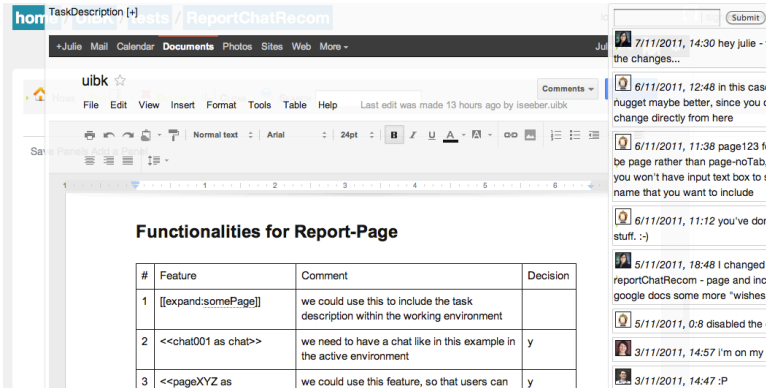


Figure 87 Embedding GoogleDocs for asynchronous and synchronous collaboration

Designers, however, came up with the idea of embedding Google Docs within MikiWiki, to complement the features missing in MikiWiki. The initial embedding efforts were then encapsulated into a template nugget for easier reuse Figure 87.

Apart from leveraging GoogleDocs, designers used MikiWiki as an asset to connect websites, YouTube videos, Dropbox and online documents to create a knowledge base for the BMP tasks. In this way, different services could be plugged into/unplugged from MikiWiki in situ to support problems solving. As one design pointed out: "[it offers] *the possibility to include other platform services that have already reached a special level of maturity. E.g. the possibility to integrate Google Docs or Dropbox – both services have good scalability and performance…" [D3]*

Pre-computed Recommender (Manually)

The main goal for the designers was to test whether a recommendation system reduces information overloading. MikiWiki was used to prototype the recommendation system, which should filter source documents based on assigned tags, offering recommendations based on the task that a user is performing and ordered by relevance.

Rather than implementing the full system, meta-designers and designers decided to produce an interactive mockup, manually writing the HTML code and including it in the sidebar. Figure 88 shows the "*Security*" tag in the HTML code, with its visualization in the tag cloud page and representation in the sidebar. The code *<a href="/search?keyword=security">* indicates that when a user clicks on the security link, a *security* tag search will trigger within MikiWiki and retrieve all the pages tagged with the keyword *security.*

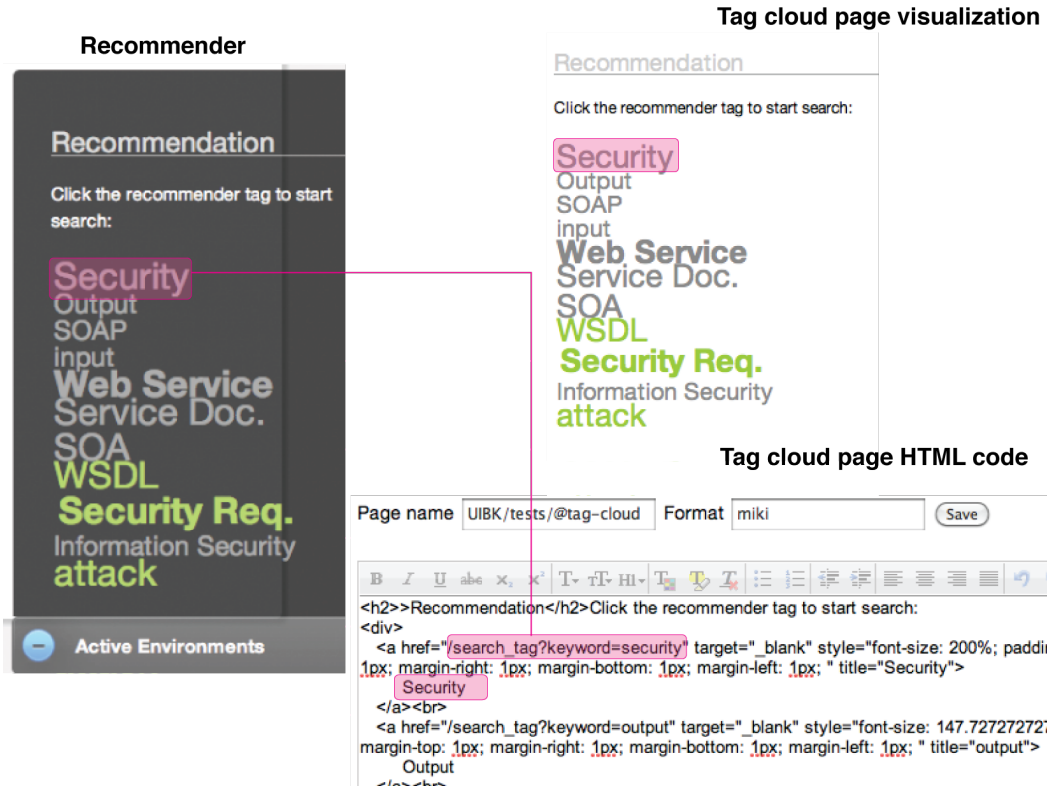


Figure 88 Appropriating a tag cloud

Styling the Document Icons

Figure 89 shows the initial knowledge base representation (on the left). Each document was associated with a simple document icon. However, this representation did not provide sufficient cues for students to search for information.

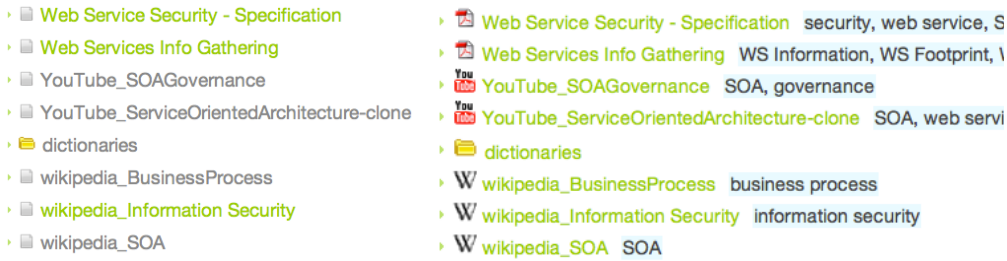


Figure 89 Evolving the knowledge base

One design requirement that emerged from the pre-test is the representation of the Knowledge Base *"Is it possible to have another presentation of available articles - That*

besides the display of the article you also have further info: associated tags (for groups with recommendations) and type of resource (manual, scientific paper, Wikipedia, YouTube)?"

For the purpose of this project and to solve this problem in rapidly, meta-designers created an identical page and linked an icon image, combining a *ShowTags* nugget (Section 8.5.2) to display the tags associated with each document. "*"I really liked the creation of the knowledge base – more or less the fake representation of a document collection.*" [D3]

While this is not a solution that can be generalized to other situations, it solves the immediate problem and could become a clear and concrete design requirement for further meta-design activity. These two examples illustrate that the openness of MikiWiki allows meta-designers and designers to collaboratively explore, appropriate it and come up with creative solutions.

Open Code and Data

To support designers to analyze students' interaction, the log data was itself available as a data page and it therefore could be read, filtered and reasoned by designers within MikiWiki itself. The log file records information for each interaction in terms of timestamp, user name, environment name, page name and its format, page relative/full path, search tag request, keyword etc. "*MikiWiki offers the end user logging functionalities that some commercial tools don't provide."* [D2]

MikiWiki makes data and code accessible to users as well as how it handles data in real time. By doing so, it provides a more open and transparent system, thus empowering users to modify the system with different granularity. *"A rather technology-oriented perspective is given – as a user you always have the possibility to change basic settings, i.e. active environments, inclusion of various nuggets, hide/unhide buttons – normally I'm used to collaborating from a pure user-perspective, e.g. Google Docs does not allow me to hide buttons or in Microsoft SharePoint the separation between different user groups (admin, user, moderator,….) is much more extended." [D3]*

On the other hand, it suggests that the code and data need to be further opened and made accessible to the users as a more radical approach to encourage cultures of participation.

8.5.4 The Boundary Zone

The boundary zone is made clearer in this design study. It is not a software artifact, but a learning, creation and evolutionary process that encourages creativity.

Learning Process

Observations from this study suggest that the collaboration process supported by MikiWiki is a continuous co-learning process. *"At first I could not imagine how we could represent MikiWiki-page names together with created tags."* [D3] *"In the end the search inside documents was a big help. I did not find this functionality when I first used MikiWiki"* [D2]

It is important from a meta-designer's perspective to support designers' tinkering and exploring of MikiWiki by providing examples as seeds, rather than going through long explanations or completing everything for them. Revealing complexity gradually and according to the situated needs of designers supports them in stepping up over time. By engaging with seeds, designers can reflect on their actions and design results, in turn creating new ways to use the systems. In this context, seed nuggets act as boundary objects to encourage mutual understanding and system evolution. A partially unfinished and underdesigned nugget invites users to explore it and complete it for their needs.

During the collaboration process, designers tended to explore MikiWiki by themselves. If they had any problems, they asked meta-designers to show them examples and teach them "*tricks*" and "*know-how*" to achieve their goals rather than waiting for meta-designers to solve their problems. Designers wanted to be independent and to be in charge of their own problems, and were willing to be involved in the meta-design process and keen to understand how the system worked. As one designer stated, *"I think that MikiWiki supported the development of my technical understanding."* [D3] Practice oriented understanding (Orlikowski 2007) of the recursive interaction between meta-designers and designers in the social context can better explain design-in-use and situated creativity in action.

With respect to the experiment itself, environments and nuggets were deliberately evolved and designed to support this process. This changed the nature of supporting experiments from primarily problem solving to both problem solving and system evolution.

Evolving Features and Requirements

Working on this case study, I had practical confirmation of how hard it is to predict in advance the evolution of design features and requirements. Moreover, temporary solutions would trigger new issues for the next iteration, and reveal some existing yet hidden issues, which were previously obscured by interaction problems with higher priority. As social and human factors of new features constantly drive new requirements, rather than trying to detect all issues in advance, we can rely on a set of flexible resources and practices to evolve the system as part of a continuous reflective and iterative process.

Designers' creative use of MikiWiki provided several opportunities for meta-design. Meta-design further enables new design and appropriation opportunities, and consequently the designers can improve their design result, which supports users creativity recursively. Users seed users from other communities and levels of participation and evolve MikiWiki during design-in-use.

The evolving process emerged through social interaction over time along two dimensions: (1) technical improvements and (2) enhanced mutual understanding. It is a co-evolution process driven by communication, co-learning and co-exploration between designers and the meta-designer.

Nevertheless several technical facts of MikiWiki made the boundary zone observable:

(1) A unique ID for each entity that users can refer to, track and discuss.

(2) Open accessible and shared documents through which users are able to open up, inspect and extend the system.

(3) Accommodating multiple perspectives through a mediation mechanism, which is done through utilizing "environments" lookup mechanism to overwrite initial settings.

All these concrete technical means together ensure the learning, exploration and co-evolution possibilities.

8.5.5 Levels of Participation

The three levels of design follow SSW methodology (Costabile et al. 2006). Figure 90 shows the three levels of participation - respectively, meta-design, design and use levels - in the Aristotele project (see Chapter 9).

At the meta-design level: meta-designers guided designers in using MikiWiki, providing examples as seeds (Fischer et al. 1994) and different nuggets for designers to design experiment environments. Based on the requirements expressed by the designers and end users as well as observation of their appropriations and recurring usage patterns over time, meta-designers evolved the MikiWiki infrastructure, using existing nuggets or creating new ones for supporting designers' activities.

At the design level: designers created workspaces for different groups, prepared the knowledge base (which was made of a set of documents related to business process modeling), created a tag cloud, embedded documents, tested MikiWiki and provided feedback [D1, D2, D3, D4]. Meta-designers were also involved in this process and collaborated closely with designers, reasoning upon and designing experimental environments. This was mainly done asynchronously. *"Collaboratively, designers used MikiWiki to remotely communicate with the other working groups, to cooperatively write the deliverable [preparing BMP tasks and requirements for the experiment], and to create as well as to share a knowledge base"* [D4].

The creation process was iterative and incremental along with the development of users' mutual understanding and designers' increasing knowledge of using MikiWiki. After the experiments, designers used different chat entries and log files to analyze end users' performance, interaction and communication during the experiments. Being able to access data was certainly very important for designers to reflect on their design tasks and environments setting, and to rapidly try out various scenarios.

At the use level: students as end-users helped to test experimental environments and used experimental environments created by designers to collaboratively write the business process model. This was done synchronously.

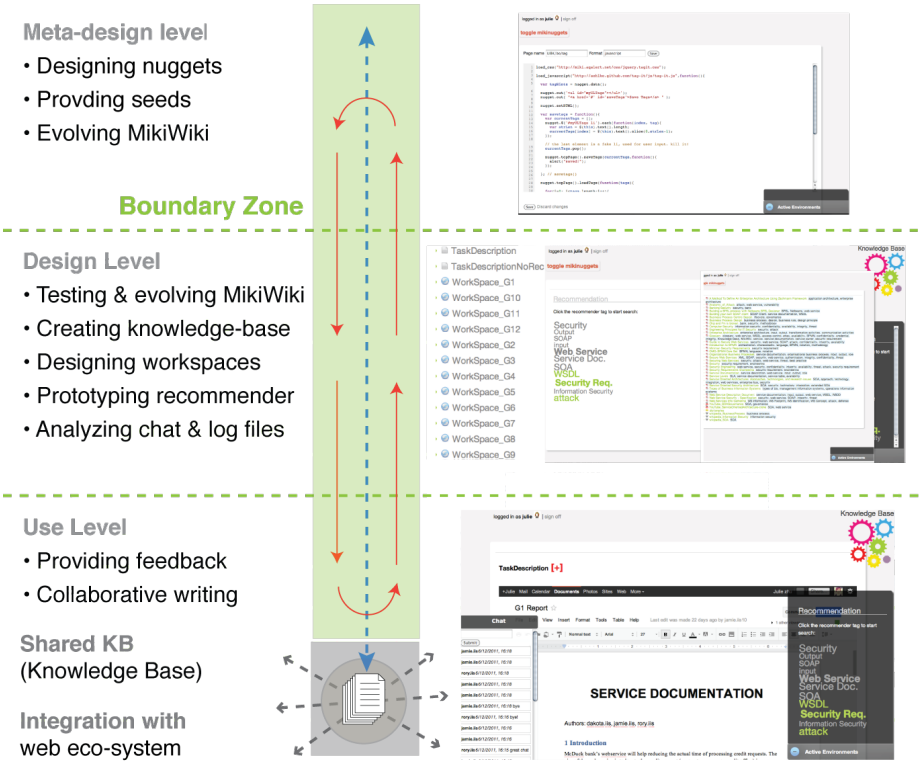


Figure 90 Collaborative prototyping of recommendation system

Different designers have been involved in different tasks. All of them have been intensely involved in learning, using and evolving MikiWiki to design the environments for the final users.

8.5.6 SER Model

Incremental Implementation - Meta-designing Nuggets

During the collaborative design process, the initial nuggets were extended and new nuggets were created according to specific requirements of this project and to better support students' design activities. The following example, the *chat* nugget, focuses on refining initial nuggets according to emergent requirements.

The design issue here was that designers wanted to provide the maximum space for students to write on, while leaving the main working area for embedding the GoogleDocs. The initial chat nugget did not fit in this context for two main reasons:

1) The initial *chat* had to be embedded within the main body of the page, sharing the working area with the Google Doc. When many chat entries accumulated, little space remains for the GoogleDoc.

2) In their pre-pre test, all the communication was done intensively via chat as students were geographically distributed. Each new chat entry however was posted from the bottom input text box. When the chat entries became very long, students always had to scroll down to the

last chat entry at the bottom to add new entries, which made it unusable in a situation requiring efficiency.

The chat nugget is a floating box (Figure 86 in Section 8.5.2), which users can resize according to the document length and position to be able to communicate with other students while working on the GoogleDoc during the experiment. The order of the chat messages and the position of the input box were also reversed, so that the latest exchanges are always visible at the top of the nugget. Some detailed modifications were requested by designers for improving its functionality over time, e.g. timestamp format, refresh rate, the chat entry array length and so on. Many other nuggets were extended and created for creating the knowledge base or for tailoring environments. This evolving process is iterative and re-enforced by continuous emergent issues as well as designers' appropriation. It is through this process that meta-designers and designers enhanced their mutual understanding.

Transforming Usage Patterns to Templates

During the knowledge base creation phase, meta-designers observed that designers needed to include many sources and create tags for each source. First, designers need to create a MikiWiki page and use the *file* nugget to include a PDF file. Then they need to include the *tag* nugget to assign tags to this specific MikiWiki page.

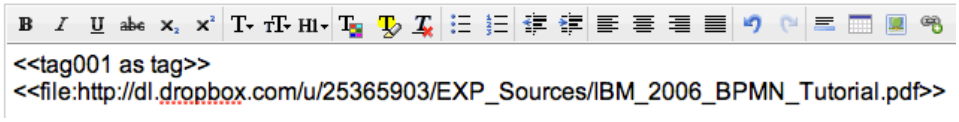


Figure 91 Syntax for creating the UIBKpdf template

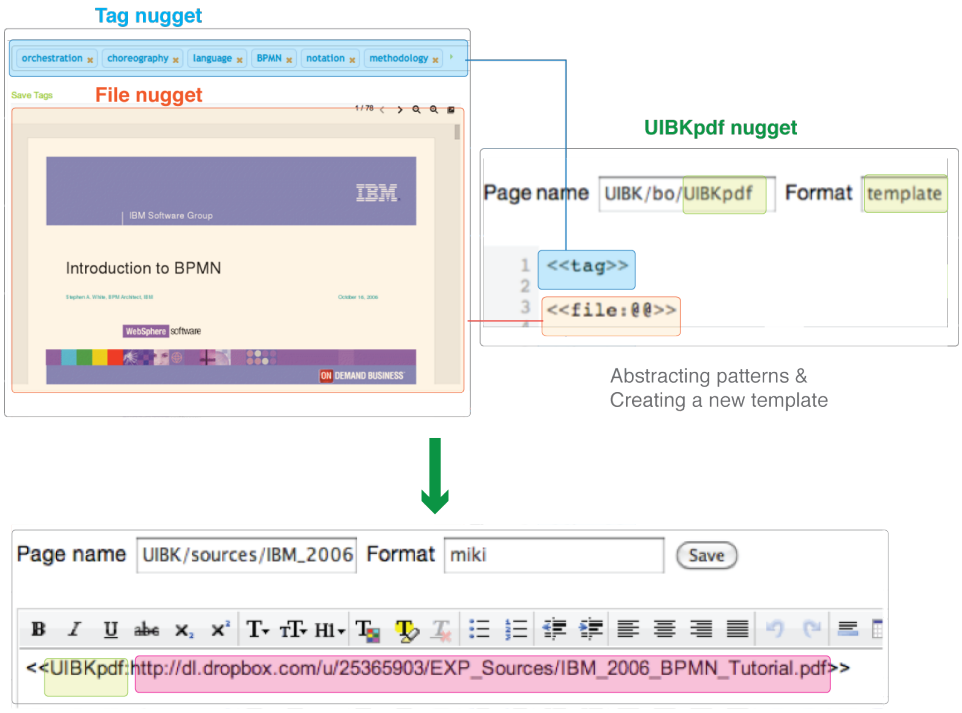


Figure 92 UIBKpdf nugget

For example the typical syntax to include the IBM_BPMN PDF file can be seen in Figure 91. This work had to be repeated for every source. The meta-designers observed this pattern and abstracted it by creating a new nugget UIBKpdf, which combines the *file* nugget and the *tag* nugget (Figure 92). UIBKpdf allows designers to combine the *tag* nugget with the *file* nugget into a single nugget. Combining two operations into one reduced both the designers' workload and the possibility of making mistakes.

In the same way, the meta-designers created two further templates for combining the *tag and video* nuggets as well as combining the *tag and website* nuggets (Figure 93).

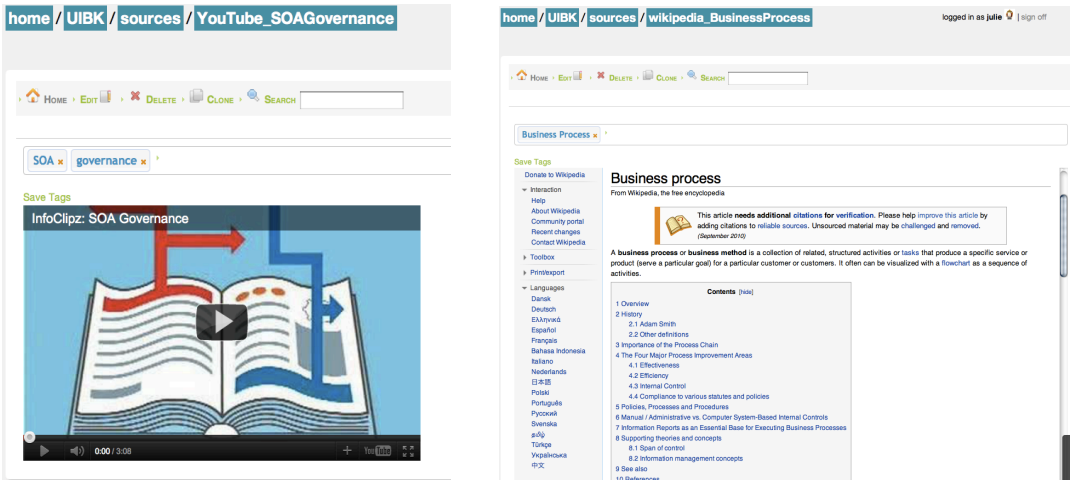


Figure 93 UIBKvideo and UIBKwebsite nuggets

The possibility of observing and formalizing emergent usage patterns (Section 5.4) underlines the openness and underdesign principles of the HMS model.

Imposing Social Navigation Rules

A distinctive HMS model design principle is that design communities create and follow their own local rules. The social rules can be either imposed or emergent as in the previous example of creating templates. In this study, designers have been restructured the social process for their testing purposes step by step. This process was exploratory rather than being clearly defined in the very beginning based on observations and feedback from pre-tests and increased understanding of what MikiWiki could do. Several nuggets were created to support designers' requirements.

The *landing-page* nugget was created based on designers' requirements. It directs students into the WSG that they belong to once they log onto MikiWiki, not exposing them to any information outside their WSG.

In the final experiment todo list, one of the requirements is to hide the *bo* folder, the *data* folder and MikiWiki pages unrelated to the writing task within each WSG, since designers wanted to limit any possibilities that might distract students from the collaborative writing task.

The *hide-subpage* nugget offers the possibility to hide any folders and their associated sub-pages.

This again confirms that the openness and the underdesign characteristics of MikiWiki empower advanced users to construct and tailor communication, impose social rules, and shape other users' behaviours according to designers' needs. As one of the students commented: "*MikiWiki didn't allow me to explore additional info than that provided there,*" while this was a concise design requirement designers wanted to achieve.

8.5.7 Critical Issues

Improved Learning Experience

With respect to the learning curve issue (Section 7.5.7), I created a thorough text-based documentation on how to use nuggets, environments and API, which has proved to be very effective. *"The quite flexible integration of tools (nuggets) and therefore the use of tools are known and don't require special introduction."* [D2] Several users suggested a multimedia tutorial to provide a more interactive and engaging learning experience, which could be introduced in the future.

Additionally, in the beginning I did not give designers a long and thorough introduction instead I tried to reveal complexity during the design process, when it was necessary. Information is unfolded step-by-step so that the user is guided through a complex task. MikiWiki is a space for designers to build things, and therefore it is important for meta-designers to understand what designers want to achieve first and introduce functionality that is relevant to them rather than introducing unnecessary complexity. When designers asked questions about how to achieve certain results or complete certain tasks in MikiWiki, I tried to provide some examples as "seeds" and explain the rationale behind them rather than solve problems directly. For instance, I created an environment to demonstrate how to hide/unhide certain elements (e.g. menu bar, folders), and how to manipulate pages (e.g. basic CRUD operations), when the menu bar is hidden, and how to access invisible folders via URL. In this way, the designers were enabled to cope with similar problems by themselves.

Openness vs. Blank Page Syndrome

When the designers started using the system, it was not always clear to them what the next obvious step should be. The system is very open, and thus at times this very openness encourages ad-hoc activities. Yet this could leave a user at a loss about what to do next. Thus, we still needed researchers to design the initial environment before co-evolution could take place.

MikiWiki was proven to be very effective for tinkering. It is easy to access something that works, clone it and tweak it into something new that still works. Conversely, it is much harder to come up with something new from scratch. To encourage cultures of participation, the

initial environment should be carefully planned, offering rewards and incentives, fostering public commitment, establishing clear contribution norms, and providing clear mechanisms for conflict solving (Grudin and Poole 2010).

Appropriate seeding can be the middle ground between openness and formal structure, hinting loosely at potential directions, allowing users to choose where to go, and supporting them along the road they decide to take.

Synchronous Communication

The main difficulties were related to the performance of the synchronous communication services. Since many experiments were run simultaneously, the system had to handle the cross-communication requirements of about 30 students, far beyond what the basic synchronous communication service was designed for.

For example, the chat used in our environment was not able to refresh immediately and this delayed collaboration among users, in some cases creating misunderstanding or inconsistent actions [D1]. *"The chat nugget did not function very well, so that the information flow between people was limited – if this could have been better – maybe we would have developed our ideas further…"* [D3]

The refresh rate of the chat blocked information flow. This problem was amplified due to the nature of the project, especially under time pressure; students had to remotely and collaboratively accomplish a complex writing task in two to three hours. Robust real time communication support, possibly relying on proven Jabber/XMPP technologies (Miller 1999) should be provided in the future.

8.5.8 Collaborative Experiences

MikiWiki could support collaboration and creativity by empowering designers with the means to achieve their goals rapidly, by allowing different levels of participation as well as to accommodate a different range of expertise and ambition.

"It's not a huge system with thousands of features that you only understand when you read a book of 1000 pages – it's small, easy to adapt to my purposes – at the same time however, it's a question of you want to do with the system. For our purposes it was enough – however if you want to use it for broader purposes, search & retrieval of information, DMS, user administration and roles, versioning etc – it might not fulfil your purposes" [D3]. Designers achieved their research objectives but also pointed out the "underdesign" characteristics, as MikiWiki is not as comprehensive and robust as other commercial products. The final hypothesis analysis from students' collaborative writing is out of my research scope. However, much greater detail on this topic can be found in (Mariani 2012).

With respect to creativity support, designers perceived MikiWiki as being associated with solution reuse, collaborative design, synchronous activity, flexibility, useful tools, not limiting work practices, adaptable environments and communication tools [D1, D2, D3, D4].

All the designers enjoyed the availability of various ready-made small tools and the possibility of recombining and modifying them rapidly – often in a matter of minutes. *"The main difference* [between MikiWiki and other collaboration tools] *is the modularity and flexibility of the system, which could be adapted with a limited effort to different usage scenarios. Also the choice of embedded objects as simple tags is a competitive advantage."* [D4] *"I really liked the creation of the knowledge base – more or less the fake representation of a document collection. […and] the performance/scalability of the system in general."* [D3] When the designer refers to "performance and scalability," she refers to openness and flexibility, where everything potentially can change without having to switch to a different system – e.g. programming on a server, compiling, and so on.

Since the designers were researchers, they were also glad to have access to extensive log files. They were surprised to find the real-time logs accessible as yet another MikiWiki page, without having to resort to different systems to retrieve the data.

Moreover, designers appreciated that they could include other familiar systems into MikiWiki, embedding GoogleDocs for collaborative writing in real time, and to linking different media resources, such as documents collected in DropBox, YouTube videos, websites and so on (Section 8.5.6).

8.6 Conclusions

This chapter described a design study in which MikiWiki was used to support researchers in designing experimental environments as part of the Aristotele project. I described how we evolved MikiWiki from the initial requirements working on both meta-design and design levels together with the experimenters. Observations of the work practices and feedback from questionnaires showed how MikiWiki complies with the HMS model aspects that support collaborative practices over time. An important reflection from this design study is that the boundary zone is revealed as learning, exploration and evolutionary process rather than a concrete software artifact. On the other hand, this process can be supported by from a technical perspective, e.g. open underdesigned system, mediation mechanism.

In summary: The design studies presented in this thesis, applying MikiWiki to different application domains, depend on the projects and users available at the time. In addition to these two design studies (Chapters 7, 8), MikiWiki was also used for other design studies not extensively explained in this thesis:

1) The Tarquinia workshop for the International Etruscan Sigla archeology information system project, where MikiWiki provided tools to annotate the system interfaces.

2) A role-playing game virtual tabletop environment design.

3) A real-time collaborative story creation environment using a comic book interface inspired by role-playing games.

Even when MikiWiki was not the ideal platform to start with, MikiWiki proved to be generic enough to be easily adapted to different domains.

The strengths and limitations of the MikiWiki are as follows:

Strength: (1) the flexibility of applying MikiWiki into the different use cases demonstrates meta-design and end-user development concepts related to how it can be used, shaped and evolved by users at use time; (2) Empirical observations of the HMS aspects support collaborative design and encourage appropriation or bricolage of existing solutions; (3) Meta-design can be hard to promote but emotionally rewarding as demonstrated by the engagement of designers when they could see code in real time in the energy feedback system project. In addition, designers were proud of being able to use MikiWiki from a somewhat technical perspective.

Limitation: (1) scripting is more difficult than customization and integration. Most participants are not willing to put the time and effort into meta-design activities unless they perceive their efforts could improve personal skills or achieve design goals; (2) Design studies were conducted within two or three months. They normally involved in a small group of users with a very tight time schedule. Some empirical results and use behavior might change over a longer period. Ideal scenarios, e.g. a richer ecological migration, transforming passive users to active producers could not be evaluated. They are also limited by users' experience.

Chapter 9

This chapter presents how MikiWiki has been used and evaluated in the collaborative design of the *Creativity Barometer* (Herrmann et al. 2011) for mobile devices. Designers and users were involved in designing a mobile version of the creativity barometer, while the meta-designer was involved in setting up and evolving the design environment as well as facilitating the collaborative design process. MikiWiki was used for the first time in a co-located setting, rather than remotely, allowing several people to see MikiWiki on a large screen, but only one to interact at any one time.

Different levels of design activity and five design iterations are analyzed. The results of the evaluation show how MikiWiki, as a prototype of the HMS model, supports collaborative design among different communities and fosters their creativity. The evaluation results are interesting as they deliver detailed insights into enabling creative and collaborative design with a socio-technical approach from the perspective of meta-design and the roles of meta-designers. Reflections on how the whole design and meta-design cycle can be improved are discussed at the end of this chapter.

9. Evaluation

The evaluation was done in collaboration with the Information and Technology Management, Group at the Institute of Applied Work Science, Ruhr-University of Bochum, Germany. Meta-designers, designers and users were tasked to collaboratively design an Android phone version of a micro-survey tool, the *Creativity Barometer* (Herrmann et al. 2011).

9.1 Creativity Barometer

The purpose of the *Creativity Barometer* is to conduct surveys to continuously understand and assess the climate of a company's creativity support. The *Creativity Barometer* allows companies to periodically repeat surveys and get instant feedback continuously. After a pre-specified time period (e.g. eight months), the company can summarize the feedback and plan interventions to improve the creativity climate. The *Creativity Barometer* was tested in four companies over several months (for instance, 99 employees produced 2673 answers in

September 2011). Since continuous surveying can disturb employees and the aim was to support them to provide their answers as “en passant” as possible, transferring the desktop-oriented browser-version to smart phones appeared reasonable (Herrmann et al. 2011; Zhu and Herrmann 2012a). Accordingly, 11 experienced users of the web browser version were available to be designers of the new mobile interface. These users had various backgrounds, and some of them had no experience in software development.

This design task – drafting the appropriate characteristics of the smart phone solution – was chosen to evaluate whether MikiWiki could support collaborative design and participants creativity.

9.2 Goals and Context of Experiments

The evaluation of MikiWiki is formative rather than summative (Scriven 1967) to address an ongoing collaborative process, as each cycle is used to reflect on the real situation, the socio-technical system, and evolution of the design environment. This approach to evaluation involves testing, learning, and designing new experiments as interacting activities, all occurring simultaneously.

The experiments are intended to evaluate:

1) Whether MikiWiki supports a collaborative design process and participants’ creativity, in particular participants’ situated creativity.

2) Whether MikiWiki supports a fluid transition between design for use and design in use, allowing a collaborative design process between meta-design, design and use.

3) Whether MikiWiki supports cultures of participation by providing lightweight means to allow participants with different background and different roles to articulate and share their ideas, which in turn enhances social creativity.

It is noted that creativity can be observed from two perspectives:

1) Design environments support creativity at the design and the use level, in that participants continuously adapt nuggets to form a design space in order to perform their design tasks at that moment and use the design space to externalize their thoughts immediately.

2) Designing design environments is an activity occurring at the meta-design level, in that the meta-designer sets up the initial design environment for the design session and constantly evolves it opportunistically to cope with emergent socio-technical issues without needing to change server-side code.

Although MikiWiki works for synchronous as well as asynchronous and distributed design collaboration, I choose a co-located setting as it allows immediate meta-design support. The co-located approach is particularly valuable in investigating meta-design support, since

emergent social-technical issues, user behavior patterns and dynamic interactions between various roles can be directly observed, influenced and recorded.

Thus, I am able to get instant feedback and improve MikiWiki at the meta-design level in an agile manner. In this way I am an observer, observing, learning about and reflecting upon MikiWiki on the fly. Furthermore, less coordination is needed and more attention is available for the actual design task. As such, situated creativity in action from meta-designers, designers and users can be better explored under time pressure and with limited resources.

The disadvantage of co-located meetings is that people cannot freely switch between working in solitude, communication or incubation phases. However, for gathering immediate feedback on the strengths or and needs for improvement of MikiWiki and the underlying meta-design concept, the focus on collocation is a reasonable approach.

9.3 Evaluation Methodology

The evaluation approach is an empirical and explorative observation-based field method. The reason for using an empirical and explorative approach is due to the nature of the socio-technical systems, in that they cannot be tested for software usability in isolation, but they should be examined within a social context (Abran 2004).

Therefore, this evaluation is mainly exploratory and hypothesis-generating rather than being focused on verifying an existing hypothesis. The whole evaluation process is simultaneous learning and test design, in that the tests are not defined in advance in an established test plan, but are dynamically designed and modified (Abran 2004).

The evaluation process is shown in Figure 94.

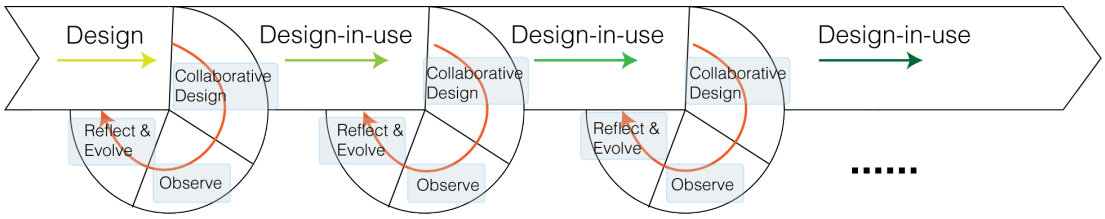


Figure 94 Evaluation process. Adapted from (MacIsaac 1995)

A design session follows these steps:

Design/Design-in-use: As a meta-designer, I prepared an environment for gathering ideas and sketching mockups in MikiWiki, with which designers drafted the creativity barometer user-interface for Android phones.

Collaborative design: Designers and users employ the environment to design the interface. Designers are participants who have designed applications, while users are participants who do not have design experiences, but have used the desktop version of the Creativity Barometer.

Observation/Feedback Collection: During the design process, I observed how users used MikiWiki to design the Creativity Barometer design. Afterwards I interview designers and users to collect feedback on how to improve the design environment. Furthermore, the interviews triggered the reflection of the participants and helped me to understand how the participants have perceived the design process.

Reflection and Evolution: Based on the observations from the previous experiment and feedback from designers and users, I evolved the MikiWiki design environment in use time for the next cycle of design-in-use.

Semi-structured Interviews

After each design session, the meta-designer conducted follow-up semi structured interviews, for a total of 13 interviews. Open-ended questions were used in qualitative research rather than to quantify the answers. I aimed to find out what participants thought about MikiWiki, their design experiences and the rationale behind their opinions.

These were the guiding questions for the interviews:

- How does MikiWiki support participants in generating and expressing their ideas?
- How does MikiWiki support participants' creativity on an individual level and on a collaborative level?
- How does MikiWiki help participants to connect and structure their design ideas and support different design phases?
- Do participants have any difficulties in using MikiWiki and how do they cope with them?
- How do they reach final agreement on design decisions?
- What is the level of satisfaction with their design results?
- What are the important differences between MikiWiki and other groupware?
- What are the best parts of using MikiWiki?
- Based on the participants' experimental experience, could they, from a subjective point of view, imagine a similar experience with clients via using MikiWiki?
- What can be improved for the next experiment?

[In01] to [In13] are used in the text to identify the 13 interviews.

Observation

Each experiment lasted 60 minutes and each design session was divided into three phases.

1) *Brainstorming and Collaborative Writing* (15 min.). Participants were required to brainstorm design requirements for Creativity Barometer, to agree on design goals, basic design elements, constrains, and to create a mood-board to illustrate design "look and feel".

2) *Sketching Ideas and Collaborative Drawing* (15 min). Participants were required to sketch the structure, navigation and components of the application.

3) *Designing with the Mockup Environment* (30m). Participants could use the mockup environment to finalize the Creativity Barometer interfaces.

Although design sessions do not directly relate to each other, certain nuggets were modified in between to support a better collaborative design process.

During the design session, I took notes during the sessions with respect to the following questions. It was possible to refine these notes by employing the video recordings afterwards.

1) How participants and the meta-designers cope with the transition between meta-design, design, design-in-use and use;

2) Whether nuggets encourage participants' appropriation with respect to underdesign;

3) How participants with different perspectives exchange their ideas and find a balance between individual preferences and collective decisions / social creativity;

4) How participants shape their design space; and

5) How participants brainstorm, articulate and finalize their creative ideas via different nuggets at different design phases with respect to divergence and convergence of ideas.

9.4 Environment Setting and Data Collection

The experiments were conducted in the ModLab in the Department of Information and Technology Management, Institute of Applied Work Science at the University of Bochum. Five collaborative design experiments supported by MikiWiki are applied and evaluated in the collocated collaboration context.

The entire design sessions were video recorded. Figure 95 shows the environment setting for conducting the experiments:

1) A large, high-resolution interactive wall (4.80m x 1.20m; 4320x1050 pixels) which seamlessly integrates three rear projection boards and displays the MikiWiki mockup environment. Touch is recognized via six cameras, which view the reflection of infrared light caused by fingers (Herrmann, 2010); The view cones of the cameras are overlapping to support uninterrupted dragging actions over the entire wall.

2) A table for users to sit and get an overview of the design stage;

3) A lectern where designers can use a keyboard to input text and interact with the screen;

4) iPads as additional input devices connected via WLAN, since the interactive wall does not support multi-user interaction. This allows participants to input text and operate actions directly on the screen or via iPads.

The environment setting is illustrated in Figure 95. There are three rear projection boards:

- C1 is on the left corner. It captures the activities between the table and the lectern, in particular focusing on the activity of the user using the keyboard. For instance, how

frequently users use text input or use the keyboard to interact with the MikiWiki environment.

- C2 is on the right corner and it captures the physical space between the table and the screen in order to understand the physical interaction space between the table area and the screen, as well as between users themselves. It provides a good perspective to observe when users are interacting directly with the MikiWiki environment via touch screen.
- C3 is at the back of the room and captures the whole MikiWiki design environment and the table area. The view cones of the cameras are overlapping to support uninterrupted dragging actions over the entire wall.

The screen-capture software records all the interactions on the MikiWiki design environment and outputs video clips, which can be used to further reflect on the design process, and on how users create new artifacts, interact, reuse, arrange and extend them.

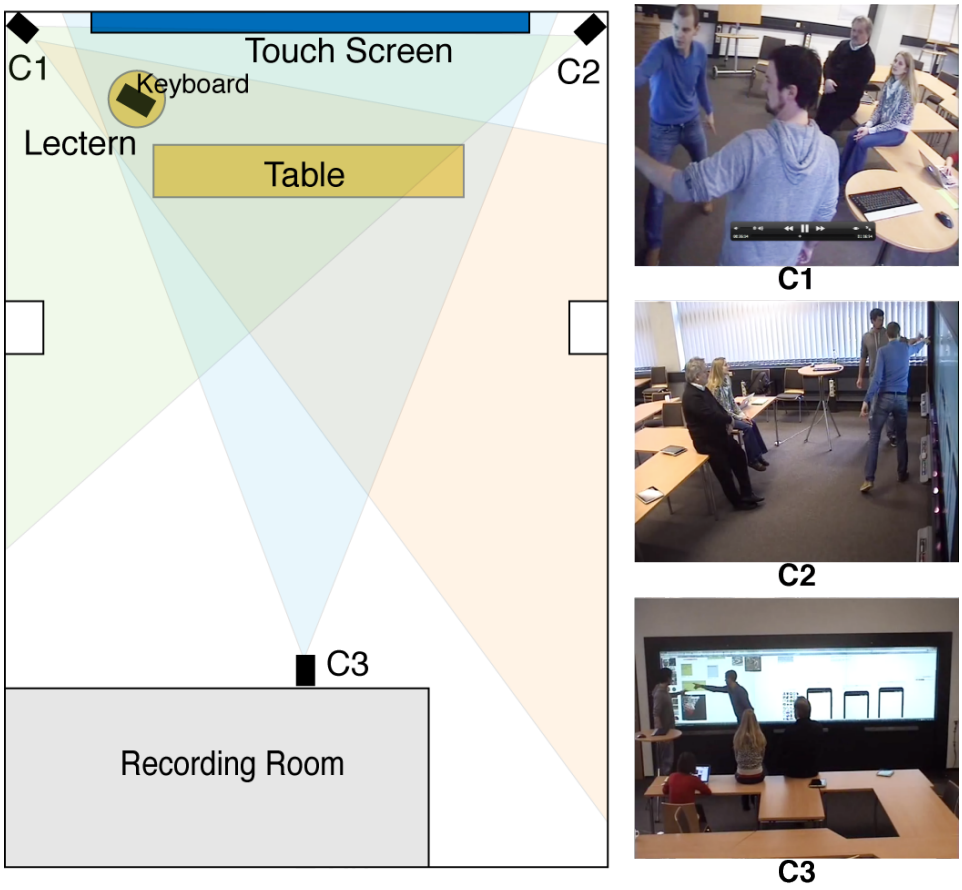


Figure 95 Experiment environment setting

After each design cycle, follow up interviews were conducted and collected as audio clips and text-based transcripts. The experiment data is collected in several formats: video clips, audio clips, video format of screenshots and log files.

9.5 Participants

The design sessions involved 11 participants (Table 8) - four female and seven male, aged from 25 to 55 years, and comprising MA, MSc and PhD students as well as associate professors. All the participants are involved in innovation, creativity, CSCW and CSCL related research and are willing to try out new technology. They have some experiences with interdisciplinary creative collaborations, and are used to using different groupware systems. Some participants are directly involved in creativity related research. Every participant has an interdisciplinary focus, ranging from computer science, and usability engineering to sociology, history and political science.

Table 8 Participant profile information

Participant (Gender)	**Age**	**Education and Expertise**
1(M)	26-30	Master in Sociology and Historical Science Organizational and Migration Research, Urban Planning, Qualitative Research Methods
2(F)	26-30	Master in Political Science & Oriental Science German Policy Development; Cooperation Development in the Middle East/ North Africa
3(M)	26-30	Master in Computer Science Privacy, CSCW, CSCL
4(M)	26-30	Master in Computer Science Creativity, User-Experience Design, Ubiquitous Computing
5(F)	26-30	Bachelor in Computer Science Video Analysis, Interaction Design, Experimental Design with Groups
6(M)	31-35	Master in Computer Science CSCW, Collaborative Modeling, End-user Participation
7(M)	31-35	Master in Computer Science CSCW, Creativity, Collaborative Modeling
8(F)	36-40	Master in Social Science Innovation Work and Processes; Storytelling; Ambient Assisted Living
9(F)	41-45	Master in Engineering Communication Technologies, Computer Sciences and Business Administration, CSCL, New Media
10(M)	41-45	Master in Computer Science Interfaces, Interaction, Usability, Cognition, CSCW
11(M)	50-	PhD in Engineering Applied Work Science, Innovation and Process Modeling, Communication Support

An overview of the participants' arrangements in groups for the different experiments is described in Table 9. Five design sessions were conducted, which were organized to involve different types of participants. Group 1 and 2 consisted of two designers; group 3 consisted of two users and two designers from the previous design session; group 4 was made purely of two users; group 5 consisted of one designer and two users. Two participants from group 1

also attended the third design session in order to validate the previous experience and evaluate improvements of the mockup design environment; therefore they were interviewed twice. The second round of interviews focused on whether they noticed any changes to the design environment from their first design session. [In01] to [In13] are used in the text to identify the 13 interviews.

I introduced use of MikiWiki and answered any usage questions related to MikiWiki during the design process. Between each experiment, the meta-designer improved the design environment according to the feedback given by the previous group.

Two participants attended the third experiment in order to validate the previous experience and evaluate improvements of the mockup environment.

Table 9 Experiment arrangements

Group	Size	Roles
Group 1	3	▪ 2 Designers (participant 4, participant 7) ▪ 1 Meta-designer
Group 2	3	▪ 2 Designers (participant 3, participant 10) ▪ 1 Meta-designer
Group 3	5	▪ 2 Returned designers from group 1 (participant 4, participant 7) ▪ 2 Users (participant 8, participant 11) ▪ 1 Meta-designer
Group 4	3	▪ 2 Users (participant 5, participant 9) ▪ 1 Meta-designer
Group 5	4	▪ 1 Designer (participant 6) ▪ 2 Users (participant 1, participant 2) ▪ 1 Meta-designer

9.6 MikiWiki Experiment Activities

The experiment environments can be seen in Figure 96, respectively from EXP1 to EXP5. Each environment has similar contents:

(1) Tasks, expressed as a todo guideline

(2) Bo folder, containing all the nuggets that are used to compose the design environment (*bo folder contents* in Figure 96)

(3) Data folder, containing all the data nuggets

(4) Icons folder, containing all the design elements for mobile devices, for instance, mobile phone icons, color icons and gesture icons

(5) The mockup_*experiment_date* MikiWiki page, for instance, *mockup_12_1211.* This is the design environment used by participants

(6) The *process* folder, containing all the MikiWiki pages for testing new or updated nuggets associated to this specific environment (*Environment Contents* in Figure 96).

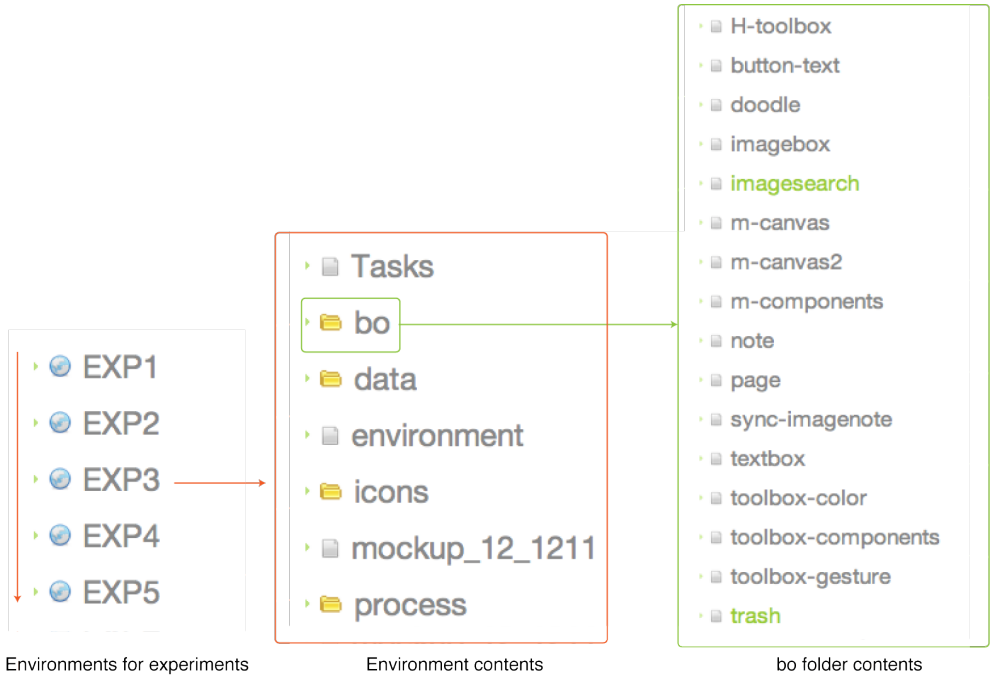


Figure 96 Experiment environments structure

Table 10 lists all the initial nuggets used to create the design environment: mockup_*experiment_date* MikiWiki page. After each cycle, in accordance with the participants' feedback and meta-designer's observation, nuggets are modified and evolved for the next cycle to better support the collaborative design process. In this way the nuggets are constantly evolving and improving. Figure 97 illustrates that participants use the *note* nugget, writing down their ideas and clustering them in different colors, while Figure 98 demonstrates participants designing a mobile interface with different *toolbox*, *canvas* and *trash* nuggets.

Table 10 Initial nuggets

Design phases	**Nuggets**	**Usage**
Start	panel	Showing different MikiWiki pages as moveable panels
Collaborative Writing	note	Writing down ideas, design specifications, and clustering them according to colors
	sync-imagenote	Translating text into visual representations, and using images to create mood boards
Collaborative Sketching	doodle	A sketch canvas for users to draw elements
Collaborative Design	toolbox	Containing android design elements
	canvas	Android phone used as a canvas
	trash	Used to remove design elements
	imagesearch	Used to search for icons from the web

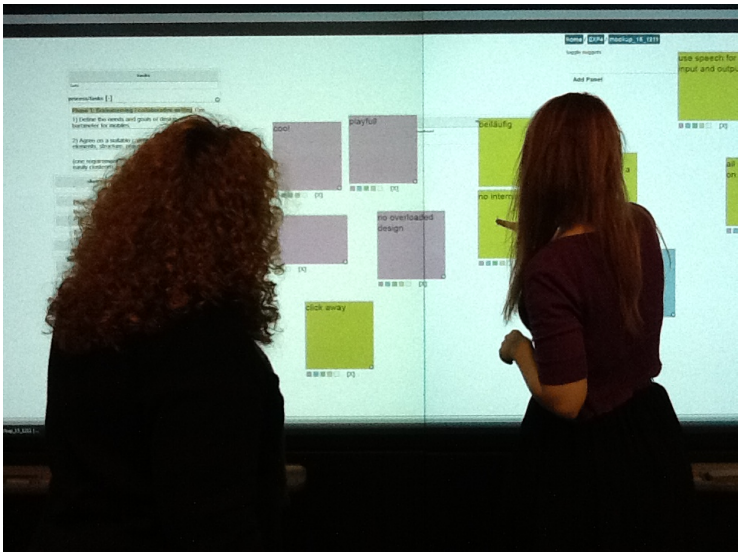


Figure 97 Brainstorming and Collaborative Writing

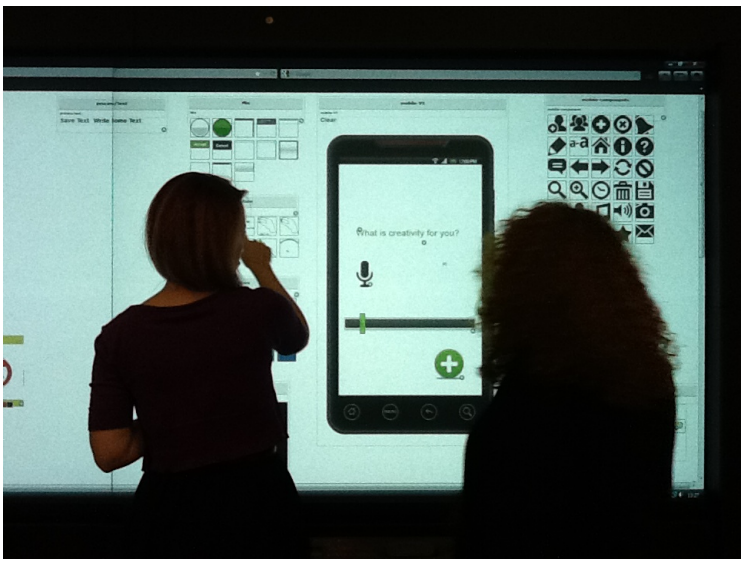


Figure 98 Collaborative Design

The design environment for the Creativity Barometer was intended to be a set of moveable panels to support direct manipulation and to fully exploit the large interactive wall. Users were able to drag, drop and resize design elements, rearrange their workspace freely and have an overview of the design stage as well as all the design resources. Nuggets were adjusted proportionally e.g. the touch area, font size, menu bar size etc. to match the large screen. Notably, the mediation mechanism of the HMS model (Section 4.2.4) is explored in this case, localizing the design environment according to the device in use (interactive wall, iPad and laptop).

The following sections describe the five experiments, focusing on technical and social issues that emerged during the design process and on how the meta-designer evolved MikiWiki and coped with these issues.

9.6.1 Drafting Creativity Barometer with Developers

Group 1 consisted of two Creativity Barometer developers as designers and one meta-designer. Two designers were involved in designing the *creativity barometer* desktop application, and they had certain design specifications in mind already. They used the sync-imagenote, doodle and mobile canvas nuggets to create their final application mockup (Figure 99).

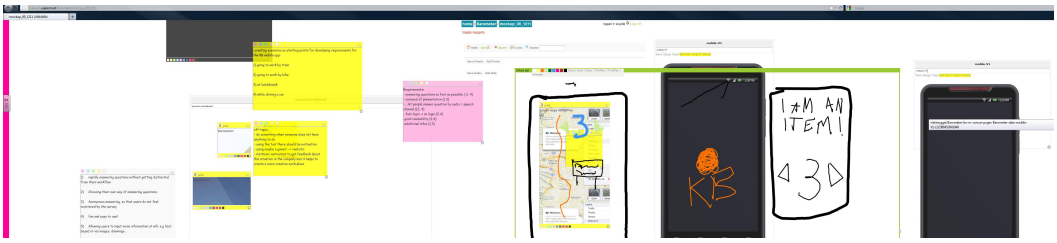


Figure 99 EXP1 Design results

Several issues, both technical and social, emerged during this experiment.

Technical issues

(1) *Note* nugget
- Font size was too small for the large-screen interaction.
- The delete button was too close to the drag handler, which could cause accidental deletion when participants tried to drag the note handler (the yellow note on the left in Figure 99).
- The handler was too narrow to be dragged around the large screen.

(2) There was a lack of a clear list of tasks available on the screen, making it hard for participants to have a clear direction of the next steps, to track what they had done, and to manage tasks according to time.

(3) *Doodle* nugget: The "*Clear*" and "*Draw*" buttons were next to each other, which – according to participant feedback - could cause designers to erase all their drawings by accident (in green in Figure 102).

(4) The meta-designer only provided one mobile phone canvas for the final mockup. However, designers used the mobile phone canvas for sketching their ideas before reaching the final mockup. They used the panel nuggets to create a new mobile phone canvas and solved this problem.

(5) The images generated by sync-imagenote nugget did not resize proportionally, which made it difficult to create a mood board and to view images.

(6) The toolbox nugget provided too many elements and there was lack of structure, making it difficult to browse and use it effectively.

Social issues

(1) At the beginning, designers were mostly talking rather than interacting with the wall, not leaving a trace of their thoughts and discussion on the wall.

(2) Designers did not take responsibility for what they were saying, and tended to forget some of their suggestions. This made it hard to track the genesis of ideas.

(3) Designer 2 was goal-oriented and questioned the benefits of creating a moodboard for the mobile application design.

(4) During the brainstorming phase, designer 2 disagreed with designer 1's idea. However brainstorming is meant to be a phase of work where people do not discuss pro and cons but just collect ideas in an associative way.

(5) The experiences that they had from their *creativity barometer* desktop application did not help them to start easily, but rather blocked their thinking at the beginning. They took more time to engage in the design process. As designer 2 commented: "*We worked on this project for such a long time. We had many meetings. We talked about so many situations* […] *We kind of focus too much on what we are doing.* [referring to the MikiWiki experiment] *I thought that it's just about collecting ideas, creating ideas, this came up that I didn't expect. It's really surprised me.*" [In02]

Meta-design in-between

Many of the technical issues that occurred in the EXP01 were addressed during the meta-design cycle before EXP02.

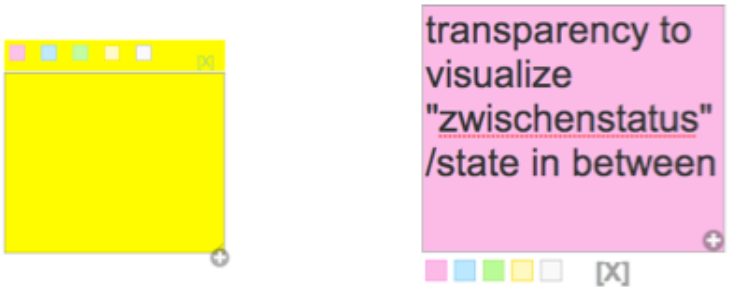


Figure 100 EXP01 note nugget (left) and EXP02 note nugget (right)

(1) The note nugget (Figure 100)

- Increased the note nugget font-size.
- Moved the color picker to the bottom of the note.
- Removed the handler and enabled the dragging interaction directly on the note text area.

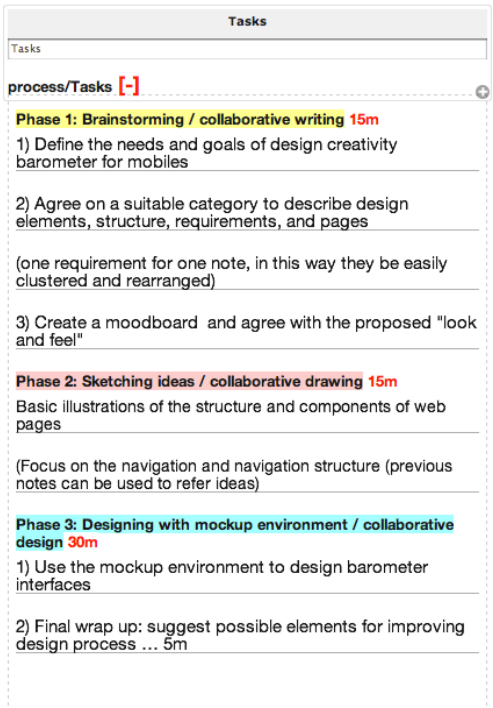


Figure 101 Design tasks in different phases

(2) The meta-designer combined the expand nugget and the panel nugget to create a design tasks panel (Figure 101) to visualize task lists. In this case, participants could move the tasks panel around and toggle or expand it to save space or refer to it in conversation.

(3) The doodle nugget: In order to separate the *Draw* and the *Clear* buttons, and since the *ThinPen* and the *ThickPen* buttons were more frequently used, the meta-designer moved the *Clear* button to the end (Figure 102).

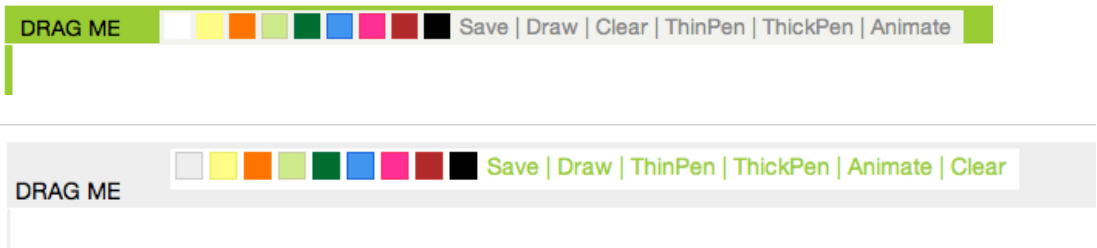


Figure 102 EXP01 doodle nugget (in green) and EXP02 doodle nugget (in grey)

(4) More android phone canvas nuggets were provided for the EXP02.

(5) By specifying the text-area background-image *'-moz-background-size': '100%'*, the image note could be resized proportionally in the Firefox browser.

(6) For the EXP02, the meta-designer simplified the toolbox design elements, which were oriented to the android phone. In addition, a toolbox with a set of colors is provided in order to see how participants interpret and use colors.

All these changes were performed by the meta-designer easily and quickly (approx. 2 hours) on the client side mainly in JavaScript or in HTML code. The existing nuggets format pages could be inspected, edited and tweaked as wiki pages from and within MikiWiki, without leaving the system or requiring additional tools. As an example, separating the order of the "*Draw*" and "*Clear*" buttons of the *doodle* nugget was merely about repositioning the "*Clear*" button code after the "*Animate*" button.

9.6.2 Drafting Creativity Barometer with Designers

Group 2 consisted of two designers and one meta-designer. The designers are experienced interaction designers and - due to the newly visualized design tasks - they just dived into the design process.

Technical issues

(1) *Doodle* nugget:

- Buttons were too small, as was their label font size
- Participants could not interact with the nuggets that were behind the *doodle* nugget. Participants could combine the *doodle* nugget with other nuggets, for instance combining the *sync-imagenote* nugget, and drawing on top of image notes. The *doodle* nugget was always automatically positioned in front of other nuggets to allow drawing.

(2) *Sync-imagenote* nugget

- There were issues with the profile picture. The initial *sync-imagenote* nugget was created to support distributed multi-user collaboration. Therefore each note header had the creator's profile picture, and if someone made any operation on the note, the profile picture would be enlarged to indicate who changed it. This profile picture was not only unnecessary in this case, but also when participants tried to type any word, the enlarged profile picture overshadowed part of the note text area and thus made it hard to read the text.
- Font-size and color pickers were too small for the large interactive wall

(3) When designer 3 tried to refresh the page there was a breakdown because they could not drag design components anymore. However, after refreshing the mockup page, they lost part of their design.

(4) The movable panels can be moved around, but they cannot be hidden, cluttering the working space and making it difficult to structure the design space and to have a clear design structure. As designer 4 commented "It would be easier for example, at the last phase, if I have also the other buttons, and also the brainstorming elements are on the same canvas. If you can structure and hide something from your perspective… I think this could provide some help…" [In04]

Solutions on the social level

Social issues that occurred in the EXP01 were addressed accordingly in the EXP02.

I became more aware of my role after the first experiment, namely as a facilitator to guide participants using MikiWiki, rather than being actively involved in the design process. As such, emergent creativity from participants could be observed better. The facilitator role does not need to be identical with that of the meta-designer. It is noted that as an action researcher, I introduced interventions during the experiments as a facilitator (therapeutic stage), afterwards I reviewed my observational notes and the videos (diagnostic stage), and then I could introduce interventions again as a meta-designer before the next experiment, the process was then repeated (Section 3.1).

In order to encourage participants to interact with the touch screen as well as document the entire design process, the facilitator made it clear to the participants that every discussion should have a trace on the screen. The facilitator reminded them of doing so during the design process when necessary. Since participants tended to forget their responsibilities and what they had discussed, leaving traces on the screen (such as using notes to assign meaning and track individual responsibility) helped to minimize this issue.

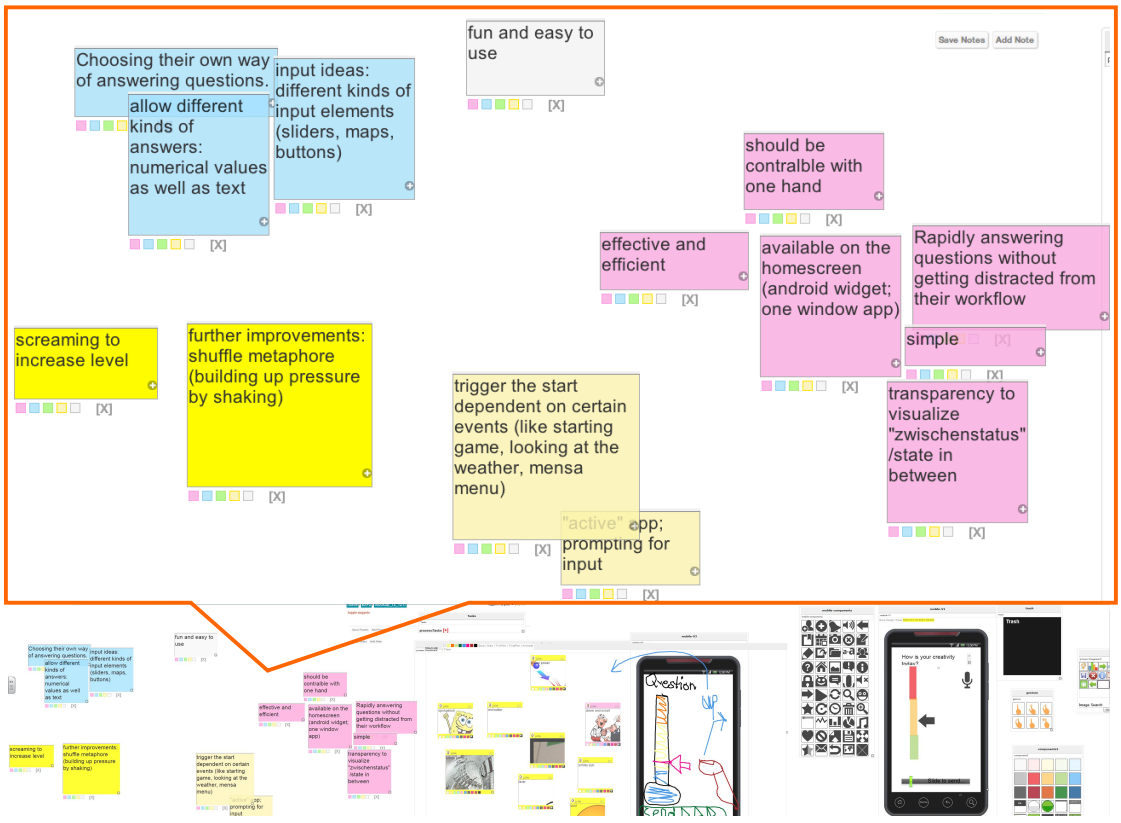


Figure 103 EXP02 design results

I provided an initial set of notes as examples of using notes. Designers therefore followed the examples, wrote exactly one idea on each note and clustered notes by color according to the content. As for creating a mood board, using searched images helped participants to visualize

their ideas more effectively. I demonstrated how to create an image note from the "*cat*" keyword, getting the designers started on the process. Designers were interested in this example and started searching and creating images for expressing more abstract concepts or emotions, for instance "*fusion power*", "*sensory*", "*boiling*" and similar.

Figure 103 illustrates the final design results. The key idea for this Creativity Barometer mockup is to support answering survey questions with only one hand. The application has a vertical slider, allowing a user to answer questions with a graduated vertical slider. A second horizontal slider is used to send or cancel answers. A final end-user can use his voice or a shaking gesture to answer the questions.

Meta-design in-between

In the course of this iteration we addressed the new problems that emerged during EXP02.

(1) *Doodle* nugget (Figure 104)

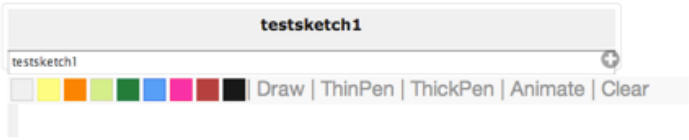


Figure 104 Doodle nugget for EXP03

- The meta-designer disabled the draggable operation, since the grabbing handler of the doodle nugget was too narrow to provide users with an intuitive dragging operation.
- The *doodle* nugget was included by the *panel* nugget to support a better dragging operation.
- The button label size and the color picker size were increased correspondingly to better match the coarser touch-wall interaction.

(2) Changes made to sync-imagenotes are the following:

Figure 105 EXP02 Sync-imagenote (left) and EXP03 Sync-imagenote (right)

- The meta-designer increased the sync-imagenote font size, and initial note size to match the large interactive wall.
- The meta-designer removed the creator's profile picture and creator's name (Figure 105)

(3) The meta-designer added an auto saving function to each nugget. Previously all the individual nuggets had their own "*save"* button, therefore each time one made changes, one had to remember to press the "*save*" button to store changes. For instance, the doodle nugget had a *save* button for saving sketches, the note nugget had a *save* button for saving notes, the mobile canvas had a *save* button for saving mobile icons, and so on. Therefore, participants had to constantly remind themselves to click different "*save*" buttons to save changes for different nuggets. It was a task that interrupted design-thinking flow and it was error prone. Adding auto saving function to each nugget supported participants' design-thinking flow and freed them from worries that their work might get lost, building greater system trust.

(4) The meta-designer updated the panel nugget by enabling minimization and expansion of the panel by double-clicking on the handler. Figure 106 shows two different states of the doodle nugget: in the expand mode, one can sketch on the sketch board; in the toggle mode, one can interact directly on the mobile canvas, which is below the doodle nugget, for instance by dragging and dropping design icons on the mobile canvas. This should effectively enable participants to switch on and off the doodling mode, without having to reposition the doodle nugget.

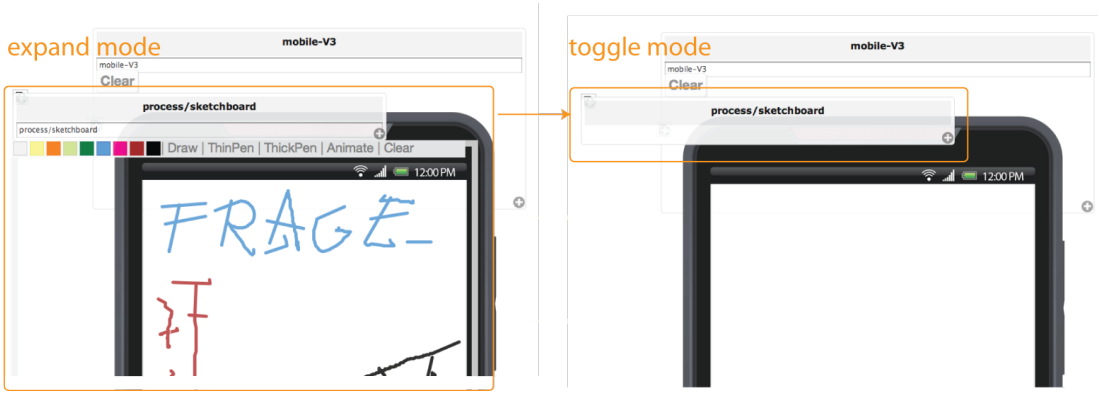


Figure 106 Panel nugget in EXP03

9.6.3 Collaboration between Developers and Users

Group 3 consisted of two *Creativity Barometer* developers, who were designers from group 1 and two users. One of the users has an art background and she was very excited to express her ideas. Participants were thinking by free association (Mayer 1983; Shneiderman 2000), using image notes representing various ideas such as a jumping cat, piano keyboard, angry

boss, horse riding, and so on. The ideas generated in the brainstorming phase were loosely connected with the final results, where they produced three different application designs.

The first one was represented as a sketch (Figure 107). Seven blue holes indicate a 1-7 answering scale. Users can blow a balloon towards the right hole to answer questions.

The second application uses a basketball to answer questions, by dragging the basketball into the basket. The number of balls that the user puts into the basket indicates the degree of the answer. The designer represented this idea by combining image notes, a mobile canvas and gesture icons.

The last design is presented by combining the mobile canvas, image notes, mobile icons and textual notes with additional design explanations. This design represents the answer as a flower that can be grown and watered with a touch gesture. This is represented by combining the mobile canvas nugget and the doodle nugget, thus allowing participants to draw on top of the mobile canvas. The flower idea was originally from participant 8, and participant 7 sketched and represented the interface in a collaborative effort. *"[…] I could see he understood my ideas, and he expressed it better in his special way."* [In07] It was a collaborative effort.

The common characteristic behind these three different solutions is the use of visual representations to prompt and to answer questions, emphasizing the fun and playful criteria of the application.

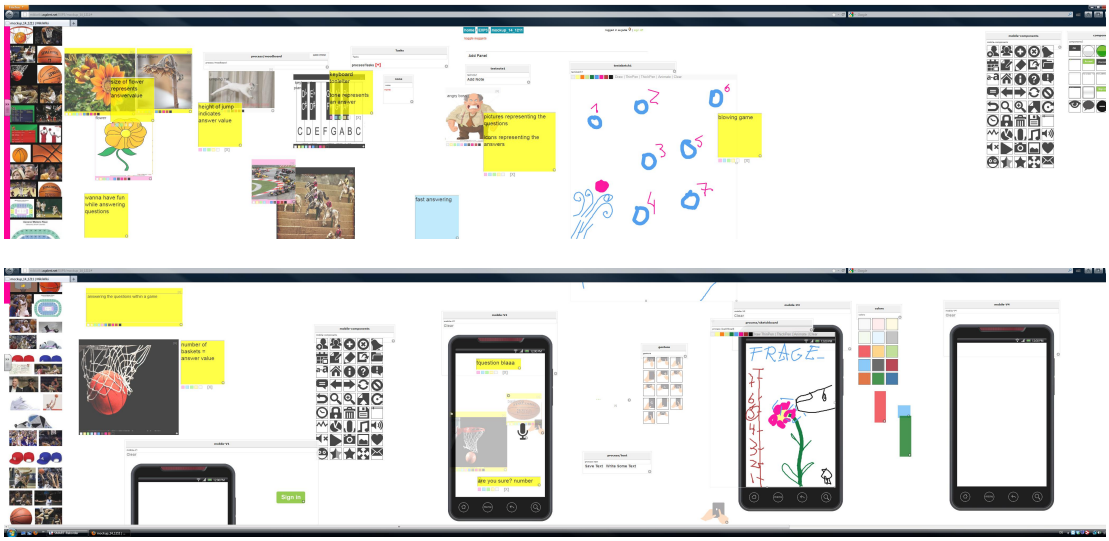


Figure 107 EXP03 design results

Technical issues

(1) Doodle nugget

- The *draw* and the *erase* modes were assigned to one single toggle button, and therefore when participants wanted to erase sketches, they needed to switch to the

erase mode. After they had done the erasing, they needed to switch back to the draw mode to continue sketching.

- Lacking an obvious indication of which mode the user is in: sometimes they were trying to sketch while they were still in the erase mode and nothing could be drawn on the canvas. Participants got confused when this happened.

(2) Panel nugget: the toggle or expand status of the panel was not saved to the data page, and so when participants refreshed or closed the design environment, the status could not reflect the real situation of how participants modified or adjusted their design environment.

Social issues:

(1) Since the interactive wall did not support multi-touch interaction, the participants had to take turns bringing their ideas to the screen. While that could slow down the process, on the other hand it gave participants time to review the design process and improve other participants' ideas. As participant 4 stated, "because we don't have multi-touch on our wall I could step back, and I could see where I am." [In06]

(2) Language barrier. As participant 11 noted, being a German having to speak in English, language was a barrier to express ideas in this situation [In03] [In08] [In12]. Participant 11 had many ideas, however, it was hard for him to express them in English. While he was struggling to put his ideas into an appropriate form, other participants went on to the next step. It also took some time to bring other participants' ideas to the screen. When it was participant 11's turn, he already forgot some of his ideas. Being able to record ideas and reflect upon them later on is crucial for design. "*Every idea I had, I need to put them somewhere, if I don't put it down, I will lose it. When I kept them, I could always drag and drop and combine them later on."* **[In06]**

(3) In terms of using MikiWiki, there was a big gap between second-time participants and new users. "*I was in a more advanced stage. I even confused others, since I used the system so fast. I knew exactly I was doing […] I learned the basics of your framework, now I could start working very fast."* **[In06] [In05]**

Both designers perceived that their increased proficiency in using MikiWiki on their second experiment was due to increased familiarity with the system compared to the first time, rather than due to the technical improvements: "*I think the main difference was that I am more familiar with the system rather than the changes you made. The auto-saving was good… you fixed some minor things."* **[In05]**

Meta-design in-between

In the course of this iteration we addressed a few issues that were observed during EXP03.

(1) The meta-designer modified the doodle nugget format page, providing the *draw* and *erase* functionality as two separate buttons rather than one single toggle button (Figure 108).

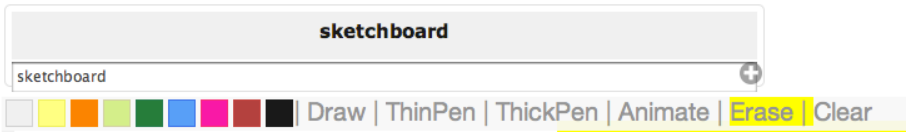


Figure 108 Doodle nugget for EXP04

(2) The meta-designer added the auto saving function for the minimized/maximized state of the doodle nugget, allowing the state of the workspace to be stored more accurately.

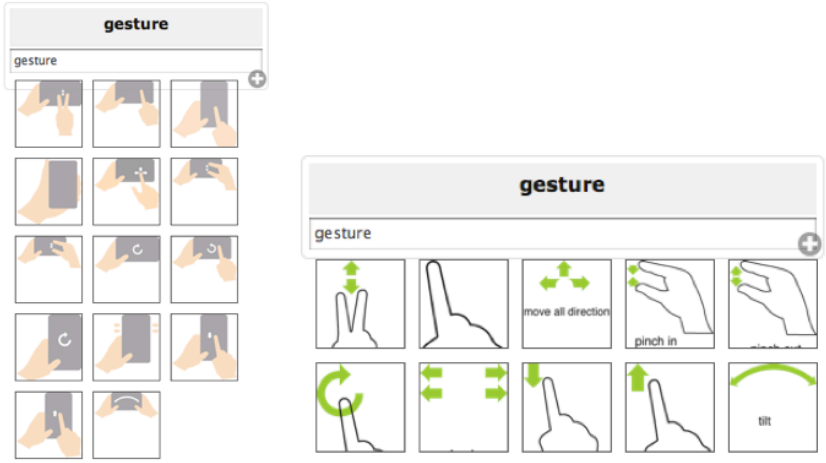


Figure 109 Gesture toolbox nugget in EXP03 (left) and in EXP04 (right)

(3) The meta-designer redesigned the gesture toolbox - whose icons were too big in the EXP03 - so that they could be used more effectively to indicate interactions for the final phase mockup design. The new gesture icons were redesigned to match the rest of the design elements' style (Figure 109).

9.6.4 Drafting Creativity Barometer with Users

Group 3 consisted of two users, who have some knowledge and experience in interaction design. They followed and completed all the design phases. Figure 110 shows the final design result. They aimed to create a simple, functional and personalized design. The right mobile canvas shows the questions page. Users can answer questions by sliding or by using voice. The mobile interface in the middle shows the statistics results; the hand performing the pinching gesture signifies the possibility of zooming in the statistics to view more detailed information.

The meta-designer encouraged the participants to express themselves in German whenever they got stuck in expressing their ideas, to ease the language barrier noted during EXP03. The social issues noted in EXP03 did not occur in this group, as it had fewer participants, with a more similar background.

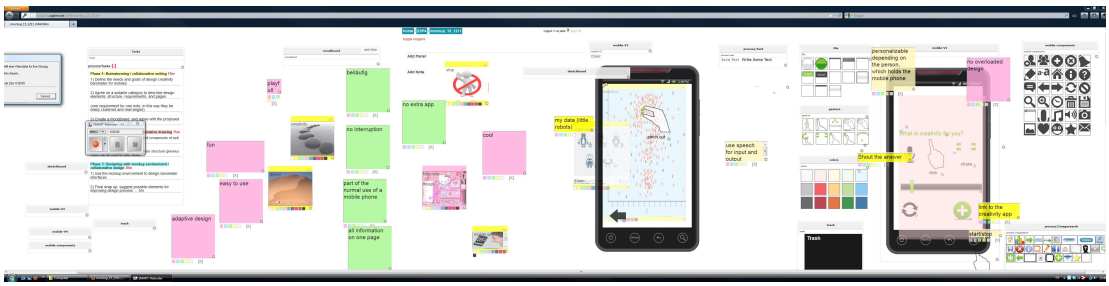


Figure 110 EXP04 design results

Technical issues

(1) The *sync-imagenote* nugget allows participants to create image notes and move them around by dragging the individual image note's handler. However, this interaction style is different from the one employed by the note nugget. The note nugget allows users to create a note and directly drag it by touching the note rather than touching a specific handle. The different interaction style caused confusion to users. Several times the participants tried to move an image note and drag its text area rather than its handler.

(2) The *doodle* nugget was not easy to use for participants, partially because the screen capture software was running at the same time (slowing down the interaction), partially because it took time for participants to understand how to overlap it to other interface elements and interact with it.

Social issues

The participants had different opinions about the application "look and feel". Eventually they designed two different mockup styles: a *robotic* style and a *hello kitty* pink style. They agreed that this application could be personalized to different situations as well as to different users to accommodate the two different styles of their final mockup application. This solution matched to one of their design criteria, i.e. supporting personalization.

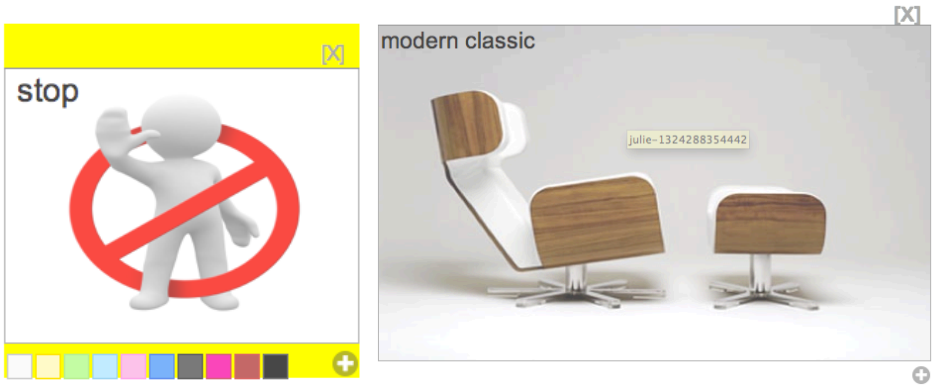


Figure 111 *Sync-imagenote* nugget in EXP04 (left) and in EXP05 (right)

Meta-design in-between

The meta-designer removed the sync-imagenote handler as well as the color picker and enabled dragging directly on the image note text-area to provide a consistent interaction style across nuggets (Figure 111). Without additional borders and colors, the updated sync-imagenote differentiated itself from the note nugget and it allowed the creation of more appealing and less cluttered moodboards.

9.6.5 Collaboration between an Experienced Designer and Users

Group 5 consisted of one designer and two users. The main technical issue was still the lack of multi touch capabilities for the interactive wall.

Figure 112 EXP05 design results

The final application mockup is illustrated in Figure 112. It is a simple interface with five buttons on the corners. The main concept is that while a user listens to music and a survey question pops up, he can hear the question and answer it with a screen gesture without looking at the application. "*If you are used to using it [referring to the application], when you use it, you don't have to look at it. This is the basic idea behind it"* **[In13]**.

Technical Issues:

(1) One of the common technical issues was when users created many notes, sketches and mockups, the whole canvas became chaotic. It was difficult for participants to filter or browse information by type or by layer.

(2) Overlapping and combining multiple nuggets provided certain advantages to externalize ideas, e.g. combining the *canvas* with the *doodle* nugget allowed sketching on top of the mobile interface. As such individual nuggets became dependent on each other. However, since MikiWiki does not support selection of multiple elements, moving multiple interrelated design elements became a tedious and repetitive task.

Social issues:

At the beginning, and in particular during the brainstorming phase, it was difficult for users to come up with design ideas and criteria. Users were observing and listening to the ideas proposed by the designer and learning how to design at the same time. In this case, the perceived gap between the designer and the users pushed the designer to take the lead in the design phases, while users were somewhat more reluctant to express their ideas [In13]. Users generated fewer solutions and shared less of their own knowledge with the group than I observed in the previous experiments.

However, users posed more questions to the designer. Through this Q&A process, both sides achieved better communication and understanding. The designer could better understand the users' concerns, while the users could understand better the potential design space, and the technical considerations and rationale behind each of the design decisions.

In Summary: During these experiments, the meta-designer was continuously evolving the design environment based on observations, interviews and participants' suggestions about features. Therefore, MikiWiki became more sophisticated in supporting Creativity Barometer design after five rounds of experiments. The needs for adaptations by meta-design became not only less but also more complex. Remaining technical issues, e.g. batch selection, layer filtering or editing, since more complex design specifications require more meta-design efforts and a longer period. However, these technical issues could be handled at a social level instead of from a technical approach, as became clear when users had more experience with MikiWiki: for example two designers pointed out during their second design session that MikiWiki was very easy to use [In05].

With respect to meta-design, due to the limited time designers and users only focused on the design level and did not carry out meta-design and EUD tailoring activities. Additionally, to support meta-design in a co-located context, a more interactive environment is appropriate; for instance, code and data can be inspected, edited in place (editing the text on the same page without refreshing it) and compiled in real-time.

9.7 Observations and Reflections

This section introduces some observations of how participants appropriated MikiWiki to cope with problems.

Importance of Initial Examples

In the first experiment (EXP01), the initial environment contained an initial example note containing all the design criteria. However this turned out to be an inappropriate prompt as the designers, influenced by this example, tried to write down as much information as possible into a single note. This made it impossible for them to rearrange and cluster ideas according to different design criteria and priorities at a later time.

The designers created only two notes, one for the scenarios (yellow note) and the other one for the design criteria associated to each scenario (pink note) (Figure 113). On the other hand, it is interesting to note how the designers adapted to the initial structure and produced an interesting bricolage. The solution that they came up with was to number the scenarios and associate design specifications to corresponding scenario numbers; for instance, *answering questions as fast as possible* (in the pink note) matches scenario 2, *going to work by bike*, and scenario 4, *while driving a car* (in the yellow note).

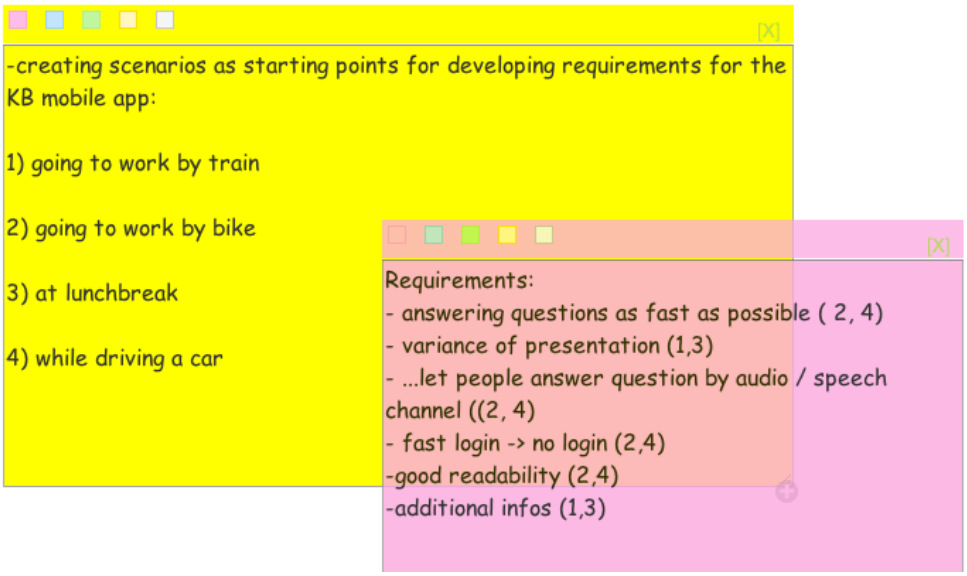


Figure 113 Design scenarios and corresponding design specifications

For the second experiment, the meta-designer created a different input example by providing sample design criteria as five different colored notes (Figure 114). In the following experiments, participants created many notes and clustered them in different colors according to different categories.

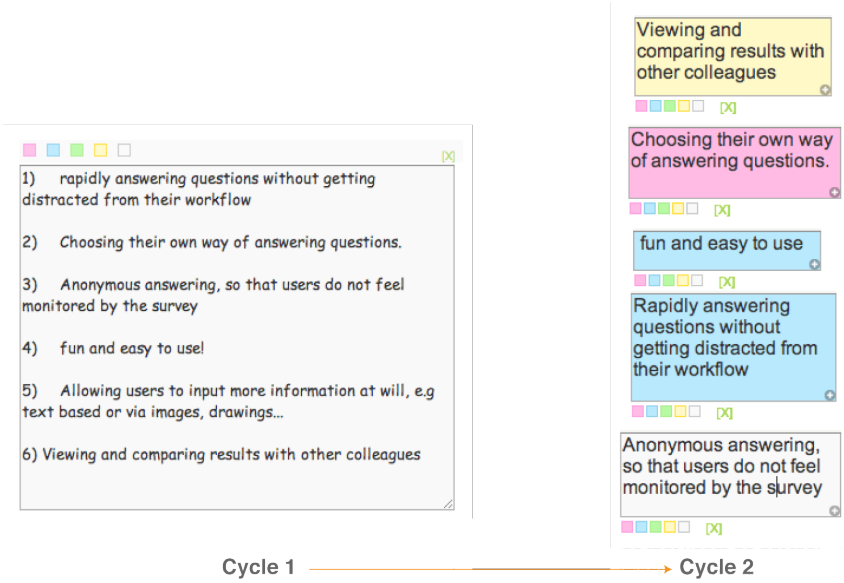


Figure 114 Note nugget example as an initial prompt

This example illustrates how a simple note example shaping designers' brainstorming activities. To this end, meta-designers should be mindful of initial examples to ease the blank page syndrome (Section 8.5.7) but also to encourage designers' creation activities.

The Seeding Process

Seeding is the driving force behind co-evolution. (Fischer et al. 2001) define "seed" as the "development of an initial system that can change over time". In this case the seed is MikiWiki itself.

An important observation in these design sessions is that evolutionary growth emerged from reseeding. The evolutionary growth phase and reseeding phase were intertwined rather than being two distinct phases since the system is open and flexible enough for reseeding and meta-design in use time.

Nuggets are the building blocks of MikiWiki, reflecting the boundary objects of the HMS model, each nugget being a self-contained "*underdesigned*" and "*open*" seed. The principle of underdesign (Section 4.2.7, Section 5.7) supports further continuous creation and evolutionary possibilities. Therefore MikiWiki can be seen as a collection of seeds, breaking down the initial system into smaller seeds: each nugget can be inspected, adapted and evolved continuously, blurring the distinction between design time and use time. The evolutionary growth phase and reseeding phase are tightly coupled together within the same system and each nugget can be seen as reflecting a micro-SER process.

Figure 115 illustrates how the meta-designer evolved nuggets in between each design session, based on her observation on how participants used the various nuggets, what difficulties they experienced, as well as their feedback. The progression of seeding nuggets,

evolving nuggets and nuggets in use as part of a continuous flow, as well as the co-evolution that took place between users, designers and meta-designers is depicted.

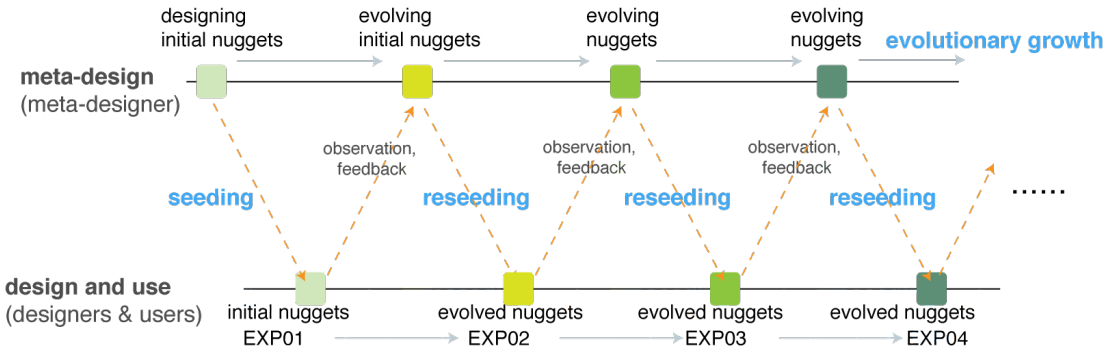


Figure 115 Evolving the nuggets over time

The initial design examples provided by meta-designers can also be seen as seeds. They are externalized design ideas and tangible artifacts that can be comprehended by users and which can inspire interaction and appropriation.

Besides nuggets there are several other types of seeds that I observed in the course of the experiments. Seeds in this context are initial examples provided by meta-designers as well as concreted artifacts without involving meta-design activities.

(1) Seeds from meta-designers

Before each session, the meta-designer showed designers how to create a note, and came up with an initial set of nuggets such as design criteria for the mobile app and a mood board to express the final application look and feel. The demonstration examples became seeds for the designers, not only showing how to use MikiWiki, but also inspiring them to other possibilities.

Figure 116 Searching for inspiration (left) and providing supplementary information (right)

Participants could take examples further by developing, refining and appropriating them. Figure 116 illustrates participants in one case using a *sync-imagenote* to search for inspiration (left), randomly associating notes with an image; and in another case using a *note* to provide additional information on their design (right), representing ranking data as little robots. Both cases are different from the originally provided examples, using notes to collect or cluster design requirements and using sync-imagenotes to create a moodboard.

(2) Seeds between designers and users

Articulating ideas between designers and users was a learning process, developing mutual understanding. Working on the same problem space within an interactive space had an effect of "reciprocal acceleration of creativity" (Zhu and Herrmann 2012b).

[In01] *The doodle nugget was very useful, because of that, we could see for example someone is developing and evolving ideas, just by looking at what he was doing on the wall. It was very inspiring to develop your own ideas. That was quite cool…*

[In03] *It's a decentralized brainstorming,* […] *so it was quite good that we can discuss every idea, you can generate more ideas from those others had before.*

[In05] *Then came the same effect, I listened to the other people, the kind of ideas, what they were writing, they were drawing, afterwards, I started thinking, thinking. Once again, with the ideas I created mine based on others…It's working.*

(3) Seeds from self

While working though the design phases, a user left a trail of artifacts that he could later on reuse and evolve for the following design phases.

[In04] *For example, I looked back to the ideas we collected earlier. One criterion was "how could we have fun with it, why don't we use this?" And you have this icon available, the microphone, just drag this microphone icon there, and how would it be pressing this button, you cry into this, it's a microphone and the louder you cry, the higher the value it is…. This was one of these incubations based on the early idea that came up later. It's always somehow iterative.*

Appropriating Tools and Processes

Traditionally, mood boards are collages of images used by graphic designers and art directors to illustrate a stylistic choice and focus their efforts. Before starting the experiments, the meta-designer showed the participants how to combine different images to create a mood board to convey the idea of a 'retro' look.

However, in EXP02, EXP03 and EXP04, rather than creating a mood board, participants combined text notes with image notes to explore the design criteria visually and further cluster them in a way that would help them envisaging their final application, appropriating the tool

and the process to perform brainstorming and explore the final look and feel of the application (Figure 117).

This may have happened because they needed to explore more possibilities and they focused on divergent thinking, making it hard to come up with a coherent style and create a single mood board.

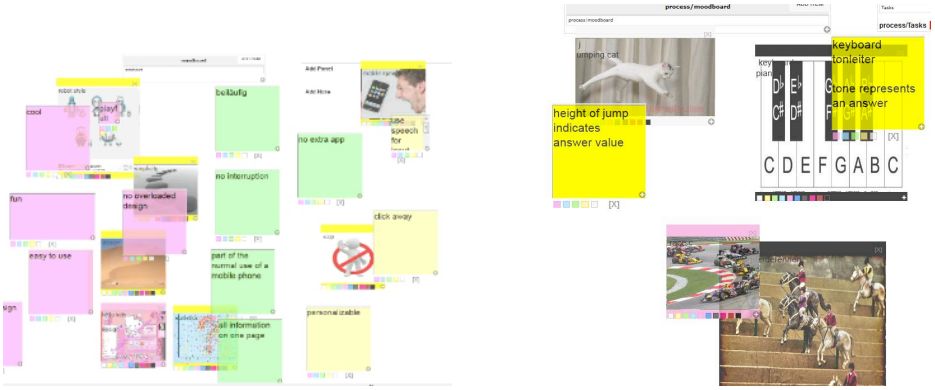


Figure 117 Clustering and combining both text and images

On the other hand, participants created a novel way to externalize their abstract textual requirements and ideas, augmenting them with images that acted as further seeds to inspire new ideas and explore the design space in a tangible and visible manner.

Multi-modal Expression

Figure 118 illustrate that two designers (in EXP01) used different means to express their ideas.

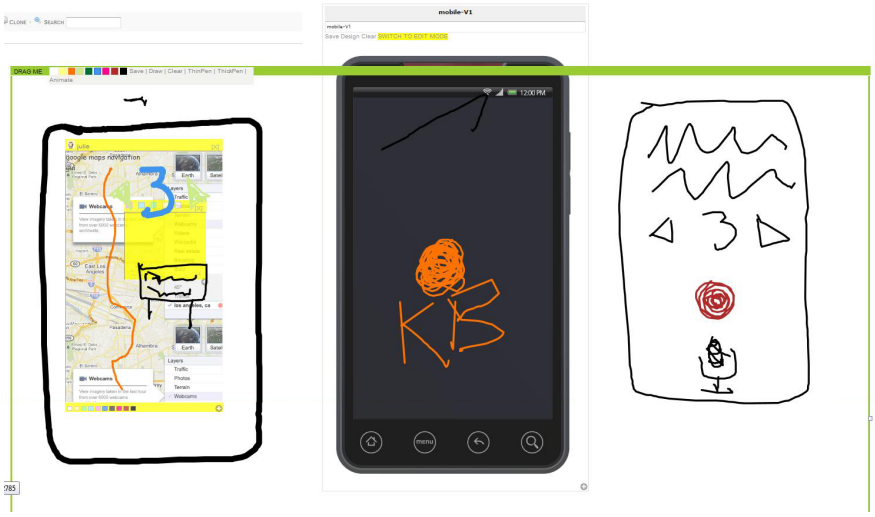


Figure 118 Different ways of expressing design ideas

Designer 2 searched a Google map via sync-imagenote and sketched on top of it (Figure 118 on the left). **"***I really like the combination of using instant web search image, and just drawing over them, just like pulling everything together* […]" **[In02]**

Designer 1 utilized the mockup mobile canvas and sketched a red button on top of it, an audio speech to answer questions, to illustrate his minimalist style. Designer 1 is less competent with sketching than Designer 2, and he preferred to express his ideas starting from existing resources, utilizing the mockup mobile canvas, rather than from scratch and redrawing something that already exists.

The final result (Figure 118 on the right) shows how the two designers merged their design ideas into a single mockup.

Conveying Shared Context

In EXP04, after finishing their mockup design, participants annotated the mockup interface with text notes (Figure 119) such as “shout the answer”, “click”, “link to the creativity app”, and so on.

Figure 119 Using annotation to convey additional context information

In this case, an intersubjective understanding between participants (Rommetveit 1974; Mørch 2007; Fugelli 2010) continually shaped and evolved throughout the course of the co-construction of the Creativity Barometer and verbal communication between participants. This intersubjectivity among participants might not be obvious, yet it is necessary for future audiences to make sense of their design. The notes as prompts and cues make their tacit shared intersubjective understanding explicit and understandable.

Furthermore, an interesting point here is the difference between documentation and short-term communication support. Normally verbal communication is more ephemeral while comparatively writing as documentation is comparatively more permanent. With respect to requirements engineering (Zhu and Herrmann 2012a), sketching or drafting mockups is not meant for documentation but as a process for externalizing and visualizing abstract internal thoughts and developing mutual understanding. From this perspective, MikiWiki is not a

platform for documenting how a system (e.g. the Creativity Barometer) should be, but is creativity-related in that it facilitates what participants must know about their mutual ideas to be creative together.

Simple Elements, Easy Repurposing

In EXP02, designers wanted to use a vertical slider to symbolize the barometer. However, the existing toolbox provided only a horizontal slider. Designer 3 proposed that “maybe it would be easier to just try to draw something like a box, just say that it’s a vertical slider...” He then used colored boxes icons (Figure 120) to create a vertical graduated slider.

In this case, the meta-designer intended the color toolbox to provide simple and more generalized design elements, therefore making them easy to be appropriated and used in many different situations.

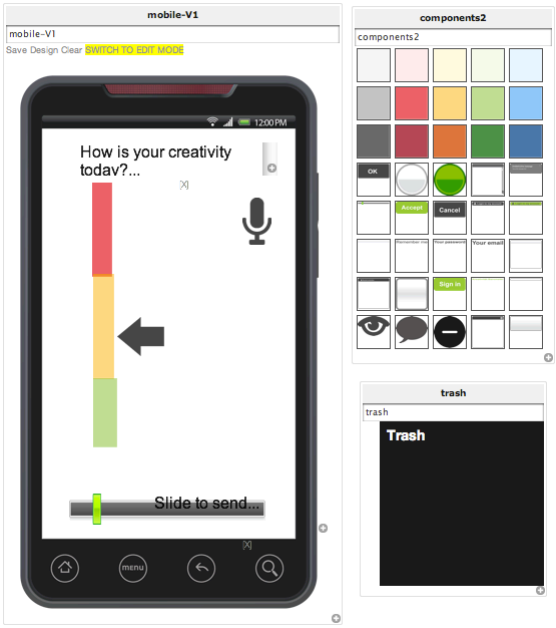


Figure 120 Repurposing color icons

Embodying Abstract Concepts

Visual interactive representations provide a sort of physicality to abstract concepts, prompting participants to play with them.

[In01] *I was quite fascinated by this mockup tool, because on the one hand, those are just screenshots or images, but it’s very useful that you can use a set of icons, you can combine them with pictures. In our case, it’s mobile phone pictures, by the connection between Google image search and your nuggets, it’s possible to implement every kind of picture… you can just work with pictures. This helps me much better than working theoretically, just talking about something, not seeing something.*

Embodied artifacts act as boundary objects to convey ideas and create shared understanding via the social manipulation and association of those concepts.

[In10] *Yes, because every tool creates shared common understanding very fast, and creates some common artifacts that you can discuss.*

Using the touch screen was a full body immersive experience, where people could communicate meaning just by turning the head, approaching a certain area of the screen, or nodding in the direction of a specific artifact.

MikiWiki was not intentionally designed for collocated rich interaction with the physical space and even external artifacts (Figure 121). It was interesting to observe how this mixed hybrid environment could allow people to solve conceptual problems via physically intuitive interaction.

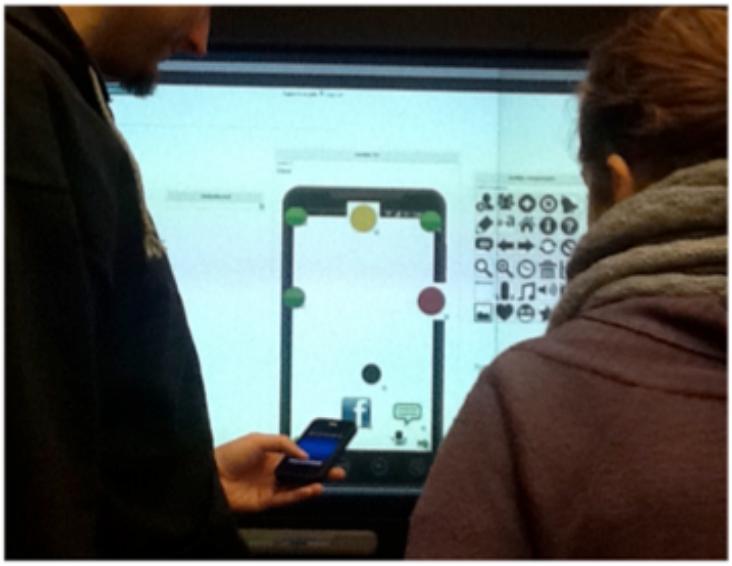


Figure 121 Using a physical mobile to augment the meaning of what is seen on the screen

[In13] *So these sort of meta terms, like simplicity, all these kinds of things, don't mean very much to me and don't help me very much in designing things. But when I have a concrete object I can do design upon, this matches my thoughts better.*

Some of this embodied social richness is present in MikiWiki in the form of awareness-oriented nuggets, but it would be worth exploring more in this direction.

9.8 Creativity Support

Creativity support is an essential aspect of meta-design. However, understanding how technology can facilitate the process of collaborative creativity is still in its infancy.

1. MikiWiki supports intertwined design phases, iteration and seamless switching back and forth [In04] [In08].

[In04] *After we came up new ideas, we had boiling, steam power, because it's called a barometer. We had this idea already, implicitly, then we switched back to the brainstorm phase, to put it down. But I think this is necessary to get this idea into mind.*

Some participants are aware of creativity flow during the design process and consciously reflect on their own practices [In04, In08, In13]. All the resources and design processes, ideas, and different design phases are blended into one dimension, where participants can directly interact, reason, and discuss with each other.

2. MikiWiki supports participants in creating and sharing design ideas via different means. Visibility is crucial in externalizing ideas, reasoning about ideas or discussing them. It supports users in coming up with new ideas, since participants have the opportunities of listening and seeing other participants' ideas and therefore being inspired by others.

[In11] *Creativity comes most of the time by being inspired by something, it's a great tool and [it is good to] brainstorm ideas with other people. That can lead to inner creativity.*

An essential aspect of meta-design is to continuously support creativity throughout the whole span between design-for-use and design-in-use to fill the gaps being left by underdesign. During the sessions, it became apparent that MikiWiki provided various features, which supported creativity in design. The combination of MikiWiki with an interactive large wall meets several creativity criteria as they have been published (Lu and Mantei 1991; Resnick et al. 2005). The interactive screen is especially useful to provide the large picture of what has been proposed so that nothing is lost and the various ideas can be flexibly grabbed to generate variations, played around with or become the basis for following ideas (Zhu and Herrmann 2012a). The following features were identified as creativity support:

A Sandbox for Tinkering

One shared reason for appreciation was that MikiWiki acted as a sandbox, that the users could play with, tinker and try things out [In02, In07, In13]. It is important to support participants to explore solutions and "what-if" (Shneiderman 2000; Mamykina et al. 2002) scenarios, trying out assumptions to assess design proposals. One participant [In02] stated: "*It was quite nice that we didn't jump from tool to tool to do different things. Brainstorming feels more like a different tool, starting from a simple GUI. We just tried what we had there to achieve what we wanted. It really felt like a little playground, when you had quite many possibilities. […]*" Using MikiWiki with an interactive large screen can be characterized as a 'sandbox for tinkering', which allows the participants to collaboratively prototype design proposals, try out, evaluate, and eventually discard or use them as a basis for ongoing work. We believe that the perception of the sandbox is supported by the easy reach and availability of a range of small tools and the easiness of designing by selecting, dragging and dropping ready-made design elements.

[In07] *It feels more alive, more real. You feel more like a user… It's a little bit like Christmas or something, you can wish something and it's there.*

[In10] *But also the combination of all nuggets, sketchboard, notes, moodboard, and the icons, gestures... It's the variation that you have. This is great for creating and implementing ideas.*

[In13] *It's probably that MikiWiki takes what we are doing with the combination of the process modeling and brainstorming to a new level, which is "does combing everything with everything?"*

Visualization and Externalization

Participants used different nuggets to externalize ideas, making tacit knowledge imaginable to others.

[In01] *I was more able to express my ideas, because of this powerful tool. For me it's more a problem to "say" what I want. MikiWIki made it much easier to "show" what I want.*

Figure 122 demonstrates that one designer used the sync-imagenote nugget to search images from the web to illustrate his flower menu concept and further used the doodle nugget to sketch his flower gesture concept. Nuggets provided lightweight means to support each participant to externalize ideas, making tacit knowledge embodied and available to others.

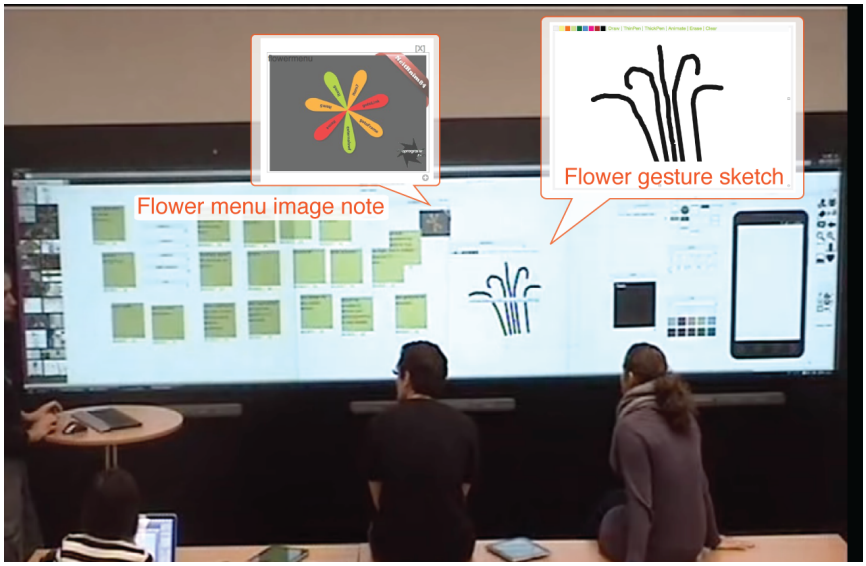


Figure 122 Visualizing and externalizing concepts

Small Tools Fostering Appropriation

The less direct affordances (Wakkary 2009) of each nugget encourage participants in appropriating their usage, and they allow participants to express their meaning in action. Different tools can be used together to achieve new behaviors [In04, In11]. Their existence does not impose any obligation to use them (Fischer et al. 1992). Participants are therefore more engaged and excited about personal meaningful ideas and seek for suitable self-expression (Resnick et al. 2005).

Tools with Simple Combination Interfaces

Small elements and features allow unexpected creative remixing and combinations. In this design case, nuggets are specific enough so that participants can understand how to use them, but general enough so that participants can find new way to appropriate them (Resnick et al. 2005).

[In04] *The good part was the transparent tools* [referring to the doodle nugget]*. We have these layers, you already have these rectangle limitations, but you can drag things over each other. For example you can put this transparent painting area over this mobile canvas and then you can paint on top of it and then you can drag it somewhere else. Maybe you can provide several layers, just compare ideas, …I think this is a nice design…*

[In10] *But you don't really have the possibility to combine with pictures, it's not "what you see is what you get"… MikiWiki is "what you see is what you get" [What you get is what you see], it can visualize your ideas very fast.*

[In11] *Especially, the pictures, there were many things we could use, the way we could combine the boards with each other, and the colors, searching words or even colors…*

[In12] *I have an overview, and at the same time, that's kind of structured and clear. You can be creative, because you have all these tools, you can draw, you can use the signs, you have colors.*

[In13] *The basic features, and the basic look and feel of what I was thinking about were probably captured very well. I managed with anything, the final thing we had there, we had icons, we had different colors, we had some shapes with some text, we had drawings, I think we probably integrated everything you offered to us. [Designer: You didn't have to use everything] Well, I felt natural, I felt good, it was reasonable to integrate all these things.*

Simple Interactions

Simple interaction and intuitive operation allow participants to focus on their design tasks at hand, rather than breaking down their design flow.

[In07] *I think it was good, when we selected ideas, we can only drag and drop to the mobile interface. It's not really a cut between different phases. It's kind of flow because you can take this, drag this and just use it.*

[In11] *At least for the part I used, the front of the screen, dragging and clustering things was very intuitive.*

Low Threshold

One of the key elements for MikiWiki to foster social creativity is a low barrier to entry (Farooq et al. 2007). Lightweight tools that can be accessed by anyone on any platform facilitate the

kind of easy, open sharing and communication that is a key component of creative collaboration. To a certain extent, it encourages socio-emotional communication and play.

[In09] *Everybody is able to use it. Whether you have a technical background or not, you just need to feel free. And do whatever you want. There's no coding. You don't need to know anything, you can just show your ideas. For other applications, you need more time to express you ideas.*

System Trust

In the early experiments MikiWiki nuggets did not support auto saving. Every nugget had to be saved individually, but the system interface did not make obvious which nuggets were related to which saving button. Users would forget saving nuggets or would save the wrong ones and accidents such as closing a browser tab would lead to a loss of the state of the system. This leads to little trust in the system and more focus on how to deal with the system than on the task at hand.

During one of the meta-design phases we removed the save buttons and enabled auto saving after every single action for all the nuggets present in the experiment.

This led to greater trust in the system. When the same participants joined a later experiment and found that auto saving was in place, they interacted more freely with the system.

[In12] *And you have trash, you can trash things, you can develop new ideas, so nothing is limited. It gives you the feeling that everything is possible.*

Continuous Refining Design Spaces

Participants were able to act on their design space and redefine it around their specific situated context. As nuggets are independent and loosely coupled, participants could recombine them to better communicate their ideas, to create either a structured design space [In01] or a more chaotic space on the canvas [In03].

[In01] *I was just lost at beginning, because I don't have a structure, but when I started to make my own structure, I am not lost anymore [….]. I like the idea that I have a digital desktop, where I can move everything around. When I want to move it away, I can do it. When I need something, I get it. Because MikiWiki has a large amount of tools,* [which] *are very powerful to develop "useful" ideas, and because I am also convinced that we can develop these ideas in practice.*

Participants are able to cope with and eventually take advantage of the freedom, creating their own workflow and design space.

[In03] *I think I like the chaos on the "whole" canvas. Sometimes it's getting on your nerves, there's always something in your way when you drag and drop something. But in general, it's good, because you can just drag what you need. It's right there, maybe somewhere, you don't even know where, but it's not that... you don't have to be so straightforward, like going*

through menus in that way. It's more visual. When you go through the menus, try to find the next item you want to add, you have to structure your thoughts. When if it's just visual, you can just drag it, you have to look for it, but it's not like you have to look for it in a special structured way. I think I like that.

A flexible, more opportunistic and less imposing design tool would facilitate creativity as it allows participants to be in charge of their own workflow.

Transforming and Structuring Ideas

Mørch suggests an "*externalized design of software*" approach, a transformational approach for GUI design, since theoretical ideas, concepts and notions provide external elements for designers to build computational artifacts (Mørch 2011b). MikiWiki supports participants to "*transform*" abstract conceptual ideas and connect them, indicating their relationships [In10]. Translating ideas from text to graphics is a process of externalizing implicit knowledge in this case. The implicit or explicit transforming trajectory allows participants to reflect upon later on and articulate previous creative ideas towards a convergent design result [In04].

[In10] *In [transforming] the ideas into icons or other visualization forms, and in posting idea notes into mockup, in combining all the artifacts…*

[In01] *My kind of connection was creating my own kind of structure. Because I knew I had on the left side my notes, what we wanted to do. Then we had this brainstorming tool, this idea by image tool. Later on, I put this also on the left bottom. I knew it was there, so I could scroll up and down, when I wanted to look up something. I always put the nuggets I was using at the moment in the middle. The old ones I put just on the left side, like on the desk, like a typical desk. Moving what I don't need at the moment to the side. That's how I could always look up notes, I could always look on the right hand for tools that I might need and drag them to the middle. This is my kind of connection...*

Tools for creativity support should consider different cognitive styles and reflect different approaches, allowing for structured and linear and free form and parallel explorations (Herrmann 2010).

Providing an Overview

Such an environment provides an overview of all ideas and the big picture they compose [In01, In06, In11]. One of the important criteria for designing heuristic support for collaborative creativity is "supporting the large picture – the visualization of rich material" (Herrmann 2010).

[In03] *I think what was cool is that it has possibility to show all three levels we had during the process. Ideas were still there, I think partly because we have a large screen. We had the ideas on the left side, and kind of image browsing we did in the middle, and when we did our mockups, we still could look at weird…I don't think we really looked into detail what we wrote down, but I could see there were lots of ideas.*

[In07] *From the beginning to the end, I really like that you can really see. It's like you can really see a mobile and you can use your application. […] For me it's not just an idea, it feels like that you can use it to design and you can realize it.*

[In11] *What worked very well was the style in which ideas could be layered over each other, and they also had a transparent background, it was quite easy to cluster them and still be able to read them. It was quite helpful. I liked the clustering notes, and the transparency of these fields, so everything was still readable.*

[In12] *It helped me a lot, because it was very 'anschaulich' [the German for 'visually comprehensible'']. You have everything. You have everything in front of you. I have an overview, and at the same time, that's kind of structured and clear. You can be creative, because you have all these tools, you can draw, you can use the signs, you have colors.*

During the output phase of EXP05, participants created an interface with five different color buttons, different colors indicating a scale of 1-5 points. However, during the review phase, participants stepped back and evaluated their implemented application. Participant 2 pointed out that the application style did not match their moodboard style, i.e. *minimalism, modern classic, limited color scheme.* They eventually came up with silver glossary buttons with numbers on to indicate the scale.

Processing History

Participants could present different, often complementary views on the problem and help each other break away from obvious solutions. During such discussions, ideas were captured, structured and stored via notes, which could be used for later design reference. As such it allowed the participants to build a shared knowledge resource that has various benefits (Shneiderman 2000).

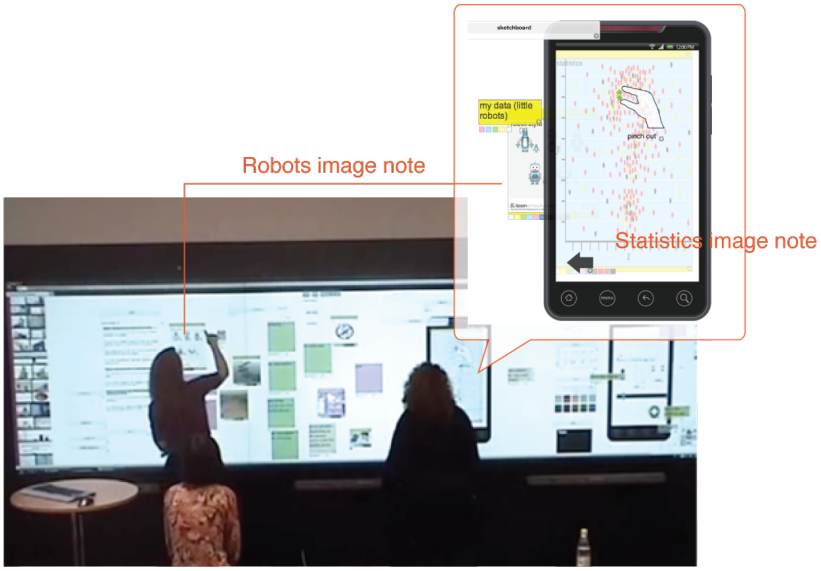


Figure 123 Borrowing design elements from brainstorming stage

An example is illustrated in Figure 123. At the final output stage, two participants started going through all the design requirements that they noted down at the brainstorming stage. When they realized that one of the requirements was how to represent survey results, they two image notes (robotic and statistical) directly into the final output phase.

[In04] *I personally think it's nice. It enables creativity, yes, it allows pace of activity and recombination of phases of incubation, because it stores the last state of the design, and when you think of something, you come to new ideas and you go back to MikiWiki, find the last state is available… and so on.*

[In13] *The other thing is that the pictures from the mood board, all these kind of things, combine together in this interface. And later on, I myself started looking at the initial ideas again, to sort of find out how I captured anything that was written there. So it's some sort of going back and forth between the different representation forms there.*

Rapid Prototyping

MikiWiki supports rapid prototyping and tinkering. As participant 9 stated, "*Instead of discussing each idea, it's better for me to visualize it and to develop something very fast, and to review the solutions.*" [In10] As undo is available, both creating and erasing content could be safely conducted [In09, In12].

[In01] […] *Dragging these icons to the mobile images, if I don't like it, I can get another one - within five minutes, I could have tried out three or five solutions for one problem. That's very efficient. So you have more time to develop more ideas in a short time. Classic brainstorming ideas, you have ideas, you have to cluster them, developing ideas, you are clustering, maybe you have some meta-clustering and you just spent too much time on ideas. With MikiWiki, you are making ideas, and trying them out at once and in real time…. In one hour, we developed four scenarios, which were quite good ideas. We had two or three prototypes, within one hour. With classic methods, you cannot develop in such a fast way.*

[In09] *[…] It's fast, you can directly show your ideas, and improve them. If I have an idea and I show it to another person, and then the other person could say, "Yeah this is good or bad, but I think it would be better…" - the other person can directly show me what he means.*

Perceived Feasibility Breeds Satisfaction

When asked whether participants were happy with their design results, they all expressed very positive opinions about their design [In01, In07, In13]. The main reason is the feasibility of their applications, as they considered that their applications could be implemented, especially with further work. Participant 11 and participant 7 were happy with the design process rather than the final results, since they perceived the final applications were idea collections and were not feasible to be implemented due to technical and time reasons.

Sketching with End Users

All participants thought that MikiWiki could support them in collaborative design with clients, since MikiWiki supports collaboration [In01] [In02], visualizing [In07] [In09] [In12] and structuring ideas [In13]. It enhances shared common understanding [In08], provides general impression of design [In11] and supports sharing design results with larger communities [In04]. One of the participants expressed his concern that he would make sure to be familiar with MikiWiki, so he could answer clients' questions and find the right resources to solve their problems on time [In03].

[In08] *Yes, because every tool to create shared common understanding [can be] very fast, to create some common artifacts that you can discuss.*

Reaching Shared Understanding

Since all the design phases and ideas could be captured on the canvas, participants could refer to them to learn each other's approaches, or to revisit the development of a creative idea without having to explicitly discuss it.

[In09] *The good thing with MikiWiki was to drag buttons like you want, you can individually put them anywhere. We can put them on the left corner, anywhere on the screen, you can show your ideas better to the other person who's working with you. You can just experiment with some ideas and improve them also.*

Generating Momentum

When participants saw a wide range of icons made available by the meta-designer, they were inspired even if the icons were not directly related to their actual ideas. These items acted as a stimulus for coming up with creative requirements. For instance, in EXP03, designers noticed the audio icon, and subsequently had the idea that audio input should be available. They further reasoned on using voice volume to indicate the rating scale. Introducing unexpected and accidental inputs can foster creativity and simulate unconventional thinking. In particular, the sync-imagenote nugget offers easy manipulation with randomness.

[In03] *What was quite good was imagenotes* [the sync-imagenote nugget]*. You could search images from Google. It was mainly for creativity, I think it was cool, because it's just giving you some input, it was not really on the point, focus, and you can say ok, you do this and that. But it's fun to use. Maybe you we should do it before the note brainstorming. It was more to open your mind, you can look for a word, and see what stupid guesses Google does, you never imagine that could relate to your search.*

In the EXP02, words came across designers' mind on the fly or from the clustered notes that they wrote down before, as they tried to search for associated images. Google search generated randomized images inspired them to come up with more ideas and added more improvised actions and thinking. For instance, when they saw a yellow smiley face with a pair

of sunglass, designer 4 immediately said, “oh, this is nice, we want fun and we want some sun…”

[In13] *I think it also helped quite a lot that we were able to search for these pictures. Well, I don’t feel them being integrated into the design as they were. But I think they guide sort of my thinking process about how this interface looks like.*

[In13] *When I started, it was quite difficult for me. Then I came across the shape thing, then I came to this Reventon Roadster car…We started by putting adverbs, simple, minimalism all this kind of stuff… this was okay, but up to that point, where we found this furniture, this car thing, I didn’t feel we had captured some meaning of these words* (Figure 124). *So the pictures helped me to gain or to reflect upon what we have found there so far, then I came across this car thing, which, I don’t know normally should be very natural to me, because I am very addicted to cars. This is for me the only art form I really enjoy.*

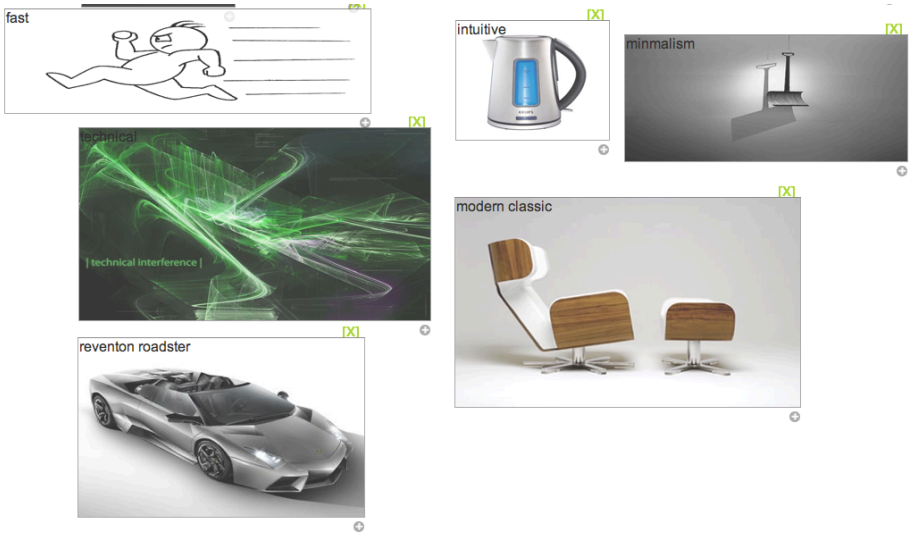


Figure 124 Moodboard created in EXP05

Convergent Flow

All participants had a similar approach to the design solutions. They came up with abstract concepts, visualized them and refined them in the end to disseminate their design results. MikiWiki certainly supports this convergent process.

[In07] *The power of your tool is that I feel inspired to work and to go on. With other tools, you just collect ideas, it’s hard to imagine, and you don’t have a creativity flow like this. You collect ideas, but you are not so much motivated to make something with this collection.*

Another observation is that creating textual notes or image notes for bringing ideas on the screen is a divergent thinking process, while clustering them in different categories with different colors is a convergent thinking process. Different design phases with different nuggets enable participants to articulate creative ideas as well as refine them further in more detail.

MikiWiki supports participants experimenting with different alternatives. The evaluation process can be augmented through comparing different solutions, defining common ground, and optimizing various design solutions. All these are crucial for creativity and incremental evolving design ideas as well as design process. With MikiWiki "*one can show ideas, another person doesn't have to agree with it, the other person can add another device, compare different ideas, and connect them somehow*." **[In09]**

Possibilities for Other Use Cases

MikiWiki provides a collaborative design environment for a broad spectrum of application areas, for instance iterative prototyping interactive system design with a focus on evolutionary participatory design. MikiWiki could be used to rapidly prototype new UI designs and bring different design teams together. It is a web-based platform, allowing design results to be easily stored and shared by communities. The wide design corridor, which is opened by MikiWiki, became obvious by the way participants used it and how their design focus was broadened and enriched. **[In09]** *"If you are not set to the mobile device, you can use it for brainstorming or mind mapping somehow, you don't always have to design some applications for a mobile device."*

One shortcoming of the environment that emerged from the interviews is that there was no private space where participants could draft ideas in isolation, without being observed by others – as required by (Lu and Mantei 1991). Currently MikiWiki does not support the differentiation between various layers which can be assigned to certain participants of design aspects and can be easily hidden or shown, although it could be extended on the client side to do so.

9.9 Critical Issues

Despite the largely positive results, this evaluation had several limitations. The meta-designer guided participants using MikiWiki on the fly, and therefore it is open to debate if users and designers can use it without guidance and tutoring. Evolving MikiWiki from a system level, however, was done solely by the meta-designer in this study. As such it is difficult to validate whether novices would benefit and be willing to tinker with the meta-design level.

Ongoing empirical investigation and clarification of the meta-design concept should take a whole series of design cycles into account, including phases of asynchronous and dislocated collaboration. Furthermore, a longer time period should be taken into account where design outcomes are used and adapted during use.

Nevertheless, it appeared reasonable to start with short cycle experiments to get an immediate feedback on:

1) The scope to adapt MikiWiki or increase its adaptability.

2) The characteristics of the socio-technical context into which MikiWiki has to be embedded.
3) The kind of explanations and interventions that have to be provided by the meta-designer.
4) Characteristics of the design task and of the involved participants.

The chosen setting is a reasonable basis to go on with the empirical investigation of meta-design. Computational environments should be designed intentionally to support different levels of participation and multimodal externalization possibilities in order to overcome the disadvantages inherent in collaborative processes between people of different backgrounds as well as at different design stages (Zhu and Herrmann 2012a), as highlighted in the HMS model.

9.10 Conclusions

This chapter described how MikiWiki was used in the Creativity Barometer project. The empirical evaluation of co-located MikiWiki sessions and of the underlying principles of the HMS model reveals that meta-design is not only an abstract concept but can be instantiated in real settings. Five design sessions demonstrated that MikiWiki not only could be quickly adapted to cope with emergent social-technical issues but also supported participants' creativity, in particular lightweight tools (nuggets) to be used and appropriated, rapidly exploring different scenarios, and externalizing and visualizing abstract ideas via multi-modes. It is important to involve different types of participants to understand whether MikiWiki support collaboration among people with different roles and backgrounds.

The adaptability of the design-environment to cope with emergent social-technical issues is the most central characteristic of meta-design and was achieved by client-side scripting and flexibly combining small components. On the other hand, the effectiveness of this model is not merely a technical issue but also relies on the whole socio-technical context - i.e.: the influence of a facilitator, who has to encourage the participants to sketch their ideas, and to get them initially used to employing a variety of the meta-design features available. Further influential factors are the duration of sessions, their cyclical repetition, the appropriate mix of participants with respect to their abilities and experiences, and the characteristics of the design task. The facilitator must be able to act as a meta-designer who can instantaneously add new features to the design-environment or modify its features (Zhu and Herrmann 2012a).

Chapter 10

This chapter provides a summary of this thesis, reviews my contributions to collaborative design research, reflects upon the work I have accomplished, including its limitations, and outlines possible areas for future exploration and development.

10. Conclusions and Future Work

This thesis comprises four phases according to the nature of the work accomplished (Section 3.2).

Chapter 1 and Chapter 2 provide the theoretical background and the state of the art relevant to this research in order to bring out synergy from different perspectives. Chapter 3 introduces the methodologies used to conduct preliminary studies, design the research, and carry it out. They shaped my understanding of design in use and provided valuable insights into real-world collaboration.

Chapter 4 presents my meta-design conceptual model, the hive-mind space model. It frames out guidelines for designing socio-technical systems to support collaborative design and social creativity.

Chapter 5 and Chapter 6 explain how the HMS is implemented, how its features are reflected in the prototype, MikiWiki, and how meta-design boundary objects are implemented technically.

Chapter 7 and Chapter 8 document how MikiWiki was used in two distinct collaborative design cases. They also include my reflections on the HMS model in response to the results of analyzing collaborative design processes, then improving MikiWiki accordingly. Evaluation of support for creativity and collaborative design is addressed specifically and documented in Chapter 9.

10.1 In a Nutshell

The goal of this research, "*to explore meta-design approaches for cultivating and supporting collaborative design*", was achieved through several steps that trace through the thesis chapters. The whole research process was followed by a series of publications (see Appendix C).

As stated in Section 1.3, my research explored the following question:

> *How do we provide a socio-technical environment to bring multidisciplinary design communities together to foster creativity, collaboration, and design evolution?*

This question is refined into the following sub-questions:

- What essential features of a socio-technical system support cultures of participation and foster social creativity?
- How can we support evolving design activities, in diverse communities, through the appropriation of design artifacts?

Sub Question 1

The HMS model attempts to answer this question by providing guidelines for designing socio-technical systems that allow social spaces to be reconfigured and support the meta-design of the communication medium.

The conceptual model frames a set of principles that define desired functionality (Section 4.2), as well as a reference architecture (section 6.1.3), and provide guidelines for implementing meta-design systems:

- Habitable environments – environments for design teams to perform design activities;
- HMS Boundary objects – means for supporting communication and collaboration, that shape social interaction and are reshaped by it;
- Boundary zone – (communication channel) a collaborative process of learning, exploring and evolving;
- Mediation mechanism – a mechanism to situatedly tailor and create personalized environments;
- Different levels of participating/tailoring – to support participation, meta-design, and EUD. Design communities are in charge of creating different environments to carry out meta-design, design, and use activities;
- Open infrastructure – a more transparent system to allow inspection and to connect to external resources and communities;
- SER model – to provide opportunities for continuous development of the system at the time of use.

These design principles are intertwined with and reciprocal to each other.

Sub Question 2

Within the HMS model, meta-designed boundary objects, its building blocks, seek to answer this second question by supporting design communities' collaborative design-in-use and overall practices. They are the means to interact with, compose, and evolve the socio-technical system through practice. However, they are also the means to support evolving design practices.

In this research, HMS boundary objects were exploited to:

1) Empower users to tailor and evolve their environments (environments that are themselves composed of tailorable boundary objects) (Section 8.5.2);

2) Allow users, over time, to structure their own tools and services (Section 7.5.5);

3) Be used, appropriated, cloned, and mixed, and to act as inspiration for creating new boundary objects (Section 7.5.2, Section 8.5.6);

4) Break down fixed roles and overcome strict levels of participation by allowing all users to access the meta-design, the design, and the use levels (Section 7.5.3).

Because the whole system may evolve through the evolutionary growth of its component boundary objects, which also make up its communication infrastructure, the evolution of HMS boundary objects may, in practice, permit communication and social policies to be reshaped.

10.2 Contributions

EUD explores tools and methods to allow end users to tailor software artifacts but addresses this objective primarily from a technical perspective. Still, more complex social aspects need to be taken into consideration. Meta-design is an appropriate approach for future EUD, because it aims to create socio-technical environments that empower users to actively engage in a continuous system-development process rather than to be passive users. At design time, it would be unrealistic to build software systems that meet as-yet-unknown social-technical challenges. Such challenges will emerge only at use time, through social interaction and a continuously changing context. Therefore, far from being a luxury, meta-design is a necessity.

This thesis, centered on meta-design, extends EUD by creating the social conditions and the design processes required for ongoing, broad participation in design collaboration. It addresses some shortcomings of meta-design, especially the absence of guidelines for building concrete, empirically assessable meta-design environments. The research explores incorporating meta-design principles into practice, as well as speeding up ongoing collaborative design and making system development more agile.

The contributions of this research to meta-design are:

(1) **The *Hive-Mind Space model***, a meta-design framework derived from SSW methodology and integrating the SER process model.

The HMS model derives from SSW methodology but focuses on enhancing creativity. It extends SSW methodology with a bottom-up approach that integrates means to encourage design communities to tinker and to appropriate those means so as to cope with emerging design problems, thus enhancing overall creativity. The mediation mechanism removes the need for explicit representation of role, of culture or of device, since it is socially based. A detailed comparison can be found in Section 4.3. The HMS model integrates the SER process that underlies the underdesign principle.

The HMS model provides guidelines for interactive design systems and stresses openness, flexibility, and accommodating emergent socio-technical issues at time of use. It focuses on enabling meta-design and supporting EUD, thus empowering users to be designers with different granularity and fostering their creativity. Different levels of participation can coexist within the same environment; thus users can seamlessly move between them, achieving higher levels of system tailorability. The model emphasizes bridging communication gaps through exchange and tinkering with boundary objects.

By encouraging diversity and user-driven innovation, it enhances social creativity. To this end, the model calls for an open evolvable infrastructure rather than fixed solutions to bring heterogeneous design teams together, as well as to involve them in the design process. This affords independence, since the architecture of the HMS model is globally interconnected and locally controlled. It frames a decentralized CoI, allowing design communities to specialize and draw on local knowledge. It also provides a malleable mediation mechanism that design communities can adapt to their situated requirements, while enabling knowledge aggregation, combining individual contributions into a shared collection.

Differentiating design levels, design roles, design processes and creativity mandates that instruments of varying granularity be supplied to better support collaboration tailorable at the individual level. On the other hand, my research explored ways to blend these differences, both to encourage tinkering and to foster richer ecologies of participation.

(2) ***MikiWiki***

To demonstrate the feasibility of implementing the HMS model, MikiWiki combines the functionalities of traditional wikis with EUD practices and meta-design concepts, within the HMS conceptual framework. It seeks to translate meta-design from an abstract idea into a practical platform that benefits users. MikiWiki is a meta-design system, empowering end users with seeds for creating knowledge, managing knowledge, and allowing it to evolve. A collaborative design process is therefore achieved by using artifacts (seeds) that support creative exploration. Specifically, MikiWiki exploits the contributions of web 2.0 and social media as innovative information technologies. The case-based incremental implementation is

documented and discussed in Chapter 7, Chapter 8, and Chapter 9. This shows MikiWiki's technical and social strengths and weaknesses as a meta-design environment.

In addition, MikiWiki extends the traditional unstructured wiki into a structured programmable wiki that supports client-side open implementation. MikiWiki thus supports not only general collaboration but also evolution at use time in the stakeholders' hands.

(3) ***Empirical insights***

This research has provided empirical insight into the meta-design model and into applying it to different domains. Design studies have shown how MikiWiki complies with the HMS model and how its features benefit collaborative design.

On the basis of various cases of applying MikiWiki, collaborative design studies have provided evidence that different phases of meta-design represent different modes rather than discrete levels.

Design studies have shown that, over time, the boundaries between design roles became blurred (Section 7.5.3). As demonstrated in case-based implementations and empirical studies, MikiWiki has evolved from a basic wiki into a comprehensive platform to support collaboration. There is no distinction between design time and time of use. Evolutionary growth emerges from (re)seeding (Section 9.7), due to the possibility for continuous meta-design, especially on the client side, from within the system itself. The initial system is made up of small components, each acting as a seed that grows over time.

On the other hand, small standalone components tackle the traditional CBSD dilemma (Section 2.4.4, Section 5.2.2). One distinctive feature of nuggets is that they are independent of one another, rather than integrated with each other by means of input/output ports as in most CBSD systems. Nuggets extend application units towards web-application environments. As social application units, nuggets are distributed and can be modified, as well as evolving through collaborative interaction. To this end, CBSD could benefit from interaction with other components as part of the bricolage approach, i.e. combining the conventional and bricolage approaches.

My research has explored various levels of creativity, namely the meta-design level, the level of opportunistic programming as bricolage, or situated creativity (Section 8.5.2), the design and use level, and the level of appropriation focuses on creativity in use (Section 7.5.2, Section, 8.5.2). Embodying and refining creativity calls for the system to have been designed to foster various types of creativity. Empirical results of my research have provided examples of reciprocal social creativity and collaborative tinkering enabled by meta-design. These findings confirm that it is possible to attain a smooth transition between design for use and design in use through continuously incremental meta-design (Section 7.5.5, Section 8.5.6). All users appreciated open code and open data, although to some extent they were also given the impression of a rather technical approach (Section 8.5.3). On the other hand, results

suggest that a more radical approach to encouraging participation and enhancing understanding (especially from a technical perspective) could be to make code and data accessible.

Furthermore, the boundary zone is not a software artifact but rather a process of learning, creating and evolution that encourages creativity (Section 8.5.4). Through interaction, meta-designers understand the needs of users and, therefore, provide better support, while users gain understanding of how to use the system and ability to explore its potential. During the learning process, users become more advanced and are then able to carry out certain meta-design activities by themselves.

With this research, I have brought out a series of insights into collaborative design based on technical, theoretical, and practical investigation. Following upon these, I have proposed and demonstrated an approach to collaborative design focused on EUD and meta-design. In doing so, I have presented not only relevant concepts, but also mechanisms for concretizing them and enabling meta-design in practice.

10.3 Limitations and Suggestions for Improvements

My implementation of MikiWiki focused on demonstrating open architecture HMS and the potential of design in use. Meta-design activities are still difficult for most designers and users, due to time constraints and learning requirements. In order to further flatten the learning curve, more examples, comprehensive documentation, and templates should be provided. Building intermediate levels to help move users up to higher levels will, in turn, foster richer ecologies of participation.

Although MikiWiki supports three levels of tailoring and despite designers' success in designing environments and modifying software, the creation of new software artifacts was left to meta-designers and participants with scripting knowledge. End-user scripting still takes a long time and imposes a steep learning curve. Meta-design activities require meta-designers, designers, and end-users to collaboratively explore, learn, and discover problems so as to devise situated solutions.

Note that MikiWiki is merely one possible instance of the HMS model. Therefore, an implementation of the HMS model need not be constrained to page-based structuring nor tied down to a given wiki model. Other possible implementations are discussed in Chapter 5.

My design studies were carried out with small groups of users in which each individual could be seen as representative of a design community. However, once a greater number of participants and larger design communities are involved, communicating and collaborating will become more complex. Another limitation concerns the mediation mechanism, which was not tested across language or culture barriers but only as a mechanism to enable users to create special environments and establish their own different modes of interaction.

The HMS model might be improved in the future through further case-based implementation and empirical study. Future long-term design studies may help assess the principles of the HMS model in action. Consequently, existing concepts may be refined and new ones might be added. Specifically, the boundary zone, both as a process and as an abstract concept, should be further explored.

10.4 Future Work and Opportunities

My future research will continue to apply MikiWiki to various collaborative design cases, enhance it, and improve its usability. Areas for further exploration include the following.

Creating 'Talk-back' Situations

The meta-design and design complexity could be reduced first by providing more comprehensive MikiWiki documentation, and then with additional contextual support, such as in-line reference documentation, auto-completion, and situated recommendations.

Because the design process can be retrieved and inspected from historical data, exploring differing visualizations of the whole design process would allow users to reflect on the collaborative design process. This, in turn, would enhance mutual understanding and increase users' awareness.

Security Model

Some JavaScript security issues should be addressed by investigating a security model. For instance, keeping track of who modified the nugget code, notifying other users, and migrating the modification with the initial code are technical, as well as social, issues.

From an implementation aspect, server-side code can be written in JavaScript via utilizing node.js (Node.js 2009). Having a single language for both the server side and the client side allows code to be reused between browser and server implementations, obviating the need to map APIs between different languages.

Balancing Personalization and Socialization

My research has explored how boundary objects influence the behavior of whole environments. It would be interesting to reintroduce some level of personal tailoring by giving users the chance to own a few 'personal' boundary objects attached to their profiles and then to follow them as they move between environments. An automatic translator, a status message, or even a pass to enter a specific environment could all be personal boundary objects, enhancing users' experience.

On the other hand, socialized code development might also be investigated. For example, a group of meta-designers could work in parallel on heterogeneous projects in MikiWiki and use MikiWiki as a hub to share, refer to, and execute their code. Issues such as versioning, change management, and code distribution and evolution merit further investigation.

Design Now!

A new design space is created by meta-design concepts and the technical validity demonstrated in this research. Design problems are solved without scripted plans or preconceptions of how to proceed. Therefore, the decision-making process is situational – that is, testing and creating on the spot. The time dimension is compressed, reducing it from several connected time spans down to simultaneous, moment-to-moment decisions. This immediacy is achieved by blending collaborative design roles, levels, and processes, which is a sine qua non for stakeholder involvement and creativity. “Design Now!” highlights the situatedness where CoPs dive into continuous interplay between drafting software solutions, on the one hand, and understanding their needs and their performance expectations for design tasks, on the other.

Further investigation of this “Design Now!” concept calls for the relevant aspects to be addressed. One such issue centers on mechanisms for seamless transition between layers of participation, between social incentives, and between processes and architecture that encourage exploration, learning, and tweaking.

Embodying meta-design concepts in the creation of artifacts and in different research domains promises to yield immense opportunities, offer sustainable services, and build innovative software systems. From here, I look forward to further exploring the field and to meeting new, like-minded people.

The reality of complexification is both an is and an ought: it happened – given the conditions ruling the earth, it was bound to happen – but it might not continue unless we wish it to go on. The future of evolution is now in our hands.

- (Csikszentmihalyi 1990)

Glossary

Application Unit (2.2.4)

Application units are defined as the smallest self-contained units to be useful in the design and implementation of end-user tailorable applications, such as word processors, drawing programs, and e-mail systems.

Appropriation (2.4.3)

Appropriation is the process by which users adopt and adapt technologies to fit them into their work practices.

Boundary objects (2.3)

Boundary objects are shared artifacts, which can be utilized to mediate communication gaps. Boundary objects are both plastic enough to adapt to local needs and constraints of the server parties employing them, yet robust enough to maintain a common identity across sites.

Boundary objects: HMS (4.2.3)

In the HMS model boundary objects are not only the content of an interaction, but can form the very medium that enables and directs communication. The HMS boundary objects are artifacts that can be continuously modified, discussed and socially negotiated. Every communication in the HMS is mediated by boundary objects. The whole set of the boundary objects forms the HMS infrastructure for communication, interaction and partitioning of space as well as being a shared knowledge base and history of all interactions.

Bricolage (2.4.4)

Bricolage is a method of expression through the selection and synthesis of components obtained from surrounding culture.

Component-Based Software Development (2.2.4)

Component-Based Software Development (CBSD) involves multiple roles. Framework builders create the infrastructure for components to interact; developers identify suitable domains and develop new components for them; application assemblers select domain-specific components and assemble them into applications; and end users employ component-based applications to perform daily tasks.

Software component (2.2.4)

A software component is a physical packaging of executable software with a well-defined and published interface.

Co-evolution (1.2.2)

From a design perspective, co-evolution means that problem space and solution space co-evolve. From a software development perspective, using the system changes the users, and as they change they will use the system in new ways. In addition, the evolving design problems that cannot be predicted at design time require systems that have enough flexibility and tailorability to cope with emergent unexpected requirements.

Collaborative design (2.1.3)

Collaborative design in this research is two-fold, collaboratively designing the software system itself and in parallel using it to conduct collaborative design projects, e.g. prototyping web applications or collaborative writing. It is an iterative and ongoing process as well as a co-learning process.

Community of Interest (1.1.2)

A Community of Interest (CoI) brings together different CoPs to solve a problem - for instance, a team of domain experts, HCI experts and software engineers interested in software development.

Community of Practice (1.1.2)

Communities of Practice (CoPs) are groups of people who share a professional practice and a professional interest - for example, architects, urban planners, research groups, software developers and end-users.

Design Now! (10.5)

Design Now! pursues an approach that emphasizes supporting situated action to allow users to solve their problems on the spot via social interaction, system appropriation and meta-system bricolage. *Design Now!* requires a flexible medium that provides immediacy of access to the different layers of participation and flexibility in modifying the medium and its tools.

End-User Development (2.2.3)

End-user Development (EUD) is a set of methods, techniques, and tools that allow people who are non-professional software developers at some point to create, modify or extend a software artifact.

Hive-Mind (1.1.1, 4.2)

In analogy with self-organized systems such as ant colonies, a hive mind implies a bottom-up system, which does not have a clearly defined hierarchy, and follows the rules of organized complexity that could lead to emerging structure and collective intelligence greater than the sum of each individual. In this case, the hive mind not only implies cultures of participation but also ad-hoc activities.

Hive-Mind Space (Chapter 4)

Hive-Mind Space (HMS) model is a meta-design conceptual model, to support collaborative design, which not only empowers end-users to design-in-use, but also addresses social creativity. It aims to translate meta-design concepts from an abstract idea into a practical platform by providing concrete and executable guidelines for designing socio-technical systems.

Meta-design (2.2.5)

Meta-design is "*design for designers*". It is a new design paradigm, which allows various stakeholders, including end users, to act as co-designers even at use time. According to this paradigm, software engineers do not design the final application, as in traditional design, but they create software environments through which different stakeholders can contribute to the design of the final application.

MikiWiki (Chapter 5)

'MikiWiki' stands for 'meta-wiki' and indicates the reflective properties of the system, e.g. adaptable to different situations and being able to evolve. MikiWiki is a structured programmable, meta-reflective wiki to operationalize the HMS model. It provides a common collaboration context across the system and opportunities for design communities to build domain-oriented environments where they can work while being aware of the activities of others. Beyond providing tools for text content production as traditional wikis, MikiWiki allows all the stakeholders to collaborate in practice design and to continuously evolve the whole wiki system.

Nuggets (5.2)

In analogy to Lego construction kits, providing simple parts with which users can create complex artifacts, nuggets are the building blocks of MikiWiki, independent from each other and can be used to create new tools or services.

SER, Seeding, Evolutionary growth, Reseeding (2.2.5.1)

The Seeding, Evolutionary growth, Reseeding (SER) model is a conceptual process model that describes the development of systems and information repositories that evolve over time. This process model aids in designing complex and open-ended systems.

SSW, Software Shaping Workshop (2.2.5.2)

The Software Shaping Workshop (SSW) is a methodology to support collaborative evolutionary design of interactive systems that support end-users to become designers of their tools.

Abbreviations

BPM, business process model

CBSD, Component-Based Software Development

CoI, Community of Interest

CoP, Community of Practice

CRUD, Create, Read, Update, Delete

CSCD, Computer Supported Collaborative Design

CSCW, Computer Supported Cooperative Work

CSLab, Computer Semiotics Laboratory

CSS, Cascading Style Sheets

EDC, Envisionment and Discovery Collaboratory

EUD, End-User Development

HCI, Human Computer Interaction

HMS, Hive-Mind Space

MikiML, MikiWiki markup language

IDE, Integrated Development Environment

L3D, Center for Lifelong Learning & Design

NSF, National Science Foundation

RI, Recommended Information

SER, Seeding, Evolutionary Growth, Reseeding

SSW, Software Shaping Workshop

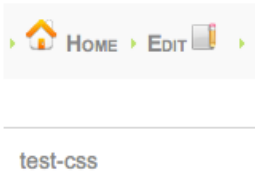
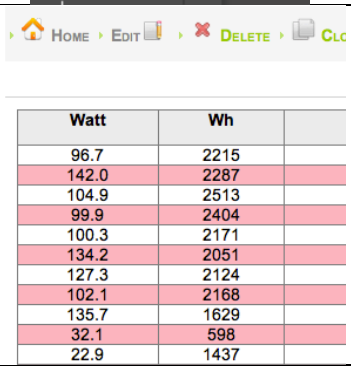
UGC, User Generated Content

URI, Universal Resource Identifier

Appendix A - Nuggets

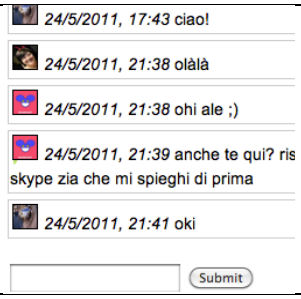
System nuggets

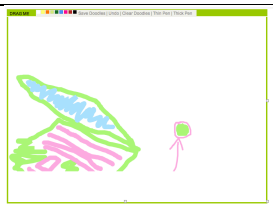
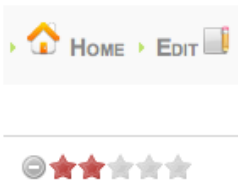
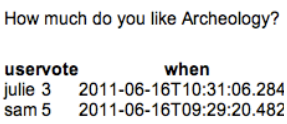
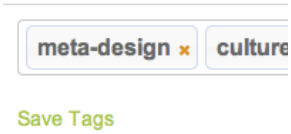
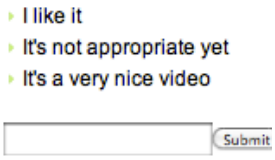
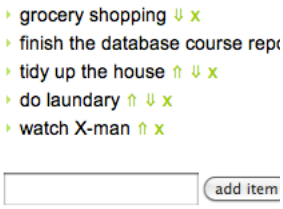
Syntax to use system nuggets is double square brackets "**[[ ]]**". System nuggets should not be modified.

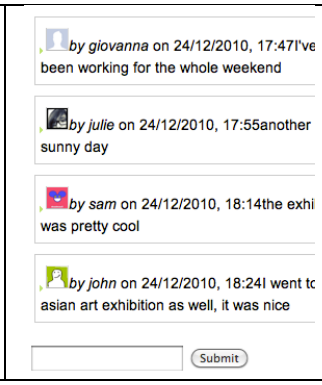
nuggets name	view	syntax	notes
link		[[test-css]]	To link a mikiwiki page [[**css**|test-css]] allows one to give the document a new name; e.g. giving the test-css file a new name "**css**", via specifying a new name before the "**|**" pipe symbol.
expand		[[expand:_help]]	This is a link to a mikiwiki page. Clicking **[+]** sign opens up a page within the current page, instead of redirecting.
include		[[include:demos/twitterWatt/beta/test-twitterEnergy]]	To include a mikiwiki page in the current page **test-twitterEnergy** is the page that is used to be included and "**demos/twitterWatt/beta/test-twitterEnergy**" is its page path.

Typical nuggets

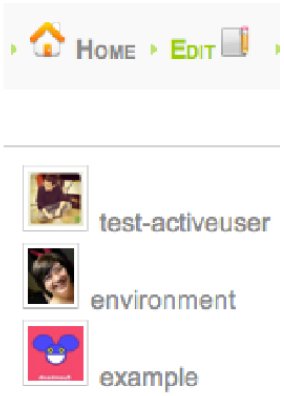
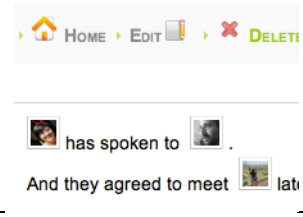
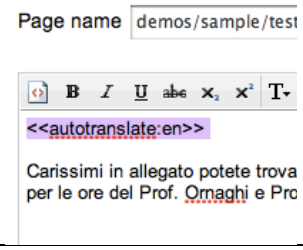
Syntax to use the rest nuggets is double angular brackets "**<< >>**".

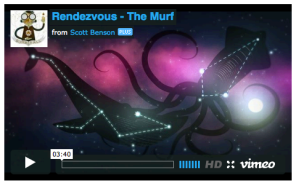
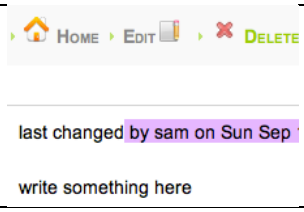
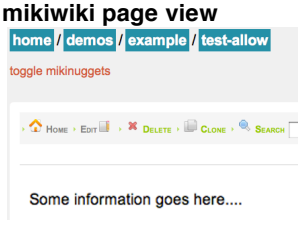
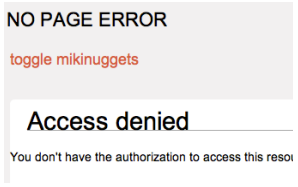
nuggets name	view	syntax	notes
chat		<<chat001 as chat>>	The chat nugget allows one to chat with others in real time. **chat001** is the chat data page. **chat** is the nugget's name

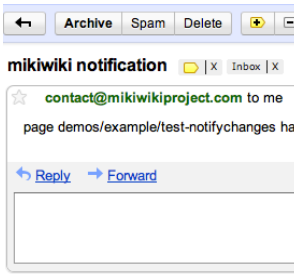
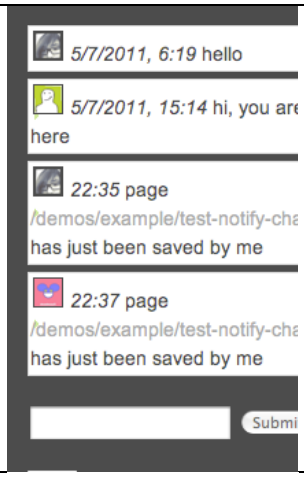
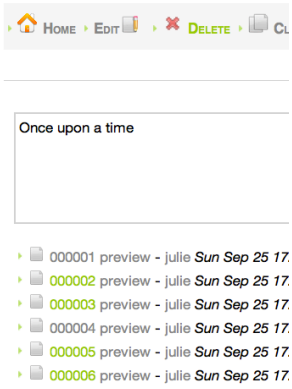
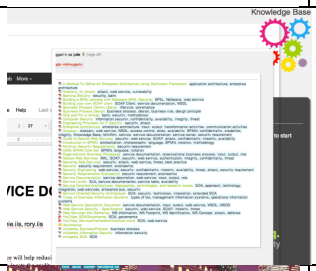
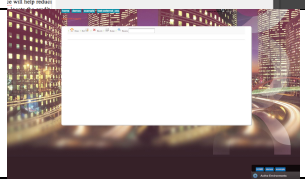
doodle		<<doodle001 as doodle>>	The doodle nugget allows one to sketch things. **doodle001** is the doodle data page. **doodle** is the nugget's name.
rank		<<rank001 as rank>>	The rank nugget allows one to rank individual things. **rank001** is the rank data page. **rank** is the nugget's name, one of the initial nuggets.
vote		<<vote001 as vote>>	The vote nugget is similar to the rank nugget, but it allows a group of people to rank certain things, and gives the final **mean value**. **vote001** is the vote data page. **vote** is the nugget's name
opinion		<<opinion001 as opinion>>	The **opinion** nugget provides more detailed information about "vote", e.g. name, score, time. **opinion001** is the opinion's data page. **opinion** is the nugget's name,
tag		<<tag001 as tag>>	The tag nugget allows one to create tags. **tag001** is the tag data page. **tag** is the nugget's name.
comment		<<comment001 as comment>>	The comment nugget allows one to leave comments. **comment001** is the comment data page, which stores comments **comment** is the nugget's name.
todo		<<todo001 as todo>>	The todo nugget allows one to create a simple task list. **todo001** is the todo data page, which can be a list of tasks or a shopping list **todo** is the nugget's name. Using ⇑ ⇓ to rearrange tasks, or x to delete a task

wall		<<wall001 as wall>>	This nugget is like a Facebook wall. Users can leave comments for other users to see. Each comment is identified by author and date. **wall001** is the wall data page, **wall** is the nugget's name

There are certain nuggets that don't need a data page. If this is the case, then one normally (not always) needs to specify/pass extra parameters.

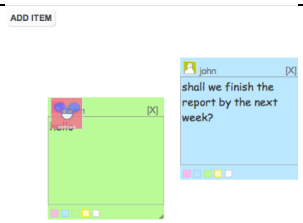
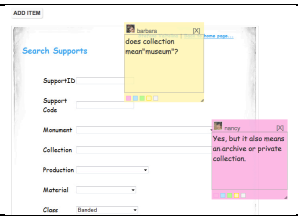
nuggets name	view	syntax	notes
activeuser		<<activeuser>>	The activeuser nugget shows online users, which page they are working on, and their profile pictures according to the parameter one specifies. <<activeuser>> - name, pic, pages <<activeuser:**name**>> - show name only <<activeuser:**page**>> - show only name+page <<activeuser:**pic**>> - show only thumbnail <<activeuser:**who-is-here**>> - show pics of people on this page
profile		<<profile:barbara>>	This nugget allows one to see a profile of a user. One can use profile picture as a way of "visual" writing. <<profile:**username**>>
autotranslate		<<autotranslate:en>>	The autotranslate nugget allows one to translate a mikiwiki page to another language. One can specify the language code after the semicolons. e.g. <<autotranslate:zh>> to translate a piece of text to Chinese.
translation-menu	Italian  Arabic	<<translation-menu>>	The translation-menu nugget provides a language drop-down menu, so one can choose a certain language that he wants the source to be translated to. This example here shows that a piece of text is translated to Arabic. In addition the whole mikiwiki page layout is adapted to the Arabic "right to left reading" habit.

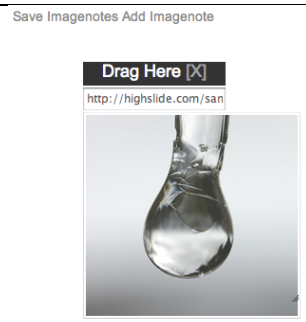
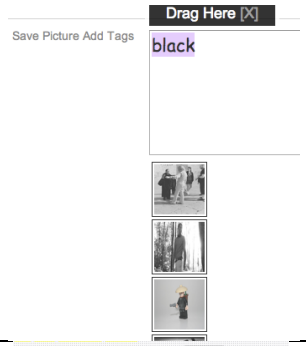
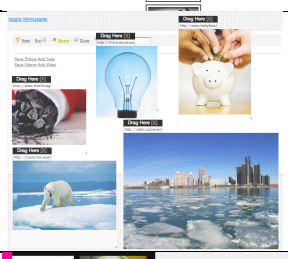
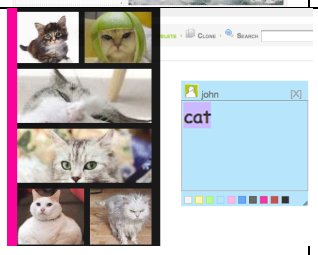
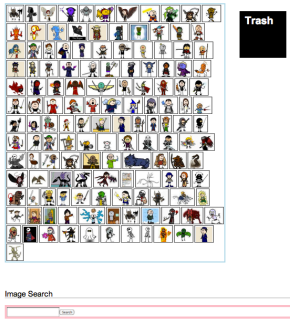
			
video	 	<<video:http://www.youtube.com/watch?v=r9LelXa3U_I&feature=relmfu>> <<video:http://vimeo.com/25584378>>	The video nugget allows one to embed videos from youtube and vimeo. Start with the **video** tag, then pass the video's URL link after the colons. <<**video:video's URL link**>>
file		<<file:http://ambient.media.mit.edu/transitive/ubicomp07papers/sadi.pdf>>	The file nugget allows one to embed external powerpoint or pdf files. Start with **file** first, then after the colons pass the file URL. <<**file:file's URL link**>>
lasteditor		<<lasteditor>>	The lasteditor nugget shows who is the last person to edit a certain page and when.
allow	**Barbara's test-allow mikiwiki page view**  **Sam's test-allow mikiwiki page view** 	<<allow:barbara, john>> Some information goes here.... <<allow:designer>>	The allow nugget lets one share information with certain people or people with certain role. <<**allow:user's name**>> In this example, **barbara** and **john** will be able to see the test-allow mikiwiki page, but since sam was not specified in the list, his access will be denied. Similarly, one can try <<allow:designers>>, so all the designers will be able to access the information, while people listed as engineers won't be able to access the information. <<**allow:user's role**>>

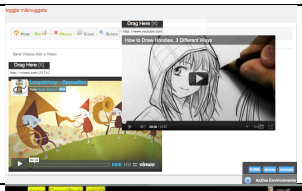
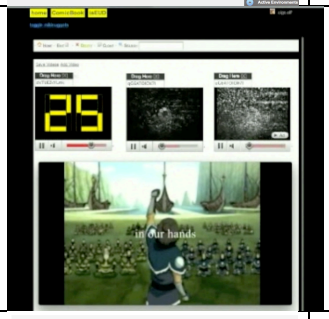
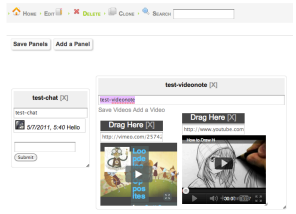
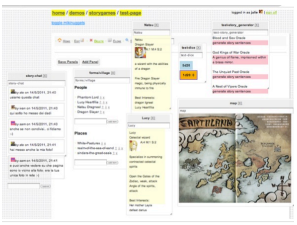
notify		<<notify:julie>>	The notify nugget lets one inform others via email when one edits any MikiWiki page. Start with **notify**, then after the colons "**:**" specify person's name to be informed <<**notify:user's name**>>. In this example, whenever the test-notify page gets modified, **julie** will get an email alert.
notify-chat		<<notify-chat:demos/data/chat002>>	The **notify-chat** nugget allows one to send modification information to the chat. In this case, people who are online will see the modification information via chat. Specify the **chat** data page name that is used to notify changes. In this case, it's the chat002 and its full path is "**demos/data/chat002**" The last two chat entries show that the **test-notify-chat** page has been modified.
history		<<history:demos/Directory/anotherpage>>	The history nugget allows one to inspect a mikiwiki page's history Start with **history**, then after the colons "**:**" specify the mikiwiki page to be inspected <<**history:file's_path**>>, in this case the file path is **demos/Directory/anotherpage** In this example, **anotherpage** has been changed 6 times.
tooltip		<<tooltip>>	The tooltip nugget allows one to preview information about the item being hovered over without clicking it.
css		<<css:http://pippatoledoshop.com/css.css>>	The css nugget allows one to style a mikiwiki page or a whole environment. <<**css:website CSS URL link**>>

website		<<website:http://www.thebestdesigns.com/>>	The website nugget allows one to include external websites for reference, research or sharing purposes. Simply pass the website URL that one want to embed after "**website:**". <<**website:website URL link**>>

More nuggets and use cases

nuggets name	view	syntax	notes
note		<<note001 as note>>	note nugget allows one to create PostIt notes. **note001** is the notes' data page. It has all the notes' data, (position, color, text), expressed in JSON format. [height":"96px","width":"159px","color":"rgb(255,255,0)","x":"478px","y":"109.1px","text":"Naruto"}]
			This is an example to show how one can use a calendar image and a note nugget to create a project schedule.
sync-note		<<sync-note001 as sync-note>>	sync-note nugget allows one to add notes and annotate in real time. sync-note also provides information about the author, who created the note. And if someone works on a note, the profile picture icon gets enlarged.
			The sync-note nugget was used in a workshop to support archaeologists and software developers discussing a live website.

imagenote		<<imagenote001 as imagenote>>	The imagenote nugget can be seen as a graphic note nugget. One can also resize your imagenote by dragging the handler at the right bottom of the note.
flickr		<<flickr001 as flickr>>	This nugget is similar to the imagenote, but the images are from flickr. **flickr001** is the datapage to store all the pictures. **flickr** is the nugget name. One can not resize your pictures in this case however.
			This is an example of how the imagenote nugget was used by students for an energy saving project. They used this nugget to explore an effective way to increase energy-use awareness.
sync-imagenote	 The result of choosing the  first cat	<<sync-imagenote001 as sync-imagenote>>	The sync-imagenote nugget not only turns text/tags to images from Google, but also allows one to write text in the notes. One can also specify an **image URL**; press enter, and the image will be inserted as the note background.
			Another example of co-creating an RPG character library and game mockup environment. The image search input box (highlighted in the pink rectangle) allows you to type character's name and retrieve images from Google. The black trash box allows players to drop characters and delete them.

videonote		<<videonote001 as videonote>>	videonote nugget can be seen as the combination of the video nugget and the note nugget. One can also **resize** a video by dragging the handler at the right bottom of the video.
			This is an example of using videonote nugget to embed a set of videos, which can be played simultaneously or in a particular sequence to convey certain concepts. The final result could be much more powerful than static images.
panel		<<panel001 as panel>>	The panel nugget allows one to refer pages within MikiWiki and make them movable panels. It gives users flexibility to combine/reuse different pages.
			A role-playing game environment is made of a set of MikiWiki pages, e.g. a form, a chat, two characters' sheets, two dice, a story seeds generator and a map. In RPG environment, players can easily create another page panel to include new MikiWiki pages to support playing games. On the other hand, each MikiWiki page has its own individual data, which is independent from the RPG environment.

Appendix B – MikiWiki API

load_javascript(url, callback)
loads JavaScript from anywhere on the web

Example: loads external JS library (http://tweet.seaofclouds.com/jquery.tweet.js) for getting johnmaeda's tweets

```
load_javascript("http://tweet.seaofclouds.com/jquery.tweet.js",function(){
  $("#mytweets").tweet({
    join_text: "auto",
    username: "johnmaeda",
    avatar_size: 48,
    count: 5,
    loading_text: "loading tweets..."
  });
});
```

load_css(url)
loads css from anywhere on the web

Example: loads pippatoledoshop site's css
load_css("http://pippatoledoshop.com/css.css");

css within mikiwiki, we have to specify the data format is raw
load_css('http://miki.aqalert.net/demos/example/API/css/mycss**?raw=y**');

currentPageName()
gets the current page name, e.g. "**foo/bar/hello**"

isString(obj)
checks the object type whether it is string

imgurl(pagename)
returns the full URL to an image contained within a page, just image URL without any page layout

rand(top)
returns a random number between 1 and top (top is the maximum value)

pick(array)
picks a random element from an array

deepclone(obj)
clones an object and all its associated sub-objects

denyaccess()
removes the mikiwiki page content and redirects to the "http://miki.aqalert.net/access-denied" page

contains(array,element)
checks whether an element is in an array

keys(obj)
returns an array containing all the keys in a hash

props(obj)
returns an array containing all the properties in an object

isActiveTab()
determines the browser window/mikiwiki page is active or not

Page methods

Page(pagename)
returns a Page object. It is always an absolute path.

Page('foo/bar') maps to the /foo/bar page

the page object has many useful methods.

To do relative lookup, we should do
Page('foo/bar').lookup(function(found_page_name){ ... })
This will look for a foo/bar starting from the current page and going through the full lookup path

pageobj.imageURL() => it returns the full URL to an image contained within a page.
the URL to an image page also contains the page layout etc, not only the image. to get a link to the image only when we use imageURL().

mypage.loadContent(function(html){ … })
loads mypage content in a HTML format

mypage.loadRaw(function(text){ … })
loads mypage raw content

mypage.getJSON(function(json_obj){ … })
gets mypage JSON data as a JavaScript object

mypage.renderTo("#my-div")
renders to a specific css selector

mypage.loadMetadata(function(metadata_as_json_obj){ … })
loads mypage metadata

Nugget methods

nugget.page
returns a page object

nugget.pagename
return the pagename

nugget.id
returns the id of the HTML element containing the nugget displayed in the page

nugget.data()
gets the data passed from to the nugget from the data page and the nugget options. The data is assumed to have been expressed either in JSON or as a comma separated list

nugget.singledata()

gets the data passed to the nugget, assuming that it is a single element

nugget.username
returns the user's name, who are using the nugget, which can be used for synchronization

nugget.updateJSON(obj)
updates the data page corresponding to the nugget with the JSON representation of the object passed as a parameter

nugget.update(text)
writes the text to data page of the nugget

nugget.$(selector)
finds a specific HTML element by looking only within this nugget HTML.
useful to avoid interferences between several nuggets of the same type on a page

nugget.out(text)
can be called many times... stores text (and HTML) to write out on the page

nugget.setHTML()
writes all stored output to the page in the area where the nugget has been embedded

nugget.isRunningInSidebar()
checks whether this nugget is running in the sidebar

Example 1

Let's create a **big** nugget**,** which prints out some text a user passes with a bigger font.
Ideally we want to use it within a mikiwiki page like this:

> This is something <<**big**:really important>> !

1) First let's create the **'big'** nugget **format** page, we need to set this mikiwiki page Format as **javascript**:
By default the **Format** of a mikiwiki page is **miki**, and the editor will be a rich editor. However, when one changes the **Format** from **miki** to **javascript**, the rich text editor will switch to a JavaScript editor to better support coding.

mystring = **nugget.singledata**(); //gets the text passed by the user

nugget.out("");
nugget.out(mystring);
nugget.out("");

nugget.setHTML(); // this creates the HTML code and renders it in a mikiwiki page

2) To use the 'big' nugget, we need to create another type of page: visualization page, whose Format is **miki**. By default the **Format** of a mikiwiki page is **miki** and when we go to the edit mode, it will be a rich text editor. It's much easier for text editing.

After we create a new mikiwiki page, we can include big nugget with double angular brackets **<<>>** syntax.

<<**big**:really important>>

In this case, we are passing "really important" via nugget.singledata().

the finally result will print out “really important” in the test-big mikiwiki page with font size 200%.

Example 2

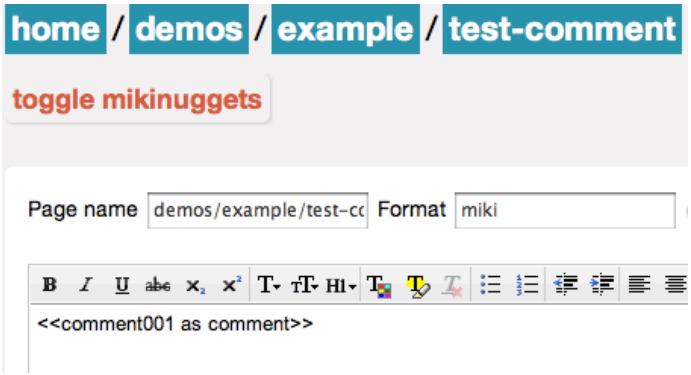
However, sometimes that data is not so simple. When the data become more complex, we need a data page to store them.

A nugget is made of three types of pages: a visualization page, a data page and a format page. The visualization page is the final rendered result, the data page contains this nugget’s data and the format page defines how to visualize the data from the data page.
I am using the “**comment**” nugget as an example here.

1) nugget visualization page: test-comment
pagename: **test-comment**
Format: **miki**

By default the **Format** of a mikiwiki page is **miki** and when one goes to the edit mode, one can use a rich text editor.

It includes the comment nugget in the visualization page: <<**nugget_data_page_name** as **nugget_name**>>



This is the rendered view of the comment nugget visualization page

2) nugget data page: **comment001**
pagename: **comment001**
Format: **comment**

When one creates a mikiwiki page, the Format is always **miki** by default. One must change the Format **miki** to "**a nugget's name**" in order to create a connection between the data page and the format page. This way mikiwiki knows how to render the data page.

The data page stores the data in JSON format. comment001 stores users comments

3) nugget format page: **comment**
pagename: **comment**
Format: **javascript**

The JavaScript in the format page defines how to process the data and how to display the data in the mikiwiki page. In this case, the Format is **comment**.

When we change the **Format** from **miki** to **javascript**, the rich text editor will switch to a JavaScript editor to better support our coding!!!

format page code:

var cs = **nugget.data();**

this gets users' comments from the comment nugget's data page, **comment001**
in this case 'cs' is an array of strings.

nugget.out("<ul>"); //writes out

```
for(var i=0; i<cs.length; i++){
  nugget.out( "<li>" + cs[i] +"</li>" );
}
```

nugget.out("
"); // closes the 'ul' tag

this goes through users comments and inserts each comment into a <li> tag
nugget.out(..) builds up a buffer of text that will be written inside the page as HTML.

To create a form to submit new comments:

```
nugget.out(  "<form action='#'>" );
nugget.out( "<input type='text' id='comment-input'>" );
nugget.out( "<input type='submit' id='comment-submit' >" );
nugget.out( "</form>" );
```

this creates the HTML code and renders it in a mikiwiki page

nugget.setHTML();

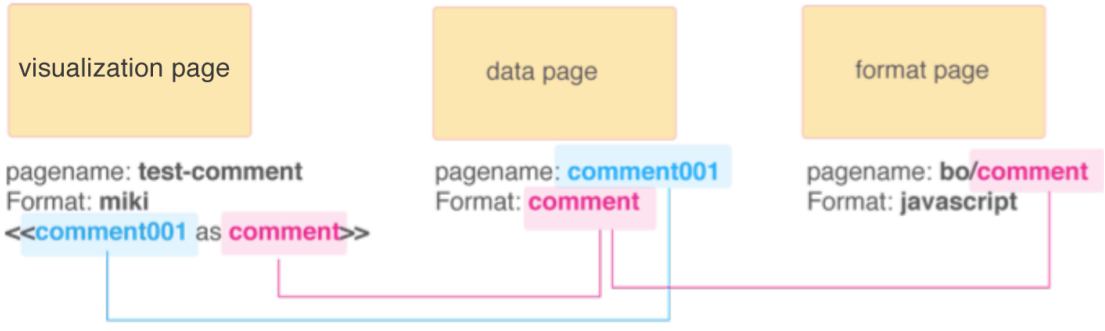
finally we connect some of the HTML elements we have created within the nugget to actions that are triggered when we interact with the nugget.

```
nugget.$('#comment-submit').click( function(){ // finds the DOM element with the 'comment-submit' id
    cs[cs.length]=nugget.$('#comment-input').val();  // finds the DOM element with the 'comment-input' id

    nugget.afterUpdating = function(){
        out = '';
        for(var i=0; i<cs.length; i++){
          out = out + "<li>" + cs[i] +"</li>";
        }
        nugget.$('ul').html(out); // finds the DOM element with  a 'ul' tag
    };

    nugget.updateJSON( cs ); //updates the comment001 data page, if a user submits a new comment
    return false;
});
```

The intertwined relationship between the visualization page, the data page and the format page of the comment nugget.



Advanced Users API in Detail

All the API functions described here can be accessed from within a *javascript* nugget.

Page Object

In some cases, a nugget requires to retrieve the data of another nugget that it references. For example, the *chat* nugget shows the thumbnail of the user who exchanges messages, which is done by retrieving the user's profile picture from the data page used to store the user's profile information.

The *Page* object allows to access the information stored within a MikiWiki page both in raw format or as a JSON object: page content, page path, page format, page meta-data, etc.

```
19 function addThumbnail(username,jqelement){
20   Page('users/'+username).getJSON(function(user){
21     jqelement.append(
22       "<img src='" + user.picture + "' title='"+username+"' style='width:20p
23 5px 0px 2px; padding:0; border: 1px solid black;'/>"
24     );
25   });
26 }
```

The above code shows how the *Page* object is used to retrieve the "*users/username*" page information, stored as JSON data. The function returns a user profile object from which one can retrieve the user picture URL - *user.picture* - and render it on the browser page as HTML.

A simple example that generates a link to a MikiWiki page is shown below, to create a Page object:


```
1 var mypage = Page(pagename);
2
```

In this example *pagename* is the full path of the page that the user wants to link.

Page has a number of useful methods to access, set or display the content of a Page. One can generate a clickable link to the referenced page by calling the appropriate method:

```
1 mypage.html_link();
```

Nugget Object

Once a nugget is executed, our code should be able to access all the run-time properties of the nugget. This is achieved via the *nugget* object.

The *nugget* object is created during the rendering process and it is passed to the relevant format page when it is executed. The *nugget* object provides contextual information that is needed to execute the code.

This is one of the simplest services of the *nugget* object:

```
1 nugget.setHTML( nugget.pagename );
```

All this nugget does is to print its own name to the screen. However, this simple operation is already implying a number of things. The nugget knows where and how to render inside the HTML page. And the nugget knows the name of the page associated with it. These are all contextual information that are generated by the run-time environment and are passed to our nugget via the implicitly generated *nugget* object.

More information is available at runtime during the nugget execution:

```
nugget.pagename
nugget.formatpagename
```

Users are able to access the data page associated with the nugget, and manipulate it as a *Page* object:

```
nugget.page();
```

One can access the context in which the nugget is rendered. The following, for example, allows a user to access the *Page* object associated with the topmost nugget in the web page, containing our nugget.

```
nugget.topPage();
```

To reposition the nugget within its container, one needs to access the html fragment enclosing the nugget. This can be done by utilizing the unique name of the nugget, which corresponds to the CSS id within the web page:

```
nugget.uniquename();
```

At the same time a nugget can access and manipulate the CSS of enclosed elements.

```
nugget.$(selector)
```

This code performs an operation equivalent to the JQuery *find* method (jQuery 2006), but it looks up only elements contained within the nugget.

Occasionally it is necessary to keep tracking who is accessing the nugget at run-time. For example it is essential for the *chat* nugget to record who is writing an entry. This can also be achieved by querying the nugget run-time properties:

```
nugget.username
```

Finally, a nugget is able to write to the HTML area that it has been assigned to.

```
nugget.out(text)
```

The *out* method appends an html fragment to an output buffer. The whole buffer is flushed with the *setHTML* method.

```
nugget.setHTML(text)
```

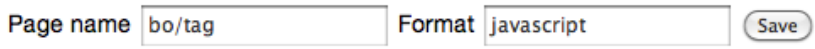
This immediately renders to the HTML in the browser. After this command is executed, it is still possible to attach function callbacks to the rendered elements to establish an interaction with the user.

Loading JavaScript and CSS dynamically

JavaScript and JQuery libraries are widely used on the web at the time of writing. Almost all successful online services support JQuery libraries that allow external websites integration. In order to take advantage of the existing web ecosystem, MikiWiki provides two functions that allow loading third party JavaScript libraries and externally defined CSS.

```
load_javascript(url, callback);
load_css(url);
```

These functions permit loading JavaScript and CSS from another page of the MikiWiki, allowing the definition of internal CSS pages and local JavaScript libraries.




```
load_css("http://miki.aqalert.net/css/jquery.tagit.css");
load_javascript("http://aehlke.github.com/tag-it/js/tag-it.js",function(){

  var tagHints = nugget.data();
  nugget.out('<ul id="myULTags"></ul>');
  nugget.out( "<a href='#' id='saveTags'>Save Tags</a> " );
  nugget.setHTML();

  // more code goes here...

});
```

Loading an external library allows users to leverage existing functionalities and immediately add new features to MikiWiki. In the above example it demonstrates using *load_css()* and *load_javascript()* functions to load the existing *TagIt* JQuery library.

Service APIs

This section describes the major service APIs that can be exploited by advanced users to meta-design more comprehensive nuggets.

Nugget Data Access

MikiWiki allows the freedom to express data in various ways. Basic supported types are plain text, as MikiML, as a comma separated list, XML and JSON structured text formats. Using the nugget object users can retrieve the data and configure parameters associated with it.

nugget.raw returns the data as it is, which empowers users with flexibility of creating their own format to handle how to parse the data, for example creating a custom format for a specific component as the way Balsamiq handles the data (see Chapter 5)

```
nugget.raw
//the raw textual content of the data page associated with the nugget

nugget.data(); //returns the data as decoded JSON
nugget.xmldata(); //returns the data as decoded XML

nugget.singledata()
/* returns the data as a single string of text,
  useful to handle nuggets with only 1 parameters */
```

On the other hand, employing one of the standard encoding methods, users can rely on existing methods. *Xmldata()* and *data()* decode respectively an XML data format and a JSON data format as a JavaScript object.

The *data()* function is able to interpret a list of comma separated text as an array, without relying on the more articulate JSON format.

Singledata() is used when only a single atomic value is expected.

Finally the data can be modified. The following two update methods, modify the underlying data by passing either the raw text or an object to be saved in a JSON format:

```
nugget.updateJSON(data);
nugget.update(text);
```

Page Data Access

In the case of a nugget, the nugget object provides an access point to the page data. If a user needs data not from the current nugget but from another page, it can be done by explicitly creating a Page object and use it to access that data.

```
mypage.loadContent(function(pagecontent){
})
//loads content as expanded html

mypage.loadRaw(function(pagecontent){
})
//loads content as raw text as stored on the server

mypage.getJSON(function(pagedata){
})
//loads a page content and interprets it as JSON format

mypage.loadMetadata(function(metadata){
})
```

The specialized functions above allow users to load the data from a specific format.

Similar to the nugget object, one can update the data of another page:

```
mypage.update( "miki", "hallo world!", function(){
// called when updated!
})
```

Media

Uploading of external resources such as images or PDF files is done via an AJAX POST call, passing the media object as a bytestream via the 'file' parameter.

This service can be accessed by posting a form and passing the file name as well as its local file path.

```
<form id="choose_pix" action='<%= @page.root? ? '/upload' : "/#{@page.closer_directory_name}
enctype='multipart/form-data' method='post'>

 <input name='file' type='file' /><br/>
 <input type='submit' value='Upload' />

</form>
```

Once a source is uploaded, a metadata page associated to this file will be generated. The file can be handled as a nugget.

Registration

The registration service is used to create a new user within the system.

```
mikiwikiRegister(name,email,url,password, function(){
//success!
});
```

This function creates a new *profile* nugget within the *users'* system folder. MikiWiki references these pages when users try to log in. Meta-designers can further enhance the profile information by adding fields such as Skype, role, profile picture, and email address. This extra users' information can then further be used by other nuggets to notify users, display their thumbnails, provide a Skype call access to the user, etc.

Versioning

MikiWiki stores the previous versions of a nugget and its metadata after every change to the nugget data. These older versions are saved as MikiWiki pages within a hidden folder that can therefore be accessed as normal nuggets.

This code retrieves a list of *Page* objects referencing the previous versions of a data nugget. With the list of previous versions, a user can access their data via employing all the methods of the *Page* object.

```
nugget.page().getHistory(function(versions){
  if (versions.length > 0){ //if there is some history
    // get access to the previous version
    versions[0].getJSON(function(pagedata){
      // do something with the loaded JSON data..
    });
  }
});
```

Alternatively, users may simply wish to restore the data nugget to a previous version

```
nugget.restore(previous_version_id);
```

or to compare the differences between the current and a previous version.

```
var diffs = nugget.diffs(previous_version_id);
```

Authentication

If one has set up authorization to a page or environment, the user has the right to change their permissions. The *allow* nugget utilizes the authentication service to accomplish this:

```
Page(pagename).authorize(readonly,readwrite);
```

The *readonly* and *readwrite* parameters are used to specify individual users that have only read access or full access to the page. Users can add one or more role fields to their own profile so that they can exploit these as wide alternatives to be passed to the authorization service, rather than individual names.

Notification

MikiWiki supports a simple notification service that sends emails to users.

```
mikiwikiNotify(users,text);
```

One can specify a list of users and a text to be sent to all of them. This is the service used by the *notify* nugget. Theoretically this service could be removed from the core of MikiWiki and externalized to a third party service that supports a JSONP interface to email sending.

Awareness

The awareness service is supported in MikiWiki by keeping track on the server-side of which users are accessing pages.

The following function returns to the client a full list of entries of users, pages, and the timestamp of the last access.

```
mikiwikiActivity(function(people_entries){
// it is an array where each entry is [username, page, time]
});
```

Since a user may have left a MikiWiki page open but unutilized, the timestamp provides useful information on when the latest update happened.

Environment

The environment API allows to gather information related to environment and folder pages.

```
mypage.subpages(function(array_of_subpage_information){
// the subpage information contains metadata on the subpages
});
```

The *subpages* function, for example, retrieves information from the server relative to the pages and sub-folders contained within an environment.

Figure 125 Tags associated with each document

This function can be used in some nuggets that override the standard folder visualization and navigation, by showing the tags of a page close to its name as seen in Figure 125. This application of the environment service is described in the context of the Aristotele experiments in Chapter 9.

Lookup

If a user does not provide the full path, then she can still look up a page by specifying only part of the name and leave the lookup mechanism handling the lookup of the particular page.

```
Page("shoppinglist").lookup(function(shoppinglist_fullpath){
    // do something here with shoppinglist_fullpath
});
```

The lookup mechanism executes the searching process starting from the page the user is currently accessing.

This method is used by the *link* nugget, which looks up the linked resource starting from the page that embeds the link.

Tagging

The tagging service allows users to assign tags to a certain MikiWiki page. The tag nugget is built based on this service and allows users to save tags to the metadata of a page and to update them at any time.

```
Page("just/a/page").loadTags(function(tags_array){
});

mypage.saveTags(tags_array , function(){
// called when saved!
});
```

Since tags are associated with MikiWiki pages, tagging can be used for search relevant information. The tagging service has been employed to prototype the Aristotele project recommender concept, which will be discussed in more detail in the design study Chapter 9.

Search

```
mikiwikiSearch(text,function(results){
// results are an array of page names
});

mikiwikiSearchTagsOnly(text,function(results){
  // results is an array of page names
});
```

The search service allows users to run searches across the whole nugget knowledge base. As shown in the code below, the *mikiwikiSearch()* and *mikiwikiSearchTagsOnly()* functions trigger a search by passing the search keywords via the *text* parameter.

The function returns all the results matching the *text* argument by triggering a callback once the search is completed.

Synchronous Communication

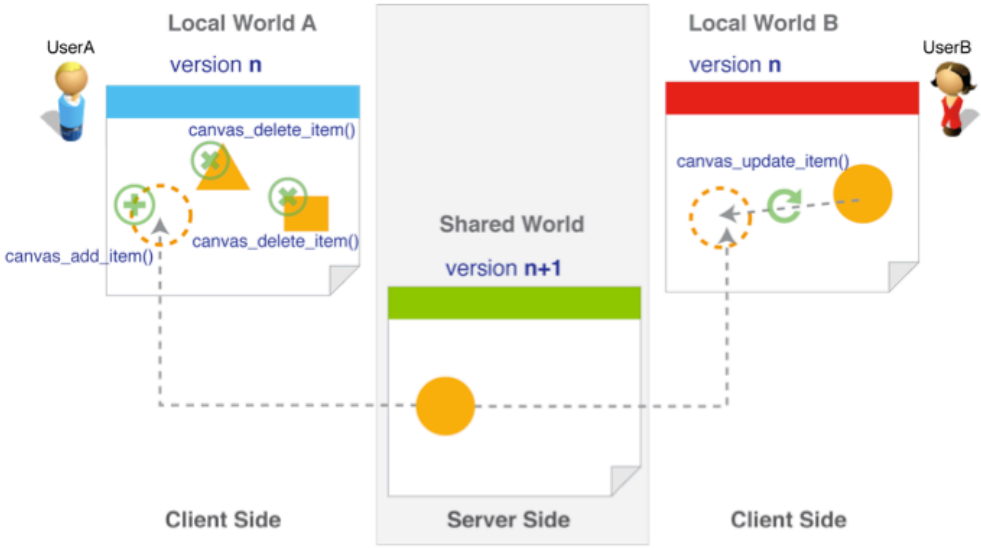
MikiWiki utilizes a centralized communication approach to support synchronous communication. The server forwards messages from and to every client, providing a simple conceptual model that fits well with web applications.

A shared world is a collection of *shared items*. *Local worlds* are copies of the shared world residing on individual browsers. A timestamp is used as a version number to keep track of different divergent local copies of the shared world. When the server holds the most recent version of the world, which is determined by the timestamp, then the shared world is sent to the out-of-date client so that it can be synchronized with its local world.

When a item is deleted from the local repository and a *canvas_delete_item()* function is called, passing to it the id of the object that has to be removed. This function contains the code to remove all visual and logical representations associated with the item.

The *canvas_add_item()* function is called and it is passed a piece of JSON describing the object that has to be created locally.

If it is determined that the remote item is more recent than the local item, then the *canvas_update_item ()* function is passed a JSON fragment representing the updated object and it determines how to re-render the local item.

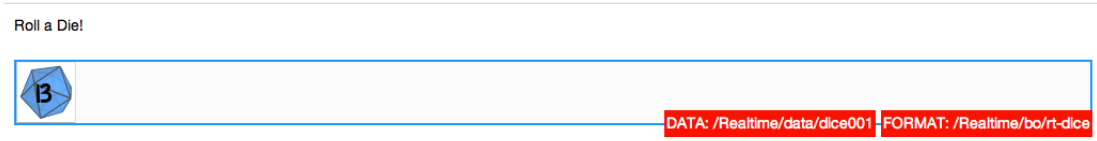


The shared world has a version number higher than the version number of two local worlds. Since the rectangle and the triangle in the local world A are not present in the shared world, the *canvas_delete_item(id)* function is invoked to remove them from local world A. Nevertheless, the circle in the shared world will be added to local world A by invoking the *canvas_add_item(obj)* function. The same circle is located differently in local world B and in the shared world; hence, the circle in local world B will be updated to the new position by invoking the *canvas_update_item(obj)* function. The version number of the shared world on the server is increased only when a new change is performed on the shared world hosted on the server.

There are three basic operations to manipulate the shared world are: *add*, *delete* and *update*. The *sharedspace_add_item()* function is used to add new items to the shared world, both locally and remotely; the *sharedspace_delete_item()* function updates the shared world by removing items; and the *sharedspace_update_item()* function applies changes to an existing item (e.g. position, size, media resource, etc);

If a user added a new item directly to the HTML page, without passing through the *sharedspace_add_item ()* function, then the run-time environment would not have any means to track and share the new item.

A Simple Example



This is a simple example, as the world contains only a single item (the die), which can be entirely represented by the die size (20 sided in this case) and by its result (13 for this particular roll).

This is the JSON data representation of the shared world after the die has been rolled:

The topmost JSON structure is a hash whose keys identify all the items in the 'shared world' – *die1* is the key identifies our only die in the world. The value associated with the key *die1* is another hash structure describing the item with a set of properties decided by the nugget author – e.g. the *size* and *value* properties in this case. One of these properties, *update*, is fixed and it stores the name of the user who last changed the item and the time of the change. This is necessary to know which changes should be overridden (by comparing timestamps) and to show to the users who made the roll (by using the associated *update.username* field).

Any JSON data page that fits this generic structure can be synchronized in real time using the synchronization APIs.

Once the die nugget gets executed, the local world is initially empty. The *nugget.realtimeSynchEvery(2000)* call starts the synchronization process with the shared world, checking for updates every 2000 milliseconds.

As soon as the first synchronization occurs, the client receives the initial JSON data describing the die stored on the server. Since the world contains only one object, *canvas_add_item()* gets invoked once, with the JSON fragment of the die being passed to it.

As shown in the code, *canvas_add_item()* has been overridden by the author of the *realtime-dice* nugget. When the new die gets created, an external web service is used to generate the picture of the die, and the picture is appended to the HTML. As soon as the die is made available on the web browser, the code connects a callback to the click event, so that the die can be rerolled by clicking it.

Once the die is rerolled, a new random number is generated and it is passed to the *sharedspace_update_item()* method. The run-time system distributes the new die value to the other users and invokes locally the *canvas_update_item()* to update the die image. This function shakes the die using a JQuery effect before retrieving a new picture for the dice and applying it.

All the synchronous communication nuggets in MikiWiki are implemented following these principles, for instance *chat*, *sync-imagenote* and *sync-note*.

Page name demos/Realtime/bo/rt- Format javascript Save

```
nugget.setHTML("<div id='rtcanvas'></div>");

var canvas = nugget.$('#rtcanvas');

function die_src(size,value){
  return "http://catchyourhare.com/diceroller/dieimage.php?sides="
    +size+"&result="+value+"&colour=1e90ff";
}

nugget.canvas_add_item = function(cfg){
  var die_image = $("<img></img>").attr('src',die_src(cfg.size,cfg.value));

  canvas.append( die_image.attr({id: cfg.id}) );

  die_image.click(function(){
    var obj = nugget.item(cfg.id);
    obj.value = rand(obj.size);
    nugget.sharedspace_update_item(obj);
  });
}

nugget.canvas_update_item = function(obj){
  var item = canvas.find('#'+obj.id);

  item.effect('shake', { times: 2 }, 100);
  item.attr('src',die_src(obj.size,obj.value));
};

nugget.realtimeSynchEvery(2000);
```

Appendix C - Publications

Zhu, L., Herrmann, T. (2012). A Meta-design Approach to Interactive Spaces, Designing Collaborative Interactive Spaces for e-Creativity, e-Science and e-Learning Workshop in AVI 2012, Capri, Italy, 12-25 May

Zhu, L., Herrmann, T. (2012). Design Now! – Elaborating Requirements in Situated Action, 2nd Workshop on Creativity in Requirements Engineering, CreaRE 2012 in conjunction with REFSQ 2012, 19 March.

Zhu, L. (2011). Collaborative Design: Design for Evolution, DESIRE'11-Creativity and Innovation in Design, DESIRE'11, Eindhoven, Netherlands, October 19-21, 2011. (pp.255-266). ACM Press

Carneiro, G. Zhu, L. (2011). ilo Cards: A Tool to Support Collaborative Design of Interactive Objects, DESIRE'11-Creativity and Innovation in Design, DESIRE'11, Eindhoven, Netherlands, October 19-21, 2011. (pp. 357-358). ACM Press

Zhu, L., Vaghi, I., Barricelli, B. R. (2011). A Meta-reflective Wiki for Collaborative Design, 7th International Conference on Wikis and Open Collaboration, WikiSym2011, California, USA. October 3-5, 2011. (pp.53--62). ACM Press

Ardito, C., Barricelli, B. R., Buono, P., Costabile, M. F., Piccinno, A., Valtolina, S., Zhu, L. (2011). Visual Mediation Mechanisms for Collaborative Design and Development, Proc. of UAHCI 2011 (pp. 3-11). LNCS 6765, Springer. ISBN: 978-3-642-21671-8.

Zhu, L. (2011). A Meta-design Framework to Support Multidisciplinary Teams' Online Collaboration. International Symposium on End-User Development. Proc. of IS-EUD 2011 (pp. 403-406). LNCS 6654, Springer. ISBN: 978-3-642-21529-2. (Piero Mussio Award)

Zhu, L., Vaghi, I., Barricelli, B. R. (2011). MikiWiki: A Meta Wiki Architecture and Prototype Based on the Hive-Mind Space Model. Proc. of IS-EUD 2011 (pp. 343-348). LNCS 6654, Springer. ISBN: 978-3-642-21529-2.

Zhu, L., Vaghi, I. (2011). MikiWiki: a Meta-design Framework for Collaboration. International Symposium on Collaborative Technologies and Systems (CTS2011), Philadelphia, USA. May 23-27, 2011. (pp.109 - 116) IEEE Press

Zhu, L., Barricelli, B. R., Iacob, C. (2011). A Meta-design Model for Creative Distributed Collaborative Design. International Journal of Distributed Systems and Technologies. 2(4), (pp.1-16).

Iacob, C., Zhu, L. (2010). Creative Processes in Collaborative Design of Software Applications. Swiss Design Network Conference 2010, Basel, Switzerland, October 28-30, 2010

Zhu, L. (2010). A Model for Cross Boundary Collaboration, 6th Nordic Conference on Human-Computer Interaction, Reykjavik, Iceland, October 16 - 20. 2010. Available online: https://sites.google.com/site/nordichi2010dc/participants

Zhu, L., Mussio, P., Barricelli, B.R., (2010). Hive-Mind Space Model for Creative, Collaborative Design, Desire 2010, DIS 2010 Aarhus, Denmark, August 16-17. (pp121--130). ACM Press

Iacob, C., Zhu, L. (2010). From the Problem Space to the Web Space: A Model for Designing Localized Web Systems. 9th International Conference WWW/Internet 2010 (ICWI2010), Timisoara, Romania, October 14-17, 2010

Iacob, C., Mussio, P., Zhu, L., Barricelli, B.R. (2010). Towards a Pattern Language for the Design of Collaborative Interactive Systems. International Workshop on Visual Formalism for Patterns , IEEE Symposium on Visual Languages and Human-Centric Computing, Madrid, Spain, September 21-25, 2010

Barricelli, B.R., Iacob, C., Zhu, L. (2010). BANCO Web Architecture to Support Global Collaborative Interaction Design. Proceedings of the 9th International Workshop on Internationalization of Products and Systems (IWIPS 2010), London, UK, July 7-10, 2010, pp. 159-162

Zhu, L. (2010). A Model for Supporting Emergent Collaboration and Practices, Marie Curie Conference, Turin, Italy, July 01, 2010

Zhu, L., Iacob, C., Barricelli, B.R. (2010). New Design Strategies: Using the Hive Mind Space Model to Enhance Collaboration. IADIS Multi Conference on Computer Science and Information Systems 2010, Freiburg, Germany, July 26-31, 2010, pp. 12-19 (Best Paper Award)

Zhu, L, (2010). Applying the Hive Mind Space Model to Foster Creative Collaboration, International Symposium on Collaborative Technologies and Systems (CTS2010), Chicago, USA. (pp.617 - 625). IEEE Press

Zhu, L., Mussio, P., Barricelli, B.R., Iacob, C. (2010). A Habitable Space for Supporting Creative Collaboration. Proceedings of International Symposium on Collaborative Technologies and Systems (CTS2010), Chicago, USA, May 17-21, 2010, (pp. 617-622). IEEE Press

Barricelli, B.R., Iacob, C., Zhu, L. (2009). Map-Based Wikis as Contextual and Cultural Mediators. Community Practices and Locative Media workshop, 11th International Conference on Human-Computer Interaction with Mobile Devices and Services, September 15-18, 2009, Bonn, Germany. Available online: http://www.uni-siegen.de/locatingmedia/workshops/mobilehci/

Zhu, L, Karatzas, K and Lee, J. 2009 Urban environmental information perception and multimodal communication: the air quality example In A Esposito, A Hussain, M Marinaro, R Martone (eds), Multimodal Signals: Cognitive and Algorithmic Issues (Selected papers from COST Action 2102 1st International Training School), LNAI vol.5398, Springer Verlag, (pp. 288-299).

Technical reports:

Barricelli, B. R., Iacob, C., Mussio, P., Zhu, L. (2010). Map-Based Wikis as Culture, Role, and Platform Mediators. Technical report Department of Computer Science and Communication, Università degli Studi di Milano RT 37-10.

Zhu, L., Mussio, P., Barricelli, B. R., Iacob, C. (2010). A Habitable Space for Supporting Creative Collaboration. Technical report Department of Computer Science and Communication, Università degli Studi di Milano RT 34-10.

References

37signals (2004) Basecamp. http://basecamphq.com/?referrer=YT30B7. Accessed 02. Oct. 2010

Abrahamse W, Steg L, Vlek C, Rothengatter T (2005) A review of intervention studies aimed at household energy conservation. Journal of Environmental Psychology 25 (3):273-291

Abran A, Moore, J.W., Bourque, P., Dupuis, R. & Tripp, L.L. (2004) Guide to the Software Engineering Body of Knowledge 2004 Version. IEEE Computer Society, Los Alamitos, CA, USA.

Ackerman MS (2000) The intellectual challenge of CSCW: the gap between social requirements and technical feasibility. Hum-Comput Interact 15 (2):179--203. doi:http://dx.doi.org/10.1207/S15327051HCI1523_5

Ackerman MS, Halverson C (2004) Organizational Memory as Objects, Processes, and Trajectories: An Examination of Organizational Memory in Use. Computer Supported Cooperative Work - CSCW 13:155-189

Alexander C, Ishikawa S, Silverstein M, Jacobson M, Fiksdahl-King I, Angel S (1977) A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York

Amabile TM (1983) The Social Psychology of Creativity. Springer

Andersen R, Mørch AI (2009) Mutual Development: A Case Study in Customer-Initiated Software Product Development. Paper presented at the Proceedings of the 2nd International Symposium on End-User Development, Siegen, Germany

Anderson C (2006) The Long Tail: Why the Future of Business is Selling Less of More. vol 10. Hyperion, New York, NY

Andriessen JHE, Hettinga M, Wulf V (2003) Introduction to Special Issue on Evolving Use of Groupware. Comput Supported Coop Work 12 (4):367-380. doi:10.1023/A:1026119416700

Anslow C, Riehle D (2008) Towards end-user programming with wikis. Paper presented at the Proceedings of the 4th international workshop on End-user software engineering, Leipzig, Germany

Anzai T (1994) Renga Notes. http://www.renga.com/archives/strings/note94/index.htm. Accessed 6 July 2004

Ardito C, Barricelli BR, Buono P, Costabile MF, Piccinno A, Valtolina S, Zhu L Visual mediation mechanisms for collaborative design and development. In: HCI International 2011, Orlando, 2011.

Arias E, Eden H, Fischer G, Gorman A, Scharff E (2000) Transcending the individual human mind - creating shared understanding through collaborative design. ACM Trans Comput-Hum Interact 7 (1):84-113. doi:10.1145/344949.345015

Arias E, Fischer G Boundary Objects: Their Role in Articulating the Task at Hand and Making Information Relevant to It. In: International ICSC Symposium on Interactive & Collaborative Computing (ICC'2000), University of Wollongong, Australia, ICSC Academic Press, Wetaskiwin, Canada, 2000. pp 567-574

Arieti S (1976) Creativity: the Magic Synthesis. Basic Books, New York

ARISTOTELE (2009) ARISTOTELE Project. http://aristotele.crmpa.unisa.it/default.aspx.

Åsand H-RH, Mørch AI (2006) Super Users and Local Developers: The Organization of End-User Development in an Accounting Company. Journal of Organizational and End User Computing 18 (4):1-21

Ashkenas J (2011a) CoffeeScript. http://coffeescript.org/. Accessed 18. July 2011

Ashkenas J (2011b) List of languages that compile to JS. https://github.com/jashkenas/coffee-script/wiki/List-of-languages-that-compile-to-JS. Accessed 21. Jan 2012

Atwood ME, McCain KW, Williams JC How does the design community think about design? In: Proceedings of Designing Interactive Systems (DIS), ACM, New York, 2002. pp 125-132
Avison DE, Lau F, Myers MD, Nielsen PA (1999) Action research. Commun ACM 42 (1):94-97. doi:10.1145/291469.291479
Bærenholdt JO, Büscher , M, Scheuer , J D & Simonsen , J , (2010) Perspectives on design research. In: Simonsen J, Bærenholdt , J O , Büscher , M & Scheuer , J D (ed) Design Research: Synergies from Interdisciplinary Perspectives. Routledge London, pp 1-15
Balsamiq (2008). http://balsamiq.com/. Accessed 13. Oct 2010
Barricelli BR (2010) An Architecture for End-User Development Supporting Global Communities. Università degli Studi di Milano, Milan
Barricelli BR, Iacob C, Zhu L (2009a) Map-Based Wikis as Contextual and Cultural Mediators. Paper presented at the Community Practices and Locative Media workshop at Mobile HCI09, Bonn
Barricelli BR, Marcante A, Mussio P, Provenza LP, Padula M (2009b) Designing Pervasive and Multimodal Interactive Systems: An Approach Built on the Field. In: Grifoni P (ed) Multimodal Human Computer Interaction and Pervasive Services. pp 243-264
Baskerville R (2001) Conducting Action Research: High Risk and High Reward in Theory and Practice. In: Trauth EM (ed) Qualitative Research in Information Systems: Issues and Trends. Idea Group Publishing, London, pp 192-217
Bechky BA (2003) Sharing meaning across occupational communities: The transformation of understanding on a production floor. Organization Science 14, No. 3:312-330
Bell B, Lewis C (1993) ChemTrains: A Language for Creating Behaving Pictures. In: 1993 IEEE Workshop on Visual Languages (Bergen, Norway). IEEE Computer Society Press, pp 188-195
Bellotti V, Rogers Y (1997) From Web Press to Web Pressure: Multimedia Representations and Multimedia Publishing. In: Proceedings of ACM CHI 97 Conference on Human Factors in Computing Systems 1997, vol 1. pp 279-286
Benford S, Bowers J, Fahlén LE, Greenhalgh C, Snowdon D (1995) User embodiment in collaborative virtual environments. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems, Denver, Colorado, United States
Benford S, Bowers, J., Fahlen, L., Mariani, J., Rodden, T. (1994) Supporting Cooperative Work in Virtual Environments. The Computer Journal 37 (8):653-668
Benford S, Greenhalgh C, Reynard G, Brown C, Koleva B (1998) Understanding and constructing shared spaces with mixed-reality boundaries. ACM Transactions on Computer-Human Interaction (TOCHI), 5 (3):185-223
Benkler Y (2006) The Wealth of Networks: How Social Production Transforms Markets and Freedom. Yale University Press, New Haven
Bentley R, Dourish P (1995) Medium versus mechanism: supporting collaboration through customisation. Paper presented at the Proceedings of the fourth conference on European Conference on Computer-Supported Cooperative Work, Stockholm, Sweden
Bentley RaH, J. A. and Randall, D. and Rodden, T. and Sawyer, P. and Shapiro, D. and Sommerville, I. (1992) Ethnographically-informed systems design for air traffic control. Paper presented at the In Proceedings of the 1992 ACM conference on Computer-supported cooperative work (CSCW '92), Toronto, Ontario, Canada
Berners-Lee T (2005) Uniform Resource Identifier (URI): Generic Syntax. http://www.ietf.org/rfc/rfc3986.txt. Accessed 19 Jan 2012
Bjerknes G, Bratteteig T (1987) Florence in Wonderland: System development with nurses. In: G. Bjerknes PE, & M. Kyng (ed) Computers and Democracy - A Scandinavian Challenge. Avebury, Avebury, UK, pp 279-295
Blum F (1955) Action research—a scientific approach? Philosophy of Science 22 (1):1-7
Boden MA (1994) Dimensions of Creativity, vol MIT Press. London
Bodker K, Pedersen J (1991) Workplace Cultures: Looking at Artifacts, Symbols and Practices. In: Greenbaum J, Kyng M (eds) There's No Place Like Home: Continuing Design in Use Design at Work: Cooperative Design of Computer Systems. Lawrence Erlbaum Associates, Hillsdale, NJ, pp 121-136

Bødker S (1998) Understanding representation in design. Hum-Comput Interact 13 (2):107-125
Bodker S, Gronbaek K (1991) Cooperative Prototyping: Users and Designers in Mutual Activity. International Journal of Man-Machine Studies 34:453-478
Bødker S, Grønbæk K, Kyng M (1993) Cooperative Design: Techniques and experiences from the Scandinavian Scene. In: Schuler D, Namioka, A. (ed) Participatory Design: Principles and Practices. Lawrence Erlbaum Associates, New Jersey pp 157-175
Borchers J (2001) A Pattern Approach to Interaction Design. Software Design Patterns. Wiley
Bourguin G, Derycke A, Tarby J-C (2001) Beyond the interfaces, Co-evolution inside Interactive Systems: A proposal founded on the Activity Theory. Paper presented at the IHM-HCI 2001, Toulouse.
Bowker GC, Star SL (1994) Knowledge and information in international information management: Problems of classification and coding. In: Information Acumen – The understanding and use of knowledge in modern business. Comparative and International Business : Modern Histories Routledge, London
Brancheau JC, Brown CV (1993) The management of end-user computing: status and directions. ACM Comput Surv 25 (4):437-482. doi:10.1145/162124.162138
Brandt J, Guo PJ, Lewenstein J, Klemmer SR (2008) Opportunistic programming: how rapid ideation and prototyping occur in practice. Paper presented at the Proceedings of the 4th international workshop on End-user software engineering, Leipzig, Germany
Bratteteig T Mutual Learning. Enabling cooperation in systems design. In: Braa KM, E. (ed) IRIS'20, Hankø, Norway, 1997. Department of Informatics, University of Oslo., pp 1-20
Brown A (1992) Design Experiements: Theoretical and Methodological Challenges in Creating Complex Interventions in Classroom Settings. The Journal of the Learning Sciences 2 (2):141-178
Brown AL, Campione JC (1994) Guided Discovery in a Community of Learners. In: McGilly K (ed) Classroom Lessons: Integrating Cognitive Theory and Classroom Practice. MIT Press, Cambridge, MA, pp 229-270
Bruns A (2008) Blogs, Wikipedia, Second Life, and Beyond: From Production to Produsage. Peter Lang Publishing, New York
Burgess J, Nye M (2008) Re-materialising Energy Use Through Transparent Monitoring Systems. Energy Policy 36 (12):4454-4459
Carlile PR (2002) A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development. Organization Science 13 (4):442-455
Carneiro G, Zhu L (2011) i o cards: a tool to support collaborative design of interactive objects. Paper presented at the Procedings of the Second Conference on Creativity and Innovation in Design, Eindhoven, Netherlands
Carroll J, Howard S, Vetere F, Peck J, Murphy J Just what do the youth of today want? Technology appropriation by young people. In: HICSS-35í02, Hawaii, 2002.
Chapanis A (1975) Interactive human communication. Scientific American 232 (3):36-42
Ciborra C (2002) The Labyrinths of Information: Challenging the Wisdom of Systems. Oxford University Press Inc, New York
Ciborra CU (1996a) Improvisation and InformationTechnology in Organization. Paper presented at the International Conference on Information Systems (ICIS),
Ciborra CU (1996b) The Platform Organization: Recombining Strategies, Structures, and Surprises. Organization Science 7 (2):103-118
Clark HH, Brennan SE (1991) Grounding in Communication. In: Resnick LB, Levine JM, Teasley SD (eds) Perspectives on Socially Shared Cognition. American Psychological Association, pp 127-149
Cloud9 (2011). http://c9.io/. Accessed 02. August 2011
Collins A (1992) Toward a design science of education. the Learning Sciences. E. Lagemann and L. Shulman New York
Collins A, Joseph D, Bielaczyc K (2004) Design Research: Theoretical and Methodological Issues. The Journal of the Learning Sciences 13 (1):15-42
Correia FF, Ferreira HS, Flores N, Aguiar A (2009) Incremental knowledge acquisition in software development using a weakly-typed Wiki. Paper presented at the

Proceedings of the 5th International Symposium on Wikis and Open Collaboration, Orlando, Florida
Costabile MF (2001) Usability in the software life cycle. In: Chang SK (ed) Handbook of Software Engineering & Knowledge Engineering, vol I. World Scientific, London, UK, pp 179-192
Costabile MF, Fogjli D, Fresta G, Mussio P, Piccinno A (2003a) Building environments for end-user development and tailoring. Paper presented at the Proceedings of the 2003 IEEE Symposium on Human Centric Computing Languages and Environments
Costabile MF, Fogli D, Fresta G, Mussio P, Piccinno A (2003b) Computer environments for improving end-user accessibility. Paper presented at the Proceedings of the User interfaces for all 7th international conference on Universal access: theoretical perspectives, practice, and experience, Paris, France
Costabile MF, Fogli D, Lanzilotti R, Marcante A, Mussio P, Provenza LP, Piccinno A (2007a) Meta-design to face co-evolution and communication gaps between users and designers. Paper presented at the Proceedings of the 4th international conference on Universal access in human computer interaction: coping with diversity, Beijing, China,
Costabile MF, Fogli D, Mussio P, Piccinno A (2006a) End-User Development: the Software Shaping Workshop Approach. End-User Development 9:195-217
Costabile MF, Fogli D, Mussio P, Piccinno A (2007b) Visual Interactive Systems for End-User Development: A Model-Based Design Methodology. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 37:1029 -1046. doi:10.1109/TSMCA.2007.904776
Costabile MF, Fogli, D., Lanzilotti, R., Mussio, P., Parasiliti P.L., and Piccinno, A. (2008) Advancing End-User Development Through Meta-Design. In: Clarke S (ed) End User Computing Challenges and Technologies: Emerging Tools and Applications. IGI Global, pp 143-167
Costabile MF, Mussio P, Provenza LP, Piccinno A Advanced visual systems supporting unwitting EUD. In: AVI '08 Proceedings of the working conference on Advanced visual interfaces, Napoli, Italy, 2008a. ACM, pp 313-316. doi:10.1145/1385569.1385621
Costabile MF, Mussio P, Provenza LP, Piccinno A (2008b) End users as unwitting software developers. Paper presented at the Proceedings of the 4th international workshop on End-user software engineering, Leipzig, Germany
Costabile MF, Piccinno A, Fogli D, Marcante A (2006b) Supporting interaction and co-evolution of users and systems. Paper presented at the Proceedings of the working conference on Advanced visual interfaces, Venezia, Italy
Crabtree A (2003) Designing Collaborative Systems: A Practical Guide to Ethnography. Computer Supported Cooperative Work. Springer-Verlag, New York
Crabtree A Design in the absence of practice: Breaching experiments. In: 5th International Conference on Designing Interactive Systems, Cambridge, MA, August 1-4 , 2004 2004. pp 59-68
Cross N (1993) Science and design methodology: A review. Engineering Design 5:63-69
Cross N (2006) Designerly Ways of Knowing. 1st edn. Springer,
Csikszentmihalyi M (1990) Flow: The Psychology of Optimal Experience. HarperCollins Publishers, New York
Csikszentmihalyi M (1996) Creativity — Flow and the Psychology of Discovery and Invention. HarperCollins Publishers, New York, NY
Curtis NA (2009) Modular Web Design: Creating Reusable Components for User Experience Design and Documentation. New Riders Press, Berkeley
Cypher A (ed) (1993) Watch What I Do: Programming by Demonstration. The MIT Press, Cambridge, MA
Darby S (2006) The Effectiveness of Feedback on Energy Consumption. University of Oxford, Oxford
Darke J (1979) The primary generator and the design process. Design Studies 1:36-44
Darroch V, Silvers RJ (1982) Interpretive Human Studies: An Introduction to Phenomenological Research. University Press of America, Washington, D.C.
Dawson C (2002) Practical Research Methods: A User-Friendly Guide to Mastering Research Techniques and Projects. How To Books Ltd

DeFanti TA, Sandin DJ, Nelson TH (1974) Computer graphics as a way of life. Paper presented at the Proceedings of the 1st annual conference on Computer graphics and interactive techniques, Boulder, Colorado
Désilets A, Paquet S, Vinson NG (2005) Are wikis usable? Paper presented at the Proceedings of the 2005 international symposium on Wikis, San Diego, California,
Dick H, Eden, H. and Fischer, G. (2011) From Consumers to Owners: Using Meta-design Environments to Motivate Changes in Energy Consumption. Paper presented at the IS-EUD'2011, Torre Canne, Italy
Dittrich Y, Lindeberg O, Lundberg L (2006) End-User Development as Adaptive Maintenance. In: Lieberman H, Paterno F, Wulf V (eds) End User Development. Springer, Dordrecht, pp 295-313
Dix A (2007) Designing for appropriation. Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI- Volume 2 978-1-902505-95-4. British Computer Society, University of Lancaster, United Kingdom
Dix AaF, Janet E. and Abowd, Gregory D. and Beale, Russell (2003) Human-Computer Interaction. Prentice-Hall, Inc, NJ, USA
Dorsey J, Glass N, Williams E, Stone B (2006) Twitter. twitter.com. Accessed 21. Dec. 2009
Dorst K, Cross N (2001) Creativity in the design process: co-evolution of problem–solution Design Studies 22(5):425–437
Dourish P (1995) Developing a reflective model of collaborative systems. ACM Trans Comput-Hum Interact 2 (1):40-63
Dourish P Extending Awareness Beyond Synchronous Collaboration. In: CHI'97 Workshop on Awareness in Collaborative Systems, Atlanta, 1997.
Dourish P (2003) The Appropriation of Interactive Technologies: Some Lessons from Placeless Documents. Comput Supported Coop Work 12 (4):465-490. doi:10.1023/a:1026149119426
Dourish P (2006) Implications for design. Paper presented at the Proceedings of the SIGCHI conference on Human Factors in computing systems, Montréal, Québec, Canada,
Dourish P, Bellotti V (1992) Awareness and coordination in shared workspaces. Paper presented at the Proceedings of the 1992 ACM conference on Computer-supported cooperative work, Toronto, Ontario, Canada,
Drucker PF (1994) The Age of Social Transformation. The Atlantic Monthly (November):53-80
Edmonds E, Candy L, Cox G, Eisenstein J, Fischer G, Hughes B, Hewett T (1999) Individual and/versus social creativity (panel session). Paper presented at the Proceedings of the 3rd conference on Creativity & cognition, Loughborough, United Kingdom,
Ehn P (1993) Scandinavian Design: On Participation and Skill. In: Schuler D, Namioka A (eds) Participatory Design: Principles and Practices. Lawrence Erlbaum Associates, Hillsdale, NJ, pp 41-77
Ehn P (2008) Participation in design things. Proceedings of the Tenth Anniversary Conference on Participatory Design 2008. Indiana University, Bloomington, Indiana
Ehn P, Kyng M (1991) Cardboard computers: Mocking-it-up or Hands-on the Future. In: Greenbaum J, Kyng M (eds) Design at Work: Cooperative Design of Computer Systems. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, pp 169–195
Eisenberg M, Fischer G (1994) Programmable Design Environments: Integrating End-User Programming with Domain-Oriented Assistance. In: Human Factors in Computing Systems, CHI'94 (Boston, MA). ACM, New York, pp 431-437
Engelbart DC (1995) Toward Augmenting the Human Intellect and Boosting Our Collective IQ. Communications of the ACM 38 (8):30-33
Engeström Y (1999) Innovative learning in work teams: Analyzing cycles of knowledge creation in practice. In: Engeström Y, Miettinen R (eds) Perspectives on Activity Theory. Cambridge University Press, pp 377-404
Engeström Y (2001) Expansive Learning at Work: Toward an Activity Theoretical Reconceptualization. Journal of Education and Work 14 (1):133-156
Engeström Y (2004) New forms of learning in co-configuration work. Journal of Workplace Learning 16:11-21
Engeström Y, Miettinen R (1999) Learning in Doing Social, Cognitive and Computational Perspectives. In: Engeström Y, Miettinen R, Punamäki R-L (eds) Perspectives on Activity Theory. Cambridge University Press, Cambridge, UK, pp 1-18

Esselink B (2000) A Practical Guide to Localization. John Benjamins Pub Co,
Farooq U, Carroll JM, Ganoe CH (2007) Supporting creativity with awareness in distributed collaboration. Proceedings of the 2007 international ACM conference on Supporting group work 978-1-59593-845-9. ACM, Sanibel Island, Florida, USA
Fischer G (1999a) Domain-Oriented Design Environments — Supporting Individual and Social Creativity. In: Gero JS, Maher ML (eds) Computational Models of Creative Design IV. Key Centre of Design Computing and Cognition, Sydney, Australia, pp 83-111
Fischer G (1999b) Symmetry of igorance, social creativity, and meta-design. Paper presented at the Proceedings of the 3rd conference on Creativity & cognition, Loughborough, United Kingdom
Fischer G (2000) Social Creativity, Symmetry of Ignorance and Meta-Design. Knowledge-Based Systems Journal (Special Issue on Creativity & Cognition), Elsevier Science BV, Oxford, UK 13 (7-8):527-537
Fischer G Communities of Interest: Learning through the Interaction of Multiple Knowledge Systems. In: IRIS'24, Norway, 2001.
Fischer G (2003) Meta-Design: Beyond User-Centered and Participatory Design. In: Stephanidis C, Jacko J (eds) Proceedings of HCI International 2003, vol Vol. 4. Lawrence Erlbaum Associates, Mahwah, NJ, Crete, Greece, June 2003, pp 88-92
Fischer G (2004) Social Creativity: Turning Barriers into Opportunities for Collaborative Design. In: deCindio F, Schuler D (eds) Proceedings of the Participatory Design Conference (PDC'04). CPSR, P.O. Box 717, Palo Alto, CA 94302, University of Toronto, Canada, July, pp 152-161
Fischer G (2009a) Cultures of Participation and Social Computing: Rethinking and Reinventing Learning and Education. In: Proceedings of the International Conference on Advanced Learning Technologies (ICALT). IEEE Press, Riga, Latvia, pp 1-5
Fischer G (2009b) End-User Development and Meta-Design: Foundations for Cultures of Participation. In: Pipek V, Rossen MB, deRuyter B, Wulf V (eds) End-User Development. Springer, Heidelberg, pp 3-14
Fischer G (2010) End-User Development and Meta-Design: Foundations for Cultures of Participation. Journal of Organizational and End User Computing 22 (1):52-82
Fischer G (2011) Social Creativity: Exploiting the Power of Cultures of Participation. Paper presented at the 7th International Conference on Semantics, Knowledge and Grids, Beijing, China
Fischer G, Giaccardi E (2006) Meta-Design: A Framework for the Future of End User Development. In: Lieberman H, Paternò F, Wulf V (eds) End User Development. Kluwer Academic Publishers, Dordrecht, The Netherlands, pp 427-457
Fischer G, Giaccardi E, Ye Y, Sutcliffe AG, Mehandjiev N (2004a) Meta-Design: A Manifesto for End-User Development. Communications of the ACM 47 (9):33-37
Fischer G, Girgensohn A (1990) End-User Modifiability in Design Environments. In: Human Factors in Computing Systems, (CHI'90) (Seattle, WA). ACM, New York, pp 183-191
Fischer G, Girgensohn A, Nakakoji K, Redmiles D (1992) Supporting Software Designers with Integrated, Domain-Oriented Design Environments. IEEE Transactions on Software Engineering, Special Issue on Knowledge Representation and Reasoning in Software Engineering 18 (6):511-522
Fischer G, Grudin J, McCall R, Ostwald J, Redmiles D, Reeves B, Shipman F (2001) Seeding, Evolutionary Growth and Reseeding: The Incremental Development of Collaborative Design Environments. In: Olson GM, Malone TW, Smith JB (eds) Coordination Theory and Collaboration Technology. Lawrence Erlbaum Associates, Mahwah, NJ, pp 447-472
Fischer G, Herrmann T (2011) Sociotechnical Systems: A Meta-Design Perspective. International Journal for Sociotechnology and Knowledge Development 3 (1):1-33
Fischer G, Lemke AC (1988) Construction Kits and Design Environments: Steps Toward Human Problem-Domain Communication. Human-Computer Interaction 3 (3):179-222
Fischer G, McCall R, Ostwald J, Reeves B, Shipman F (1994) Seeding, Evolutionary Growth and Reseeding: Supporting Incremental Development of Design Environments. In: Adelson B, Dumais S, Olson J (eds) Proceedings of ACM Conference on Human Factors in Computing Systems (CHI'94), vol 1. ACM, New York, pp 292-298

Fischer G, Ostwald J Seeding, Evolutionary Growth, and Reseeding: Enriching Participatory Design with Informed Participation. In: Binder T, Gregory J, Wagner I (eds) Proceedings of the Participatory Design Conference (PDC'02), Malmö University, Sweden, 2002. CPSR, pp 135-143

Fischer G, Scharff E Meta-Design—Design for Designers. In: Boyarski D, Kellogg W (eds) 3rd International Conference on Designing Interactive Systems (DIS 2000), New York, 2000a. ACM, pp 396-405

Fischer G, Scharff E (2000b) Meta-design: design for designers. Paper presented at the Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques, New York City, New York, United States,

Fischer G, Scharff E, Ye Y (2004b) Fostering Social Creativity by Increasing Social Capital. In: Huysman M, Wulf V (eds) Social Capital and Information Technology. MIT Press, Cambridge, MA, pp 355-399

Florida R (2002) The Rise of the Creative Class and How It's Transforming Work, Leisure, Community and Everyday Life. Basic Books, New York, NY

Floyd C (1987) Outline of a Paradigm Change in Software Engineering. In: Bjerknes G, Ehn P, Kyng M (eds) Computers and Democracy: A Scandinavian Challenge. Avebury, Aldershot, UK, pp 191-210

Fogli D, & Piccinno, A. (2005) Environments to support context and emotion aware visual interaction. IJVLC 16:386-405

Fogli D, Fresta G, Mussio P (2004) On electronic annotation and its implementation. Paper presented at the Proceedings of the working conference on Advanced visual interfaces, Gallipoli, Italy,

Fong A, Valerdi R, Srinivasan J (2007) Boundary Objects as a Framework to Understand the Role of Systems Integrators. Systems Research Forum 2 (1):11-18

Fraser M, Glover T, Vaghi I, Benford S, Greenhalgh C, Hindmarsh J, Heath C (2000) Revealing the realities of collaborative virtual reality. Paper presented at the Proceedings of the third international conference on Collaborative virtual environments, San Francisco, California, United States

Frere-Jones S (2005) POP MUSIC 1+1+1= 1 The new math of mashups. The New Yorker.

Fried L (2011) Tweet-a-Watt. http://www.ladyada.net/make/tweetawatt/. 2011

Fugelli P (2010) Intersubjectivity and objects of knowledge: Making sense across sites in software development. PhD thesis, University of Oslo, Oslo

Gal U, Yoo Y, Boland RJ (2004) The dynamics of boundary objects, social infrastructures and social identities. Sprouts: Working Papers on Information Systems 4:193-206

Galison P (1997) Image & logic: A material culture of microphysics. The University of Chicago Press, Chicago

Gantt M, Nardi BA (1992) Gardeners and Gurus: Patterns of Cooperation Among CAD Users. In: Bauersfeld P, Bennett J, Lynch G (eds) Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems. ACM, New York, pp 107-117

Ghosh RA (2006) Code: Collaborative Ownership and the Digital Economy. The MIT Press, Cambridge, MA

Giaccardi E (2004) Principles of Metadesign: Processes and Levels of Co-Creation in the New Design Space. Ph.D. Dissertation, CAiiA-STAR, School of Computing, Plymouth, UK

Ginige JA, Silva BD, Ginige A (2005) Towards End User Development of Web Applications for SMEs: A Component Based Approach. LNCS 3579 3579/2005:489-499. doi:10.1007/11531371_62

Girgensohn A (1992) End-User Modifiability in Knowledge-Based Design Environments. Ph.D. Dissertation, University of Colorado at Boulder, Boulder, CO

GitHub (2008). https://github.com/. Accessed 03. Jan 2011

Glaser BG, Strauss AL (1967) The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine, Chicago

Gold R (1958) Roles in sociological field observation. Social Forces 36:217-213

Gordon C (2006) Wikis--a disruptive innovation. KMWorld Magazine. http://www.kmworld.com/Articles/News/News-Analysis/Wikis--a-disruptive-innovation-15802.aspx. Accessed 12 Sept. 2011

Gough HG (1979) A creative personality scale for the Adjective Check List. Personality and Social Psychology 37 (8)
Greenbaum J, Kyng M (eds) (1991) Design at Work: Cooperative Design of Computer Systems. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ
Greif I (ed) (1988) Computer-Supported Cooperative Work: A Book of Readings. Morgan Kaufmann Publishers, San Mateo, CA
Grudin J (1989) Why Groupware Applications Fail: Problems in Design and Evaluation. Office: Technology and People 4 (3):245-264
Grudin J (1994a) Computer-Supported Cooperative Work: History and Focus. IEEE Computer 27 (5):19-26
Grudin J (1994b) Groupware and social dynamics: eight challenges for developers. Commun ACM 37 (1):92-105. doi:10.1145/175222.175230
Grudin J (2006) Enterprise Knowledge Management and Emerging Technologies. Paper presented at the Proceedings of the 39th Annual Hawaii International Conference on System Sciences - Volume 03
Grudin J, Palen L (1995) Why groupware succeeds: discretion or mandate? Paper presented at the Proceedings of the fourth conference on European Conference on Computer-Supported Cooperative Work, Stockholm, Sweden,
Grudin J, Poole ES Wikis at work: success factors and challenges for sustainability of enterprise Wikis. In: WikiSym 2010, NY, USA, 2010. In Proceedings of the 6th International Symposium on Wikis and Open Collaboration (WikiSym '10). ACM, pp 1-8. doi:10.1145/1832772.1832780
Guilford JP (1950) Creativity. American Psychologist 5:444-454
Hallam E, Ingold T (2007) Creativity and Cultural Improvisation. Association of Social Anthropologists Monographs. Berg Publishers, Oxford
Harel I, Papert S (eds) (1991) Constructionism : research reports and essays, 1985-1990, by the Epistemology & Learning Research Group, the Media Laboratory, Massachusetts Institute of Technology. Ablex Publishing Corporation, Norwood, NJ
Hawkes T (1977) Structuralism and Semiotics. University of California Press
Hebdige D (1979) Subculture: The Meaning of Style. Routledge
Henderson A, Kyng M (1991) There's No Place Like Home: Continuing Design in Use. In: Greenbaum J, Kyng M (eds) Design at Work: Cooperative Design of Computer Systems. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, pp 219-240
Henderson K (1991) Flexible Sketches and Inflexible Data Bases: Visual Communication, Conscription Devices, and Boundary Objects in Design Engineering. Science Technology Human Values 16, No. 4:448-473
Herrmann T (2008) Socio-technical Appropriation of Web2.0 for continuing learning on the job. Paper presented at the CSCW San Diego, CA, USA,
Herrmann T (2009a) The Socio-Technical Walkthrough (STWT): A Means Of Research-Oriented Intervention Into Organizations. Sprouts Working Papers on Information Systems 9
Herrmann T (2009b) Systems Design with the Socio-Technical Walkthrough. In: Handbook of Research on Socio-Technical Design and Social Networking Systems: (2-volumes). IGI Global. doi:10.4018/978-1-60566-264-0.ch023
Herrmann T (2010) Support of Collaborative Creativity for co-located Meetings. In: Randall DS, Pascal (ed) From CSCW to Web 2.0: European Developments in Collaborative Design. Computer Supported Cooperative Work. Springer London, London, pp 65-95. doi:10.1007/978-1-84882-965-7_4
Herrmann T, Carell A, Nierhoff J (2011) Creativity barometer: an approach for continuing micro surveys to explore the dynamics of organization's creativity climates. Paper presented at the Proceedings of the 8th ACM conference on Creativity and cognition, Atlanta, Georgia, USA
Herrmann T, Loser K-U, Jahnke I (2007) Socio-technical Walkthrough (STWT): a means for Knowledge Integration. International Journal of Learning Organisation 14 (5):450-464
Herz J (2002) Harnessing the Hive: How Online Games Drive Networked Innovation. Release 10 20 (9):1-21

Hevner A, Chatterjee S (2010) Introduction to Design Science Research. In: Hevner A, Chatterjee S (eds) Design Research in Information Systems: Theory and Practice, vol 22. vol Integrated Series in Information Systems. p 320
Hevner AR, March ST, Park J, Ram S (2004) Design science in information systems research. MIS Q 28 (1):75-105
Hewitt C (1986) Offices are open systems. ACM Trans Inf Syst 4 (3):271--287. doi:http://doi.acm.org/10.1145/214427.214432
Hillier BaL, A (1979) Architecture as a discipline. Architectural Research 5 (1):28–32
Hoffman R, Blue A, Guericke K, Ly E, Vaillant J-L (2009) LinkedIn. linkedin.com. Accessed 23. Aug. 2010
Hopkins J (2000) Component primer. Commun ACM 43 (10):27-30. doi:10.1145/352183.352198
Houston D, Ferdowsi A (2007) Dropbox. dropbox.com. Accessed 26. Nov. 2010
Hult M, Lennung, S. (1980) Towards a definition of action research: A note and bibliography. Journal of Management Studies 17:241-250
Hurley C, Chen S, Karim J (2005) YouTube. http://www.youtube.com/. Accessed 11. Nov 2011
Hutchins E (1995) Cognition in the Wild. The MIT Press, Cambridge
Hutchins EL, Hollan JD, Norman DA (1986) Direct Manipulation Interfaces. In: Norman DA, Draper SW (eds) User Centered System Design, New Perspectives on Human-Computer Interaction. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, pp 87-124
iGoogle (2007). http://www.google.com/ig. Accessed 12. March 2011
Illich I (1973) Tools for Conviviality. Harper and Row, New York
Ingalls D, Palacz K, Uhler S, Taivalsaari A, Mikkonen T (2008) The Lively Kernel A Self-supporting System on a Web Page. In: Self-Sustaining Systems. Springer-Verlag, pp 31-50
Innes JE, Booher DE (1999) Consensus Building as Role Playing and Bricolage. the American Planning Association 65 (1):9-25
Instruments N (1983) LabVIEW. http://www.ni.com/labview/whatis/.
Jenkins H (2006) Confronting the Challenges of Participatory Cultures: Media Education for the 21st Century. http://www.henryjenkins.org/2006/10/confronting_the_challenges_of.html.
Jennings P Tangible Social Interfaces: Critical Theory, Boundary Objects, and Interdisciplinary Design Methods. In: Proceedings of Creativity & Cognition, London, April, 2005. ACM, pp 176-186
Jones D (2006) Building and integrating a small office intranet. Linux J 2006 (151):8
jQuery (2006). http://jquery.com/. Accessed 03. November 2010
jsFiddle (2010). http://jsfiddle.net/. Accessed 30. Nov 2010
Kapor M (2006) Architecture is Politics (and Politics is Architecture). Mitch Kapor's Blog.
Kellogg WA (2007) Supporting Collaboration in Distributed Teams: Implications for e-Research (Contribution to the ECSCW Workshop "Realising and Supporting Collaboration in e-Research"). http://www.e-researchcommunity.org/docs/ecscw07/submissions/Kellogg.pdf.
Kelly K (1994) Out of Control: The New Biology of Machines, Social Systems, & the Economic World. Basic Books Cambridge, Massachusetts
Kensing F, Madsen KH (1991) Generating Visions: Future Workshops and Metaphorical Design. In: Greenbaum JaK, M. (ed) Design at Work - Cooperative Design of Computer Artifacts. Lawrence Erlbaum, Hillsdale, NJ, pp 155-168
Klein M, Sayama H, Faratin P, Bar-Yam Y (2003) The Dynamics of Collaborative Design: Insights From Complex Systems and Negotiation Research. Concurrent Engineering 11 (2):201-209
Knuth DE (1983) Literate Programming. Department of Computer Science, Stanford University, Stanford, CA
Konkola R (2001) Harjoittelun kehittämisprosessi ammattikorkeakoulussa ja rajavyöhyketoiminta uudenlaisena toimintamallina. In: Tuomi-Gröhn T, Engeström Y, Young M (eds) Koulun ja työn rajavyöhykkeellä. Uusia työssäoppimisen mahdollisuuksia. University Press, Helsiniki, pp 148-186

Koskinen I, Battarbee K (2003) Introduction to user experience and empathic design. In: Suri JF, Battarbee, K., and Koskinen, I. (ed) Empathic Design: User Experience in Product Design. IT Press, Helsinki, pp 37-50
Krahn R, Ingalls D, Hirschfeld R, Lincke J, Palacz K (2009) Lively Wiki a development environment for creating and sharing active web content. Paper presented at the Proceedings of the 5th International Symposium on Wikis and Open Collaboration, Orlando, Florida
Kruse HCJ, Slagter R, Hofte Ht (2000) Collaborative Component Software: The CoCoWare framework and its application. Telematica Instituut, Enschede, The Netherlands
Kvan T (2000) Collaborative design: what is it? Automation in Construction 9 (4):409-415
Laventure D (2011) Bricolage. International Collaborative Dictionary of Communications.
Lawson B (1979) Cognitive Strategies in Architectural Design. Ergonomics 22 (1):59-68
Leadbeater C, Miller P (2004) The pro-am revolution: how enthusiasts are changing our society and economy. Demos
Lee CP (2005) Between chaos and routine: boundary negotiating artifacts in collaboration. Paper presented at the Proceedings of the ninth conference on European Conference on Computer Supported Cooperative Work, Paris, France,
Lee CP (2007) Collaborative Design and the Science of Design. Paper presented at the ACM Conference on Human Factors and Usability (CHI), San Jose, California,
Legard R, Keegan J, Ward K (2003) In-depth interview. In: Ritchie JaL, J. (ed) Qualitative research practice A Guide for Social Science Students and Researchers. Sage Publications, pp 139–169
Lehrich C (2005) RPGs and Bricolage: Theory and Practice. http://www.indie-rpgs.com/archive/index.php?topic=14371. Accessed 09. September 2011
Leitheiser R, Wetherbe, J. (1986) Service Support Levels: An Organizational Approach to End-User Computing. MIS Quarterly 10 (4):337-349
Leonard D, Rayport JF (1997) Spark Innovation Through Empathic Design. Harvard Business Review, vol 76.
Lerner RM (2004) At the forge: Bricolage templates. Linux Journal 2004 (119)
Leuf B, Cunningham W (2001) The Wiki Way: Collaboration and Sharing on the Internet. Addison-Wesley Professional, Massachusetts
Lévi-Strauss C (1968) The Savage Mind. University Of Chicago Press
Lewis CH, Olson GM (1987) Can the Principles of Cognition Lower the Barriers of Programming? In: Olson GM, Soloway E, Sheppard S (eds) Empirical Studies of Programmers, vol 2. Ablex Publishing Corporation, Lawrence Erlbaum Associates, Norwood, NJ, pp 248-263
Lieberman H (ed) (2001) Your Wish is My Command: Programming by Example. Morgan Kaufmann, San Francisco
Lieberman H, Liu H (2006) Feasibility Studies for Programming in Natural Language. In: Lieberman H, Paterno F, Wulf V (eds) End-User Development. Springer, Dordrecht,
Lieberman H, Paternó F, Klann M, Wulf V (2006) End-User Development: an Emerging Paradigm In: Lieberman H, Paterno F, Wulf V (eds) End User Development. Springer, Dordrecht, pp 1--8
Lipsitz L, Reisner T (1973) The computer and education, vol 9. Educational Technology
Loarne SL (2005) Bricolage versus creativity what's the difference? Paper presented at the 21th EGOS (European Group for Organizational Studies) colloquium, Berlin
LSL (2003). http://wiki.secondlife.com/wiki/LSL_Portal. Accessed 18. May 2011
Lu IM, Mantei MM (1991) Idea management in a shared drawing tool. Paper presented at the Proceedings of the second conference on European Conference on Computer-Supported Cooperative Work, Amsterdam, The Netherlands
Ludicorp (2004) Flickr. http://www.flickr.com/. Accessed 06. April 2010
Lutters WG, Ackerman MS (2002) Achieving safety: a field study of boundary objects in aircraft technical support. Paper presented at the Proceedings of the 2002 ACM conference on Computer supported cooperative work, New Orleans, Louisiana, USA
Lutters WG, Ackerman MS (2007) Beyond Boundary Objects: Collaborative Reuse in Aircraft Technical Support. Comput Supported Coop Work 16 (3):341-372. doi:10.1007/s10606-006-9036-x

MacIsaac D (1995) An Introduction to Action Research. http://physicsed.buffalostate.edu/danowner/actionrsch.html.
Mackay WE (1990) Patterns of Sharing Customizable Software. In: Halasz F (ed) Proceedings of ACM CSCW'90 Conference on Computer-Supported Cooperative Work. ACM, New York, pp 209-221
MacLean A, Carter K, Lovstrand L, Moran T (1990) User-tailorable systems: pressing the issues with buttons. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people, Seattle, Washington, United States
Madiajagan M, Vijayakumar B (2006) Interoperability in Component Based Software Development. World Academy of Science, Engineering and Technology (22):68-76
Majhew DJ (1992) Principles and Guideline in Software User Interface Design. Prentice Hall, Englewood Cliffs, NJ
Malone TW, Lai K-Y, Fry C (1992) Experiments with Oval: A Radically Tailorable Tool for Cooperative Work. In: Turner J, Kraut R (eds) Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work. ACM, New York, pp 289-297
Mamykina L, Candy L, Edmonds E (2002) Collaborative creativity. Commun ACM 45 (10):96-99
Marcante A, Provenza LP (2008) Social Interaction through Map-based Wikis. PsychNology Journal 6:247-267
March ST, Smith GF (1995) Design and natural science research on information technology. Decision Support Systems 15 (4):251–266
Mariani C (2012) Recommender System in Ambienti Collaborativi: Uno Studio Sperimentale. TESI DI LAUREA DI Università degli Studi di Milano, Milan
Marlow C, Naaman M, Boyd D, Davis M (2006) HT06, tagging paper, taxonomy, Flickr, academic article, to read. Paper presented at the Proceedings of the seventeenth conference on Hypertext and hypermedia, Odense, Denmark
Martin J (1983) Managing the data-base environment. Prentice-Hall
Masanori S, Kazuhiro H, Hiromichi H (2004) Caretta: a system for supporting face-to-face collaboration by integrating personal and shared spaces. Proceedings of the SIGCHI conference on Human factors in computing systems 1-58113-702-8. ACM, Vienna, Austria
Mathes A (2004) Folksonomies-Cooperative Classification and Communication Through Shared Metadata. University of Illinois Urbana- Champaign, US
Mattessich PW, Monsey BR (1992) Collaboration--what makes it work: a review of research literature on factors influencing successful collaboration. Amherst H Wilder Foundation, Saint Paul, MN
Mayer RE (1983) Thinking, problem solving, cognition. W.H. Freeman and Company, New York
mediaWiki (2002). http://www.mediawiki.org/wiki/Help:Templates.
Miller DS, Smith JG, Muller MJ (1992) TelePICTIVE: computer-supported collaborative GUI design for designers with diverse expertise. Proceedings of the 5th annual ACM symposium on User interface software and technology 0-89791-549-6. ACM, Monteray, California, United States
Miller F, Stiksel M, Breidenbruecker M, Willomitzer T (2002) last.fm. last.fm. Accessed 16. March 2010
Miller J (1999) Jabber/XMPP. http://www.jabber.org/. Accessed 30. Dec. 2011
Miller J, Glassner B (1997) The inside and outside: finding realities in interviews. In: Silverman D (ed) Qualitative Research: Theory, Method and Practice. Sage, London,
Mills KL (2003) Computer-supported cooperative work (CSCW). Computer-Supported Cooperative Work
MockingBird (2009). https://gomockingbird.com/. Accessed 08.Sep 2010
Mørch A (1995) Application units: Basic building blocks of tailorable applications. Human-Computer Interaction 1015:45-62. doi:10.1007/3-540-60614-9_4
Mørch A (1996) Evolving a Generic Application into a Domain-oriented Design Environment. Scandinavian Journal of Information Systems 8 (2):63-89

Mørch A (1997) Three Levels of End-User Tailoring: Customization, Integration, and Extension. In: Kyng M, Mathiassen L (eds) Computers and Design in Context. MIT Press, Cambridge, MA, pp 51-76
Mørch A (1998) Tailoring Tools for System Development. Journal of End User Computing 10 (2):22-30
Mørch A (2001) Component-based design and software readymades. InterMedia, University of Oslo. http://imweb.uio.no/seminarer/designingdesign/Component_based_design.htm. Accessed 10. April 2012
Mørch A, Andersen R (2009) Mutual development: A case Study in Customer-initiated Software Product Development. Paper presented at the Presentation at IS-EUD 2009, Siegen
Mørch A, Stevens G, Won M, Klann M, Dittrich Y, Wulf V (2004a) Component-based technologies for end-user development. Communications of the ACM 47:59-62
Mørch AI (2003) Aspect-Oriented Software Components Evolutionary Growth and Control in User Tailorable Systems. In: Adaptive Evolutionary Information Systems. Idea Group Publishing, Hershey PA, USA, pp 105-124
Mørch AI (2007) The Evolution of Design Critics Towards Collaborative Learning: A Socio-Technical Approach.
Mørch AI (2011a) Evolutionary Application Development: Tools to Make Tools and Boundary Crossing. In: Pekkola HIS (ed) Reframing Humans in Information Systems Development. Springer, pp 151 - 171
Mørch AI (2011b) Externalized Design: Expressing Social Ideas in User Interfaces. In: Coakes E (ed) Knowledge Development and Social Change through Technology: Emerging Studies. 1 edition edn. IGI Global, Hersey, PA, pp 64-84
Mørch AI, Andersen R (2010) Mutual Development: The Software Engineering Context of End-User Development. Journal of Organizational and End User Computing 22 (2):36-57
Mørch AI, Åsand H-RH, Ludvigsen SR (2007) The Organization of End User Development in an Accounting Company. In: End User Computing Challenges and Technologies: Emerging Tools and Applications. Information Science Reference, pp 102-123
Mørch AI, Engen BK, Åsand H-RH (2004b) The Workplace as a Learning Laboratory: The Winding Road to E-learning in a Norwegian Service Company. ACM Press,
Mørch AI, Mehandjiev ND (2000) Tailoring as Collaboration: The Mediating Role of Multiple Representations and Application Units. Computer Supported Cooperative Work 9 (1):75-100
Muller MJ, Kuhn S (1993) Participatory Design. Communications of the ACM - Special Issue on Participatory Design 36 (6):24-28
Mumford E (1987) Sociotechnical Systems Design: Evolving Theory and Practice. In: Bjerknes G, Ehn P, Kyng M (eds) Computers and Democracy. Avebury, Aldershot, UK, pp 59-76
Mussio P, Pietrogrande, M., Protti, M. (1991) Simulation of Hepatological Models: a Study in Visual Interactive Exploration of Scientific Problems. Journal of Visual Languages and Computing 2:75-95
Myers BA, Kosbie DS (1996) Reusable hierarchical command objects. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems: common ground, Vancouver, British Columbia, Canada,
Myers BA, Pane JF, Ko A (2004) Natural programming languages and environments. Commun ACM 47 (9):47-52. doi:10.1145/1015864.1015888
Nakakoji K, Yamamoto Y, Nishinaka Y, Kishida K, Ye Y Evolution Patterns of Open-Source Software Systems and Communities. In: Proceedings of International Workshop on Principles of Software Evolution (IWPSE 2002), Orlando, FL, May 19-20 2002. pp 76-85
Nardi B (1996) Context and Consciousness and Consciousness Activity and Human-Computer Interactions. The MIT Press, Cambridge, Massachusetts
Nardi BA (1993) A small matter of programming: perspectives on end user computing. MIT Press

Nardi BA (1997) The Use of Ethnographic Methods in Design and Evaluation. In: Helander MG, Landauer TK, Prabhu PV (eds) Handbook of Human-Computer Interaction, vol 1. second edn. Elsevier Science B.V., Amsterdam, pp 361-366
Nardi BA, Miller JR (1990) A Ethnographic Study of Distributed Problem Solving in Spreadsheet Development. In: Halasz F (ed) Proceedings of ACM CSCW'90 Conference on Computer-Supported Cooperative Work. ACM, New York, pp 197-208
Nardi BA, Miller JR (1991) Twinkling Lights and Nested Loops: Distributed Problem Solving and Spreadsheet Development. International Journal of Man-Machine Studies 34:161-184
Netvibes (2005). http://www.netvibes.com/en. Accessed 05. Feb 2011
Newman MW, Sedivy JZ, Neuwirth CM, Edwards WK, Hong JI, Izadi S, Marcelo K, Smith TF (2002) Designing for serendipity: supporting end-user configuration of ubiquitous computing environments. Paper presented at the Proceedings of the 4th conference on Designing interactive systems: processes, practices, methods, and techniques, London, England
Nielsen J (1993) Usability Engineering. Academic Press, Boston, MA
Node.js (2009). http://nodejs.org/. Accessed 12. June 2011
Norbert AS, J, rg G, ler, Torsten H, Shin'ichi K, Christian M, ller T, Wolfgang R, Petra R, Peter S, Ralf S (1999) i-LAND: an interactive landscape for creativity and innovation. Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit 0-201-48559-1. ACM, Pittsburgh, Pennsylvania, United States
Norman DA (1990) The Design of Everyday Things. Currency Doubleday, New York
Norman DA (1993) Things That Make Us Smart. Addison-Wesley Publishing Company, Reading, MA
Norman DA, Draper SW (eds) (1986) User-Centered System Design, New Perspectives on Human-Computer Interaction. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ
Nunamaker JJF, Briggs RO, Mittleman DD, Vogel DR, Balthazard PA (1996) Lessons from a dozen years of group support systems research: a discussion of lab and field findings. J Manage Inf Syst 13 (3):163-207
Nygaard K Program Development as a Social Activity. In: Kugler H-J (ed) Proceedings of Information Processing 86, 1986. North-Holland, pp 189--198
O'Day VL, Bobrow DG, Shirley M (1996) The social-technical design circle. Paper presented at the Proceedings of the 1996 ACM conference on Computer supported cooperative work, Boston, Massachusetts, United States,
O'Reilly T (2006) What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.
Olsen DR, Jr., Buxton W, Ehrich R, Kasik DJ, Rhyne JR, Sibert J (1984) A Context for User Interface Management. In: IEEE Computer Graphics and Applications. pp 33-42
Oppermann R (ed) (1994) Adaptive User Support. Lawrence Erlbaum, Hillsdale, New Jersey
Orlikowski WJ (1996) Evolving the notes: organizational change around groupware technology. In: Groupware and teamwork. John Wiley & Sons, Inc., pp 23-59
Orlikowski WJ (2007) Using technology and constituting structures: A practice lens for studying technology in organizations. In: Ackerman MSaH, Christine A. and Erickson, Thomas and Kellogg, Wendy A. (ed) Resources, Co-Evolution and Artifacts Theory in CSCW (Computer Supported Cooperative Work). Springer-Verlag New York, Secaucus, NJ, USA
Osborn AF (1963) Applied imagination: principles and procedures of creative problem-solving. Scribner, New York
Ousterhout JK (1998) Scripting: Higher-Level Programming for the 21st Century. Computer 31 (3):23-30. doi:10.1109/2.660187
Palen L (1999) Social, Individual & Technological Issues for Groupware Calendar Systems. In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 99). Pittsburgh, pp 17-24
Papert S (1993) The Children's Machine: Rethinking School in the Age of the Computer. Basic Books
Pariser E (2011) The Filter Bubble: What the Internet Is Hiding from You. Penguin Press HC, New York

Pelto PJ (1970) Anthropological research: The structure of inquiry. Harper & Row, New York
Petre M, Blackwell AF (2007) Children as Unwitting End-User Programmers. Paper presented at the Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing
Petre M, Green TRG (1993) Learning to Read Graphics: Some Evidence that Seeing an Information Display is an Acquired Skill. Journal of Visual Languages and Computing (4):55-70
Pettinari CaH, C. (1998) Notes toward an Applied Ethnography. Work Interaction and Technology Research Group. Kings College London
Piccinno A (2005) Software environments for supporting End-User Development. University of Bari, Bari
Piirainen K, Kolfschoten G, Lukosch S Unraveling Challenges in Collaborative Design: A Literature Study. In: Groupware: Design, Implementation, and Use 15th International Workshop, Peso da Régua, Douro, Portugal, 2009. Lecture Notes in Computer Science, Vol. 5784.
Pipek V (2005) From Tailoring to Appropriation Support: Negotiating Groupware Usage. PhD Thesis, University of Oulu, Oulu, Finland
Polanyi M (1966) The Tacit Dimension. Doubleday, Garden City, NY
Popovic V (1996) Design activity structural categories. In: N. Cross HCaKD (ed) Analysing Design Activity. Wiley Press, Sussex, pp 211-220
Powell A, Moore JE (2002) The focus of research in end user computing: Where have we come since the 1980ties? Journal of End User Computing 14 (1):3-22
Prilla M, Herrmann, T. (2007) Semantically Integrating Heterogeneous Content: Applying Social Tagging as a Knowledge Management Tool for Process Model Development and Usage. Paper presented at the 11th International Conference on Knowledge Management and Knowledge Technologies, Messe Congress Graz, Austria,
Randall D, Harper R, Rouncefield M (2007) Fieldwork for Design Theory and Practice Computer supported cooperative work
Razavi R (2010) Web Pontoon: a method for reflective web applications. Paper presented at the International Workshop on Smalltalk Technologies, Barcelona, Spain
Repenning A, Ioannidou A (2006a) Mobility Agents: Guiding and Tracking Public Transporation Users. In: Proceedings of the AVI Conference 2006. Venice, p (this volume)
Repenning A, Ioannidou A (2006b) What makes End-User Development Tick? 13 design guidelines. In: Lieberman HP, Fabio; Wulf, Volker (ed) End User Development. Human–Computer Interaction Series, vol 9. Springer, pp 51-85
Repenning A, Ioannidou A, Phillips J (1999) Collaborative Use & Design of Interactive Systems. In: Hoadley C (ed) Proceedings of the Computer Supported Collaborative Learning (CSCL '99) Conference. Stanford University, Palo Alto, CA, pp 475-487
Resnick M, Myers B, Nakakoji K, Shneiderman B, Randy Pausch, Selker T, Eisenberg M (2005) Design Principles for Tools to Support Creative Thinking. IJHCI, 36 edn.,
Rice AK (1958) Productivity and social organisation: The Ahmedabad experiment. Tavistock, London
Riehle D, In (2006) How and Why Wikipedia Works: An Interview with Angela Beesley, Elisabeth Bauer, and Kizu Naoko. In: Proceedings of the 2006 International Symposium on Wikis (WikiSym '06). ACM, New York, NY, pp 3-8
Rittel H (1984) Second-Generation Design Methods. In: Cross N (ed) Developments in Design Methodology. John Wiley & Sons, New York, pp 317-327
Rittel H, Webber MM (1973) Dilemmas in a General Theory of Planning. Policy Science:155-169
Rittel H, Webber MM (1984) Planning Problems are Wicked Problems. In: Cross N (ed) Developments in Design Methodology. John Wiley & Sons, New York, pp 135-144
Rogers Y (1994) Exploring obstacles: integrating CSCW in evolving organisations. Paper presented at the Proceedings of the 1994 ACM conference on Computer supported cooperative work, Chapel Hill, North Carolina, United States
Rommetveit R (1974) On Message Structure: A Framework for the Study of Language and Communication. John Wiley & Sons Ltd
Rose K (2004) Digg. http://digg.com/. Accessed 13. Oct. 2010

Ruby (1996). http://ruby-lang.org/. Accessed 05. May 2011
Sanders EB-N (2000) Generative Tools for CoDesigning. In: Scrivener BaW (ed) Collaborative Design, vol 1. vol 2. Springer-Verlag, London, pp 3-12
Sanders EB-N, Stappers PJ (2008) Co-creation and the new landscapes of design. CoDesign 4 (1):5-18
Sandkuhl K, Nentwig L, Manhart S, Lafrenz P Redesigning CSCW-systems for network computing-experience from the HotCon project. In: Proceedings of the Sixth Euromicro Workshop on Parallel and Distributed Processing (PDP '98), Madrid, Spain 1998. pp 318 - 324
Saul G, David M (1994) Real time groupware as a distributed system: concurrency control and its effect on the interface. Proceedings of the 1994 ACM conference on Computer supported cooperative work. ACM, Chapel Hill, North Carolina, United States
Schachter J (2003) Delicious delicious.com. Accessed 28. Aug. 2010
Schmidt K (1991) Riding a tiger, or computer supported cooperative work. Paper presented at the Proceedings of the second conference on European Conference on Computer-Supported Cooperative Work, Amsterdam, The Netherlands,
Schön DA (1983) The Reflective Practitioner: How Professionals Think in Action. Basic Books, New York
Schön DA (1987) Educating the Reflective Practitioner. Jossey-Bass, San Francisco, CA
Schuler D, Namioka A (eds) (1993) Participatory Design: Principles and Practices. Lawrence Erlbaum Associates, Hillsdale, NJ
Scriven M (1967) The methodology of evaluation. In: R. W. Tyler RMG, & M. Scriven (ed) Perspectives of curriculum evaluation. IL: Rand McNally, Chicago, pp 39-83
Scrivener SAR, Ball, L. J., and Woodcock, W. Collaborative Design. In: Proceedings of CoDesigning 2000, Coventry, England, 2001. Springer-Verlag
Seitamaa-Hakkarainen P, Minna U (2006) Facilitating social creativity through collaborative designing. Paper presented at the Proceedings of the 7th international conference on Learning sciences, Bloomington, Indiana
Shen W, Hao Q, Li W (2008) Computer supported collaborative design: Retrospective and perspective. Computers in Industry 59:855–862
Sherman RRaW, R.B. (1988) Qualitative Research in Education: Focus and Methods. Routledge, London
Shirky C (2010) Cognitive Surplus — Creativity and Generosity in a Connected Age. Penguin Press, New York, N.Y.
Shneiderman B (2000) Creating creativity: user interfaces for supporting innovation. ACM Transactions on Computer Human-Interaction 7 (1):114-138
Shneiderman B (2007) Creativity Support Tools: Accelerating Discovery and Innovation. Communications of the ACM 50 (12):20-32
Shneiderman B, Fischer G, Czerwinski M, Resnick M, Myers B (2006) Creativity Support Tools: Report From a U.S. National Science Foundation Sponsored Workshop. International Journal Of Human–Computer Interaction 20 (2):61–77
Simon HA (1996) The Sciences of the Artificial. third edn. The MIT Press, Cambridge, MA
Sinatra (2008). http://www.sinatrarb.com/ Accessed 08. April 2011
SnipSnap (2003). http://www.snipsnap.org/. Accessed 06. September 2011
Snow CP (1993) The Two Cultures. Cambridge University Press, Cambridge, UK
Snowdon D, Greenhalgh, Chris, Benford, Steve What You See Is Not What I See: Subjectivity in Virtual Environments. In: Framework for Immersive Virtual Environments (FIVE'95), London, UK 1995.
Spahn M, Dörner, C., Wulf, V. (2008) End User Development: Approaches Towards a Flexible Software Design. Paper presented at the Proc. of ECIS 2008,
Sproull L, Kiesler S, Kiesler SB (1991) Connections: new ways of working in the networked organization.
Sprow E (1992) Chrysler's Concurrent Engineering Challenge. Manufacturing Engineering 108 (4):35-42
Star SL (1989) The structure of ill-structured solutions: boundary objects and heterogeneous distributed problem solving. In: Gasser L, Huhns MN (eds) Distributed artificial intelligence, vol 2. Morgan Kaufmann Publishers Inc., pp 37-54

Star SL, Griesemer JR (1989) Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. Social Studies of Science 19 (3):387-420
Stevens G (2009) Understanding and Designing Appropriation Infrastructures: Artifacts as boundary objects in the continuous software development. Universität Siegen, Siegen
Stevens G, Pipek V, Wulf V (2009) Appropriation Infrastructure: Supporting the Design of Usages. Paper presented at the Proceedings of the 2nd International Symposium on End-User Development, Siegen, Germany,
Stiemerling O (2000) Component-Based Tailorability. PhD, University of Bonn, Bonn
Subrahmanian E, Monarch I, Konda S, Granger H, Milliken R, Westerberg A, Group TN-D (2003) Boundary Objects and Prototypesat the Interfaces of Engineering Design. Comput Supported Coop Work 12 (2):185-203. doi:10.1023/a:1023976111188
Suchman LA (1985) Plans and Situated Actions: The Problem of Human-Machine Communication. Xerox Palo Alto Research Center, Palo Alto, CA
Suchman LA (1987) Plans and Situated Actions. Cambridge University Press, Cambridge, UK
Surowiecki J (2005) The Wisdom of Crowds. Anchor Books, New York
Sutcliffe AG, Mehandjiev N (2004) End User Development. Communications of ACM 47 (9):31-32
Szyperski C (2002) Component Software: Beyond Object-Oriented Programming. Addison-Wesley Longman Publishing Co., Inc., London
Tapscott D, Williams AD (2006) Wikinomics: How Mass Collaboration Changes Everything. Portofolio, Penguin Group, New York, NY
Tatsubori M, Suzumura T (2009) HTML templates that fly: a template engine approach to automated offloading from server to client. Paper presented at the Proceedings of the 18th international conference on World wide web, Madrid, Spain,
Taylor DW, Berry PC, Block CH (1958) Does Group Participation When Using Brainstorming Facilitate or Inhabit Creative Thinking. Administrative Science Quarterly 3 (1)
Thackara J (2005) In the Bubble: designing in a complex world. The MIT Press, London, England
Thoeny P (1998) TWiki - the Open Source Enterprise Wiki and Web 2.0 Application Platform. http://twiki.org/. Accessed 08. Nov. 2011
Tomkins C (1996) Duchamp: A Biography. Henry Holt
Trigg RH, Moran TP, Halasz FG (1987) Adaptability and Tailorability in NoteCards. In: Bullinger H, Shackel B (eds) Proceedings of INTERACT'87, 2nd IFIP Conference on Human-Computer Interaction (Stuttgart, FRG). North-Holland, Amsterdam, pp 723-728
Trist EL (1981) The Sociotechnical Perspective: The Evolution of Sociotechnical Systems as a Conceptual Framework and as an Action Research Program. In: VanDeVen AH, Joyce WF (eds) Perspectives on Organization Design and Behavior. Wiley, New York, NY
Tropes T (2004) Television Tropes and Idioms. http://tvtropes.org/pmwiki/pmwiki.php/Main/HomePage. Accessed 12 Jan 2012
Tsui ABM, Wong JLN (2009) In Search of a Third Space: Teacher Development in Mainland China. In: Chan CKK, Rao N (eds) Revisiting the Chinese Learner: Changing Contexts, Changing Education. Springer, Hong Kong, p 376
Tuomi-Gröhn T, Engeström Y, Young M (2003) From Transfer to Boundary-crossing Between School and Work as a Tool for Developing Vocational Education: An Introduction. In: Between school and work: new perspectives on transfer and boundary-crossing. Emerald Group Publishing Limited, pp 360-366
van Aken JE (2007) Design Science and Organization Development Interventions Aligning Business and Humanistic Values. Applied Behavioral Science 43 (1):67-88
Verbeke G (2001) A Future Focus on Collaborative Design. Collaborative Architectural Design
Victor B, Boynton AC (1998) Invented Here: Maximizing Your Organization's Internal Growth and Profitability. Harvard Business Review Press
Vitharana P (2003) Risks and challenges of component-based software development. Commun ACM 46 (8):67-72. doi:10.1145/859670.859671

Volkoff O, Strong D, Elmes M (2002) Between a Rock and a Hard Place: Boundary Spanners in an Erp Implementation. Paper presented at the Americas Conference on Information Systems, Dallas, TX
von Hippel E (2005) Democratizing Innovation. MIT Press, Cambridge, MA
Wakkary R (2009) Anything is a Fridge: The Implications of Everyday Design. Interactions September/October:12-17
Wales J, Sanger L (2010) Wikipedia. http://en.wikipedia.org/wiki/Wikipedia. Accessed 19. Dec. 2009
Wallas G (1926) The art of thought. Harcourt, Barce & World, New York
Wanstrath C, Hyett P, Preston-Werner T (2008) GitHub. https://github.com/. Accessed 03. Jan 2010
Warr A, O'Neill E (2005) Understanding design as a social creative process. Paper presented at the Proceedings of the 5th conference on Creativity & cognition, London, United Kingdom
Webb B, Webb S (1932) Methods of Social Study. Longmans Green, London
Weick KE (1993) Organization Redesign as Improvisation. In: Glick IGPHaWH (ed) Organizational Change and Redesign Ideas and Insights for Improving Performance. Oxford University Press, New York, pp 346-379
Wenger E (1998) Communities of Practice: Learning, Meaning, and Identity. Cambridge University Press, UK
Wenger EaM, Richard and Snyder, William (2002) Cultivating Communities of Practice: A Guide to Managing Knowledge. Harvard Business School Press, Boston, MA, USA
Wertsch JV (1998) Mind as Action. Oxford University Press, New York, USA
Wikipedia (2005) XBee. http://en.wikipedia.org/wiki/XBee. Accessed 12. Dec. 2011
Wikipedia (2008) Kill A Watt. http://en.wikipedia.org/wiki/Kill_A_Watt. Accessed 28. Nov 2011
Wong J, Hong J (2008) What do we "mashup" when we make mashups? Paper presented at the Proceedings of the 4th international workshop on End-user software engineering, Leipzig, Germany
Wulf V, Golombek B (2001) Direct activation: a concept to encourage tailoring activities. Behaviour and Information Technology 4 (1):249-263
Wulf V, Pipek V, Won M (2008) Component-based tailorability: Enabling highly flexible software applications. Int J Hum-Comput Stud 66 (1):1-22. doi:10.1016/j.ijhcs.2007.08.007
Wulf V, Rohde M (1995) Towards an integrated organization and technology development. Paper presented at the Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods & techniques, Ann Arbor, Michigan, United States
Xwiki (2003). http://www.xwiki.org/xwiki/bin/view/Main/WebHome. Accessed 21. Oct. 2011
Yakura EK (2002) Charting time: Timelines as temporal boundary objects. Academy of Management Journal 45, No. 5:956-970.
Ye Y (2001) Supporting Component-Based Software Development with Active Component Repository Systems. Ph.D. Dissertation, University of Colorao at Boulder, Boulder, Colorado
Ye Y, Fischer G (2007) Designing for Participation in Socio-Technical Software Systems. In: Stephanidis C (ed) Proceedings of 4th International Conference on Universal Access in Human-Computer Interaction (Beijing, China). Springer, Heidelberg, pp 312-321
Youngblood G (1986) Metadesign: Toward a Postmodernism of Reconstruction. Linzer Veranstaltungsgesellschaft. http://90.146.8.18/en/archives/festival_archive/festival_catalogs/festival_artikel.asp?iProjectID=9210.
Zhu L A Meta-design Framework to Support Multidisciplinary Teams' Online Collaboration. In: Costabile MF, Dittrich Y, Fischer G, Piccinno A (eds) IS-EUD 2011, Torre Canne, 2011. End-User Development. Springer Berlin / Heidelberg, pp 403-406. doi:10.1007/978-3-642-21530-8_51
Zhu L, Barricelli BR, Iacob C (2011a) A Meta-design Model for Creative Distributed Collaborative Design. Journal of Distributed Systems and Technologies 2 (4)

Zhu L, Herrmann T (2012a) Design Now! – Elaborating Requirements in Situated Action. Paper presented at the CreaRE 2012 in conjunction with REFSQ 2012, Essen, Germany, 19. March
Zhu L, Herrmann T (2012b) A Meta-design Approach to Interactive Spaces. Paper presented at the Designing Collaborative Interactive Spaces for e-Creativity, e-Science and e-Learning workshop in AVI2012, Capri, Italy
Zhu L, Iacob C, Barricelli BR New Design Strategies: Using the Hive Mind Space Model to Enhance Collaboration. In: IADIS Multi Conference on Computer Science and Information Systems 2010, Freiburg, 2010a. pp 12-19
Zhu L, Mussio P, Barricelli BR (2010b) Hive-mind space model for creative, collaborative design. Paper presented at the Proceedings of the 1st DESIRE Network Conference on Creativity and Innovation in Design, Aarhus, Denmark
Zhu L, Mussio, P., Barricelli, B.R. and Iacob, C. (2010) A habitable space for supporting creative collaboration. Paper presented at the Collaborative Technologies and Systems (CTS 2010), Chicago
Zhu L, Vaghi IR, Barricelli BR A Meta-reflective Wiki for Collaborative Design. In: International Symposium on Wikis and Open Collaboration, WikiSym2011, California, 2011b. ACM,
Zuckerberg M, Saverin E, Moskovitz D, Hughes C (2004) Facebook. http://www.facebook.com/. Accessed 03.Dec 2011