Identifying, Relating, and Evaluating Design Patterns for the Design of Software for Synchronous Collaboration

Claudia Iacob

Department of Computer Science

University of Milan

A thesis submitted for the degree of

Doctor of Philosophy

2012

To my father.

Abstract

Many working environments require that geographically distributed or co-located work group members work together - supported by software - in developing and refining one commonly shared resource in the same time. In Computer-Supported Collaborative Work, this is defined as *synchronous collaboration*. Domains subject to such type of collaboration are many, some such examples are drawing, searching, text editing, or game solving. Technology has helped switching from the real to the virtual, simplifying such collaborative efforts and providing technological support for more efficient and faster collaborations. Several software applications, developed as either research projects or commercial products, exist and are used today in synchronous collaborative settings in domains such as drawing, searching, text editing, and game solving. In order to further support such developments, there is a growing need for knowledge capturing and sharing with respect to the challenges and the concerns to be faced in designing such tools. Surely, documentations of existing applications help, but it is only a larger and perhaps more general repository of knowledge that would do a better job.

Several approaches for building such repositories of knowledge exist, the approach this thesis is further exploring is *design patterns*. On one hand, I aim at bringing methodological support to design pattern research in answer to the scarce landscape of methods and techniques for both identifying design patterns in interaction design and generating pattern languages based on existing collections of patterns. I focus mainly on one area of interaction design, i.e. the design of applications addressing synchronous collaboration, and I target four domains in the area, i.e. drawing, text editing, searching, and game solving. On the other hand, I am interested in better understanding how design patterns are used and what is the impact of using them in collaborative design processes. I first focus on the collaborative processes involving novice designers, aiming to correlate the findings from this initial study with those obtained after investigating similar processes involving experienced users of patterns.

This work primarily impacts design pattern research at a methodological, theoretical, practical and empirical level. Secondarily, the findings described throughout the thesis inform behavioural research and human-computer interaction. At a methodological level, I describe two methods addressing design pattern research; one is used for identifying design patterns in interaction design, while the second one is used for generating pattern languages out of existing collections of patterns. At a theoretical level, I describe the results of applying the pattern identification method in the area of synchronous collaboration, providing 15 design patterns addressing the design of applications targeting this area. Practically, I developed a software application able to support the semi-automation of the pattern language generation method and the execution of queries on the output of this generation process. At the empirical level, I present a case study designed to bring some light into the matter of collaborative use of design patterns. The results of this case study aim at identifying strategies novice designers develop while collaboratively using design patterns during interaction design processes.

Acknowledgements

I would like to thank all those who have helped, guided, and inspired me throughout this process. First and foremost, I would like to extend a huge thank you to Thomas Herrmann, Corina Sas, and Paolo Bottoni for the valuable suggestions and feedback on the thesis. Thank you!

I would also like to thank Daniela Fogli, Loredana Parasiliti Provenza, Ernesto Damiani, and Piero Mussio for their willingness to support and guide me through this work, providing me with lots of insights.

Many thanks to all the participants in the workshops for the inspiring ideas and great designs and to Alessandra Agostini, Giuseppe Boccignone, and Stefano Valtolina for the support.

It's been great meeting and exchanging ideas with Nicole Schadewitz, Nina Gellersen, Irmi Wachendorff, Travis Purrington (at the Swiss Design Network 2010 workshop); Peter Baumgartner, Klaus Marquardt and the EuroPLoP2011 workshop participants - Christian Köppe, Reinhard Bauer, Dirk Schnelle-Walka; Hanyuda Eiiti, Joseph Yoder, and the AsianPLoP2011 workshop participants - Eduardo B. Fernandez, Rebecca Rikner, David West, Raj Datta; Kiran Kumar on the PLoP2011 submission; Elisa Giaccardi, Gerhard Fischer, Peter Forbrig, and the participants in the EICS2011 and ISEUD2011 DCs; Julie Zhu and all the DESIRE fellows; the group in Bochum led by Thomas Herrmann; the student volunteers from MobileHCI2010, ECSCW2011, and CSCW2012; all the participants in the two DESIRE summer schools.
Thank you all, you've made this journey absolutely captivating!

La final, dar in primul rand, multumesc familiei mele fara de care nu as fi scris azi randurile acestea.

Publications

The list of publications from this work includes:

- Iacob, C. 2012. Using Design Patterns in Collaborative Interaction Design Processes. ACM International Conference on Computer Supported Cooperative Work (CSCW'12), pp. 107-110.
- Iacob, C. 2010. Design Patterns as Tools to Support Social Creativity and Knowledge Management in Collaborative Design Processes. Journal of Information & Knowledge Management, World Scientific Publishing Co., Vol. 10, No. 4, pp. 343-350.
- Iacob, C., Fogli, D. 2011. Connecting Patterns: An Ontology Approach to a Pattern Language Definition. ACM International Conference on Pattern Languages of Programs.
- Iacob, C., Damiani, E. 2011. On the Use of Design Patterns in Collaborative Design Processes. ACM International Conference on Creativity and Innovation in Design, DESIRE'11, pp. 245-254.
- Iacob, C. 2011. Strategies in the Collaborative Use of Design Patterns. Collaborative usage and development of models and visualizations Workshop, ECSCW2011.
- Iacob, C. 2011. Scenario-Based Design in Design Pattern Mining. 18th International Conference on Engineering Design (ICED2011), vol. 6, pp. 1-10.
- Iacob, C. 2011. Designing Systems for Synchronous Collaboration: From Collaborative Software to Design Patterns. 16th European Conference on Pattern Languages of Programs (EuroPLoP2011).
- Iacob, C. 2011. Does Software Speak Creativity? Creative Processes and Techniques in the Design of Software Applications. Create'11 Interaction Design Symposium.

- Iacob, C. 2011. A Design Pattern Mining Method for Interaction Design. ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS2011), pp. 217-222.
- Iacob, C. 2011. Identifying, Relating, and Evaluating Design Patterns for the Design of Software for Synchronous Collaboration. ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS2011) (doctoral consortium) pp. 323-326.
- Iacob, C. 2011. Design Patterns in the Design of Systems for Creative Collaborative Processes. International Symposium on End-User Development (IS-EUD2011) (doctoral consortium) pp. 359-362.
- Iacob, C. 2011. Mining for Patterns in the Design of Systems for Synchronous Collaboration. Asian Conference on Pattern Languages of Programs (AsianPLoP2011).
- 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), pp. 112-119.
- Iacob, C., Mussio, P., Zhu, L., Barricelli, B.R. 2010. Towards a Pattern Language for the Design of Collaborative Interactive Systems. Workshop on Visual Formalism for Patterns, IEEE Symposium on Visual Languages and Human-Centric Computing, EASST, Vol. 31.
- Iacob, C. 2010. Design Patterns and Web Interactive Systems for Supporting Creative Collaborative Design Processes. DESIRE'10 - Creativity and Innovation in Design (poster).

Contents

List of Figures

Chapter 1

Introduction

The goal of this thesis is twofold. On one hand, I aim at bringing methodological support to design pattern research in answer to the scarce landscape of methods and techniques for both identifying design patterns in interaction design and generating pattern languages based on existing collections of patterns. I focus mainly on one area of interaction design, i.e. the design of applications addressing synchronous collaboration, and I target four domains in the area, i.e. drawing, text editing, searching, and game solving. On the other hand, I am interested in better understanding how design patterns are used and what is the impact of using them in collaborative design processes. I first focus on the collaborative processes involving novice designers, aiming to correlate the findings from this initial study with those obtained after investigating similar processes involving experienced users of patterns.

1.1 Landscape: Research Context

Many working environments require that geographically distributed or co-located work group members work together - supported by software - in developing and refining one commonly shared resource in the same time. In Computer-Supported Collaborative Work, this is defined as *synchronous collaboration*. Domains subject to such type of collaboration are many, some such examples are drawing, searching, text editing, or game solving.

- Architects, graphic designers, artists and even software analysts often gather together round the same table, sketching their ideas through drawings, diagrams, and sketches, negotiating and discussing based on them. These processes are highly collaborative since their complexity exceeds the capabilities of one single individual and requires the collaboration with others [43]. Moreover, such processes often require the presence of all those involved.

- Searching, on the other hand, has long been portrayed as an individual activity - each individual querying a topic, without any intervention from others. However, in the absence of collaborative means for searching, librarians, researchers or even groups of individuals - interested in, for example, planning a trip together - have come up with various workarounds to turn searching into a collaborative task [84]. Emails with intermediate search results sent back and forth between members of a group or instant messaging throughout a searching session are a few examples of such workarounds. Recent studies have identified other such strategies and researchers have started showing a large interest in the direction of collaborative synchronous search [85], [84], [97].

- Text editing seen as a collaborative activity is not a surprise for anyone. Whether it is a collaborative paper writing effort [6], or a collaborative activity of creating the summary of a lecture, text editing processes require the jointly development and refinement of one resource (the edited document) by more people. Often collaboratively edited documents are sent back and forth among the contributors via e-mail or other such means, leading to what Adler et al. define as ”crude asynchronous manner for jointly editing office suite documents” [6]. The consequences of such working strategies could easily lead to inconsistencies in the document and confusion among those who edit it. In addition to that, having each contributor working on his/her own version of the document, makes the problem of integration of all contributions even more complex. In answer to that, both research and industry have focused on collaborative synchronous text editing, as well.

- Game solving is probably one of the best examples of synchronous collaboration. Played for entertainment or as competitions, games bring together round the

> same board more players all having the same goal in mind - reaching the solution of the game. Consider, for example, a group of friends trying to solve a puzzle together. They have a common goal - that of reaching a complete puzzle - and several constraints - there is only one puzzle they are building together and the pieces forming the puzzle are available to all of them. They all share the puzzle board, gradually reaching the final solution of the game. In a similar line of thought, consider crossword solving. The goal in collaborative crossword solving is reaching a complete filled in crossword and the game is open to contributions from more than one player. Therefore, the crossword board is shared, each contributor being allowed to add something to it. Due to their character, games are most often played in the presence of all the players involved.

Technology has helped switching from the real to the virtual, simplifying such collaborative efforts and providing technological support for more efficient and faster collaborations. However, most research and development of technology to support collaboration has been directed towards asynchronous collaboration contexts [72] in which collaborators work together, but not in the same time, time not being an issue. Therefore, today, there is a growing interest in supporting synchronous interaction, as well. Throughout this work, synchronous collaboration refers to both co-located and remote collaboration. Surely, there are differences between the two and these differences could bring to light various other challenges, but these are beyond the scope of this thesis. Several software applications, developed as either research projects or commercial products, exist and are used today in synchronous collaborative settings in domains such as drawing, searching, text editing, and game solving. In order to further support such developments, there is a growing need for knowledge capturing and sharing with respect to the challenges and the concerns to be faced in designing such tools. Surely, documentations of existing applications help, but it is only a larger and perhaps more general repository of knowledge that would do a better job.

In requirements engineering, the requirements of particular software applications are captured through structures such as use cases or task descriptions. While a use case describes a dialogue between the user and the system, modelling the interaction provided by the system, a task description defines what the user and the system do together [66]. A task description details the problem for which a solution is needed; customers may come up with solution examples in terms of what the system might do. On the other hand, a use case can only describe those problems for which the analyst can illustrate a solution. Moreover, use cases do not distinguish between the requirements and the solution.

During design phases, several approaches for capturing knowledge are used today, even if documenting knowledge is often considered a resource-intensive process often skipped or performed inadequately [113]. Guidelines are an example of structures which help capturing design knowledge and support the establishment of a clear design process. Formal and semi-formal models such as UML diagrams are used for modelling the central design-knowledge artefacts explicitly. Such models can be further used to generate code through model-driven software development or other similar techniques. Documentation regarding design rationale and design-decision is, in some cases, added to the system documentation. Also, techniques for reverse engineering of existing systems capture design knowledge embedded in such system. Using design patterns as tools to capture and share knowledge is common to both software engineering and design.

The approach this thesis is further exploring is *design patterns*. The concept was initially defined in architecture [7], [9] as a tool for documenting best practices in the area of urban design. The idea was soon adopted by other fields, including software engineering and Human-Computer Interaction (HCI), and more particular *interaction design*. The main idea behind the concept of design pattern has not changed across fields, even if the template of defining patterns and their audience may differ from one field to another. Software engineering uses design patterns to express Object-Oriented software design practices. Such patterns address communities of OO software developers and are defined in a code-oriented manner, being always accompanied by the snippet of source code which illustrates the application of the pattern. An example of such a pattern is Singleton which describes with illustrative code examples the solution for enforcing a class to have only one instance. The complete list of the patterns described in software engineering is further provided in Section 2.2.

Interaction design views patterns as ways to ”document best practice solutions to support user interface designers in their daily work in order to improve their productivity and make the design process more efficient” [64]. The audience of interaction design patterns, i.e. design patterns[1] are communities of designers focused on graphical user interface (GUI) and interaction design. A design pattern is defined by a problem, the proven solution to tackle the problem and any other relevant information which would make this pair (problem, solution) understandable. Several areas of interaction design have been subject to being documented through patterns, some example including web design [103], social interfaces [30], accessibility [45]. Collaboration is not an exception, patterns for designing for collaboration have been written by Schadewitz et al. [90], or Lukosch and Schummer [94]. While Lukosch and Schummer look at collaboration from a technological perspective (considering issues such as letting users identify themselves before using an application, rewarding positive participation of groups, allowing users to collaborate over the use of files), addressing the design of applications which support collaboration, Schadewitz et al. take a different stand, addressing behavioural patterns in cross-culture collaboration. None of the available collections of patterns specifically address synchronous collaboration.

A particularly interesting aspect related to design patterns is that they may act as boundary objects [100], [26], [99] in that they support the interaction with, negotiation around and building upon an idea [41]. Design patterns facilitate communication among collaborating designers, mediating communication gaps. For example, faced with a challenging task, designers may turn to patterns in order to make themselves understood to one another, to explain each other concepts and to identify the problems to be faced and their proposed workarounds.

Usually, a collection of patterns contains inter-related patterns. Since patterns document proven solutions to recurring design problems [22] and design problems are never isolated, the patterns documenting them are not defined in isolation one from another. A structure formed of a collection of patterns and all the relationships between these patterns is defined as a *pattern language* [8]. The interesting thing about pattern languages is that they help pattern users navigate within a collection of patterns

[1]Throughout this thesis, design patterns will refer to interaction design patterns.

and better understand the area the collection addresses. An isolated pattern brings valuable insight about a specific problem, but it does not necessarily support moving beyond that and understanding the implications of applying the solution proposed for it. This is precisely where the pattern language comes into place, connecting the dots and allowing one to easily grasp the relationships between patterns.

There are several scenarios in which this is helpful. First, patterns are written with the purpose of *capturing knowledge*. Grasping and understanding this knowledge asks for unfolding the individual concepts and the relationships between them. Hence, traversing the collection supported by an underlying logic helps creating representations of the area addressed by the collection. Secondly, patterns are written with the purpose of *describing recurring problems and proven solutions to tackle them*. Hence, their goal is to support designers in reusing proven solutions and not having to reinvent the wheel. Using a solution, however, does not always limit to that and often asks for considering related problems as well. Support in finding such related problems is brought by the relationships between the patterns documenting them. Lastly, patterns are written with the purpose of *sharing knowledge* helping to build an integrated repository of knowledge. Adding to such a repository asks for connecting newly acquired knowledge to the existing base, such connections needing to be explicit. Pattern languages are by definition such structures, allowing the identification of relationships between different bodies of knowledge.

1.2 Gap: Research Motivation and Questions

A closer look at the existing collections of patterns brought to light several limitations with respect to *design pattern research*. Some of these limitations are addressed by this thesis and they include:

1. The process followed by pattern writers for identifying patterns is rarely described. At a closer look, the methodological landscape with respect to pattern mining is scarce and rarely subject to generalization. The lack of structured methods to be used in identifying patterns for interaction design pointed to an

open research question: *How to identify design patterns for interaction design?* Further on, the definition of such methods needs evaluation cycles able to support the evolution and optimization of the method itself. Such evaluation cycles should target different areas of interaction design and, possibly, a comparative analysis of the method's application across such areas. In order to add to the already existing collections of patterns addressing collaborative application design and for answering the lack of a collection specifically addressing synchronous collaborative applications, one such evaluation cycle would need to consider the area of synchronous collaboration.

2. Most of the existing collections of patterns are described together with the relationships identified between these patterns. Such structures are called *pattern languages*. Even if most of the authors present their patterns in the form of a language - connecting the patterns and sometimes describing the relationships identified between them - none of them describes the process followed to get to these relationships. This creates the need for the definition of a method to support pattern authors in automatically generating a language out of a collection of patterns. Therefore, an additional research question addressed by this thesis is: *How to generate pattern languages from collections of design pattern?*

3. The methodological support needs to be accompanied by tool support able to make the application of such methods easier and more efficient. Only a close analysis of the method can provide the requirements for a tool to support its application. However, the general question to be addressed is: *What tool support is needed for the application of the methods described above?*

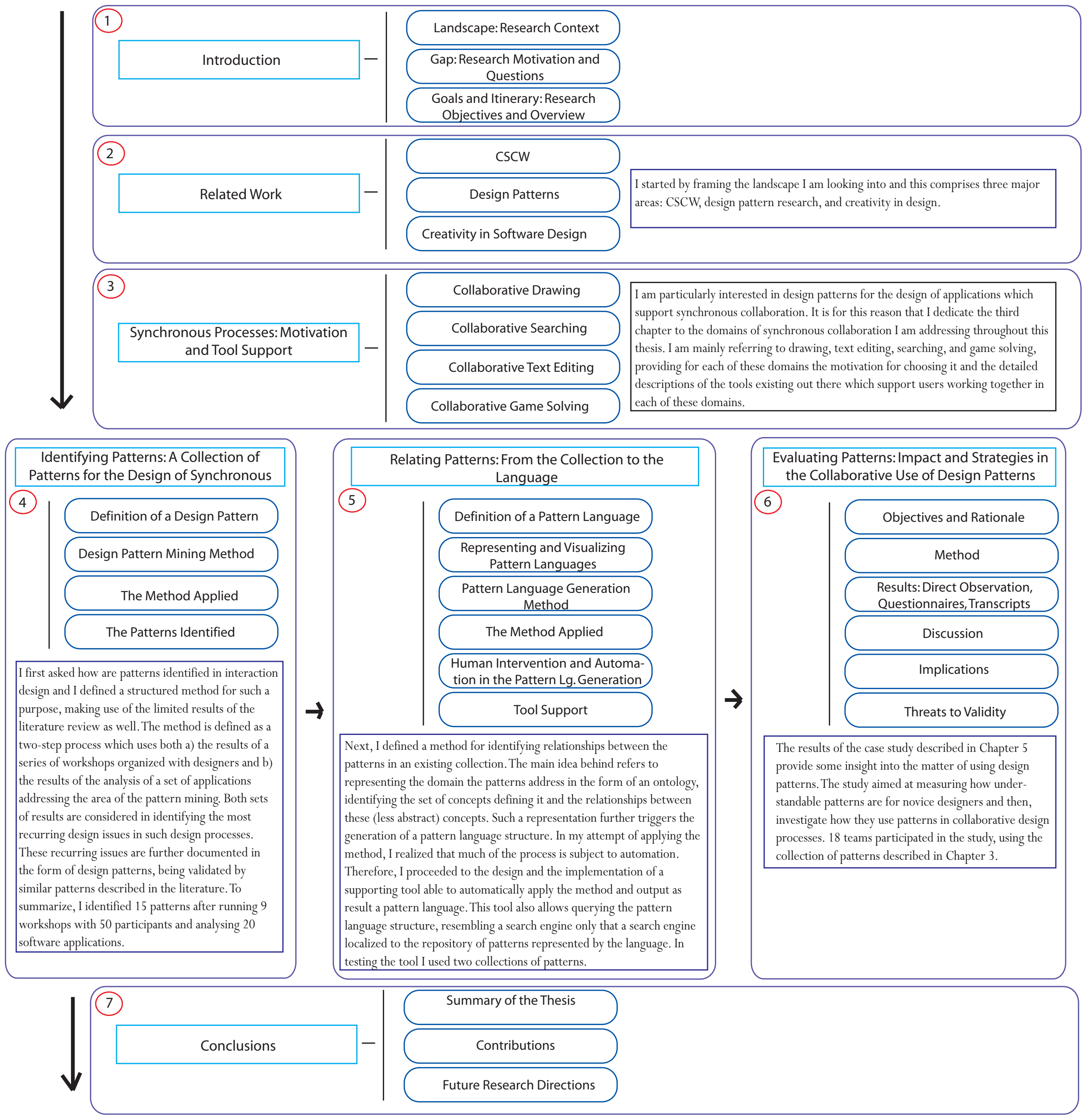
4. There is an abundance of available collections of design patterns. One might ask how these patterns are used and, moreover, by whom. The few studies run in this direction brought to light some valuable findings. However, due to the empirical character of such work, many more inquiries need to be investigated and, possibly, answered in order to frame an understanding of the behavioural strategies

developed by pattern users and of the impact of using patterns in the first place. Some such inquiries include: *How are design patterns used in collaborative design processes?*, *How do designers perceive the use of design patterns in collaborative design processes?*

1.3 Goals and Itinerary: Research Objectives and Overview

Surely, answering all the questions addressed previously requires long-term collaborative efforts. The goal of this thesis is to address some of these questions and possibly suggest others; therefore, in this section I point out how far the present work goes, what goals it tries to accomplish and what issues it leaves open (Figure 1.1). A more detailed discussion on the contributions brought by this thesis is presented in Section 7.2. In a first instance, I go through existing literature in order to identify and frame the landscape of the work related to what this thesis addresses. The literature review focuses mainly on three areas: CSCW, design patterns and creativity in software design. With respect to CSCW, I look into the major challenges collaborative contexts impose to the design of software applications able to support such contexts. Surely such a list could very well stand for the topic of a brand new thesis and it is for that reason that my interest mainly relates to those challenges covered by the work I present further on in the thesis.

In terms of design patterns, I provide a bit of historical background able to clarify how the concept came to life and how it was further adopted. I look into available definitions of design patterns and collections widely known in various domains and design areas. Also, I describe work done in providing methodological support in identifying design patterns and in empirical research aiming to investigate the usage of design patterns in actual design processes. Since design patterns are rarely defined in isolation, most of the authors structuring them in pattern languages, I look into the concept of pattern language and in ways to generate such structures from existing pattern collections.


1
Introduction
Landscape: Research Context
Gap: Research Motivation and Questions
Goals and Itinerary: Research Objectives and Overview
2
Related Work
CSCW
Design Patterns
Creativity in Software Design
3
Synchronous Processes: Motivation and Tool Support
Collaborative Drawing
Collaborative Searching
Collaborative Text Editing
Collaborative Game Solving
Identifying Patterns: A Collection of Patterns for the Design of Synchronous
4
Definition of a Design Pattern
Design Pattern Mining Method
The Method Applied
The Patterns Identified
Relating Patterns: From the Collection to the Language
5
Definition of a Pattern Language
Representing and Visualizing Pattern Languages
Pattern Language Generation Method
The Method Applied
Human Intervention and Automation in the Pattern Lg. Generation
Tool Support
Evaluating Patterns: Impact and Strategies in the Collaborative Use of Design Patterns
6
Objectives and Rationale
Method
Results: Direct Observation, Questionnaires, Transcripts
Discussion
Implications
Threats to Validity
7
Conclusions
Summary of the Thesis
Contributions
Future Research Directions

I started by framing the landscape I am looking into and this comprises three major areas: CSCW, design pattern research, and creativity in design.

I am particularly interested in design patterns for the design of applications which support synchronous collaboration. It is for this reason that I dedicate the third chapter to the domains of synchronous collaboration I am addressing throughout this thesis. I am mainly referring to drawing, text editing, searching, and game solving, providing for each of these domains the motivation for choosing it and the detailed descriptions of the tools existing out there which support users working together in each of these domains.

I first asked how are patterns identified in interaction design and I defined a structured method for such a purpose, making use of the limited results of the literature review as well. The method is defined as a two-step process which uses both a) the results of a series of workshops organized with designers and b) the results of the analysis of a set of applications addressing the area of the pattern mining. Both sets of results are considered in identifying the most recurring design issues in such design processes. These recurring issues are further documented in the form of design patterns, being validated by similar patterns described in the literature. To summarize, I identified 15 patterns after running 9 workshops with 50 participants and analysing 20 software applications.

Next, I defined a method for identifying relationships between the patterns in an existing collection. The main idea behind refers to representing the domain the patterns address in the form of an ontology, identifying the set of concepts defining it and the relationships between these (less abstract) concepts. Such a representation further triggers the generation of a pattern language structure. In my attempt of applying the method, I realized that much of the process is subject to automation. Therefore, I proceeded to the design and the implementation of a supporting tool able to automatically apply the method and output as result a pattern language. This tool also allows querying the pattern language structure, resembling a search engine only that a search engine localized to the repository of patterns represented by the language. In testing the tool I used two collections of patterns.

The results of the case study described in Chapter 5 provide some insight into the matter of using design patterns. The study aimed at measuring how understandable patterns are for novice designers and then, investigate how they use patterns in collaborative design processes. 18 teams participated in the study, using the collection of patterns described in Chapter 3.

Figure 1.1: The Map: Overview of the Thesis

Much of the work described in this thesis revolves around interaction design processes. In order to support and possibly enhance such processes, I look into creative techniques and models documented in the literature and able to fit the context of software design. Some of the techniques investigated (i.e. scenario-based design, mockups, and sketches) are used further on during the data collection phases described by this thesis. Even if not directly related to the results obtained through this work, creative techniques and models helped shape the methodology used in getting to these results. The discussion on creativity and its impact on the work presented in this thesis concludes the literature review and opens the way to the third chapter, entirely dedicated to the synchronous processes of interest to this work.

Since I am particularly interested in four domains subject to synchronous collaboration (namely drawing, searching, text editing, and game solving), in Chapter 3, I describe the motivation for choosing each of these domains. Moreover, I look into the efforts developed in implementing software applications to support synchronous collaboration in each of these domains. I briefly describe a subset of these efforts as a collection of 20 applications - commercial products or research projects.

As the title of the thesis suggests, the core of this work revolves around three major goals: 1) identifying, 2) relating, and 3) evaluating design patterns. Each goal is further discussed in a separate chapter, as follows:

1. Chapter 4 describes a method - together with the process of its application - for identifying design patterns in interaction design. The method is defined by both a series of workshops involving designers asked to design applications in the area of the pattern mining and an analysis of a collection of software applications supporting the area targeted by the mining. The workshops bring together teams of designers and a facilitator. Provided with a design task, the designers are observed in action and the design issues they address are collected. The application analysis aims at identifying those design issues recurring in the implementations of these applications. The overall aim of the method is to identify those commonly recurring design issues in both the design processes followed by designers and in actual implementations already running and used. I describe

the application of the method in the area of synchronous collaboration and the results of this application, i.e. a collection of 15 design patterns.

At a methodological level, I used design workshops for several reasons. On one hand, I aimed at observing the participants involved in the workshops in action. That allowed them to focus entirely on the design task and not feel the burden of remembering or abstracting previous or present design tasks. On the other hand, the design issues discussed by them were brought to light without any external bias and they were not required to analyse their process, but just follow it. Hence, their focus was on the task itself and not on the final goal of the whole study - i.e. identifying patterns. In addition to that, during the workshops the participants were encouraged to use several creative techniques able to support them in exploring and externalizing their ideas and design options - scenario-based design supported the design space exploration, while sketching and mockups helped them externalize their ideas and communicate based on them.

2. Chapter 5 describes a method - together with the implementation of a tool to support the method's application - for relating the design patterns in a collection. The method aims at building an ontology for representing the domain addressed by the collection of patterns and triggering the generation of a structure comprising these patterns and the relationships between them on the basis of this ontology-based representation. Applying the method on the collection of patterns described in Chapter 4 brings to light several areas subject to automation. As result of that, I designed and implemented a tool able to generate a pattern language from an existing collection of patterns using the method described above and to support the execution of queries on the result of this generation process. As test cases for the tool, I use two already existing collections of patterns, addressing two different design areas - synchronous collaboration and GUI design.

3. Chapter 6 describes a case study aiming to evaluate the usage of design patterns in collaborative design processes involving novice designers. For that, I run a series of design workshops during which I ask teams of novice designers

to use design patterns in their design processes. My interest is to identify the strategies the teams use in working with the patterns and to assess the impact using the patterns has on the overall design process, both in terms of how the teams perceived it and how it is actually captured through the transcripts. The patterns used during the workshops are those identified and described in Chapter 4. The reason for this is two-fold. On one hand, I aimed at evaluating the patterns themselves through the workshops and based on the feedback received from the participants I rewrote and clarified their content. On the other hand, the collection has a medium size and fits well the context designed for the workshops.

A summary of the thesis together with a detailed discussion on the contributions provided by this work and the future directions brought to light by it are addressed in the last chapter.

Chapter 2

Related Work: CSCW, Design Patterns and Creativity

This chapter goes through existing literature in order to identify and frame the landscape of the work related to what this thesis addresses. The literature review focuses mainly on three areas: CSCW, design patterns and creativity in software design. With respect to CSCW, I look into the major challenges collaborative contexts impose to the design of software applications able to support such contexts. In terms of design patterns, I am interested in documented methods for identifying and relating patterns and in studies run for understanding how patterns are used. Moreover, since much of this thesis covers interaction design processes, I also look into creative techniques and models documented in the literature and able to fit the context of software design.

2.1 Computer Supported Cooperative Work

Back in 1988, the originators of the term "Computer Supported Cooperative Work", Irene Greif and Paul Cashman, commented that they "coined the phrase partly as a shorthand way of referring to a set of concerns about supporting multiple individuals working together with computer systems" [49]. However, this loose description, as noted by Liam Bannon, hardly allows the emergence of a coherent research area. Therefore, the proposal of CSCW as an "umbrella term" rephrases this definition as:

"What at first sight might appear to be a weakness of the field, having such a diversity of backgrounds and perspectives, is seen by us as a potential strength, if utilized properly. We believe that for the moment the name CSCW simply serves as a useful forum for a variety of researchers with different backgrounds and techniques to discuss their work, and allows for the cross-fertilization of ideas, for the fostering of multi-disciplinary perspectives on the field that is essential if we are to produce applications that really are useful." [17]

Developing as a research area, CSCW is trying to answer questions such as:

- What are the specific characteristics of cooperative work as opposed to work performed by individuals in seclusion?
- What are the reasons for the emergence of cooperative work patterns? [16]
- How can computer-based technology be applied to enhance cooperative work relations?
- How can computers be applied to alleviate the logistic problems of cooperative work?
- How should designers approach the complex and delicate problems of designing systems that will shape social relationships?

Therefore, the two-fold focus of CSCW comprises: a) understanding the nature and the characteristics of cooperative work, and b) designing computer-based technologies to support it [16]. In close relation to the discussion above is the recurring question: What should be included in the category of "groupware" or "CSCW applications"? As pointed out in [50], "categorization of an application is less helpful than considering its use in a particular setting". A straightforward example is a mail management system which would hardly belong to the groupware category if only used as a means of broadcasting messages within an organization.

2.1.1 Classifications and Modes

A categorization of groupware is adapted by Grudin who uses two dimensions: time and space (Figure 2.1). Therefore, collaboration can be carried out in a single place, or in more places. Similarly, collaborators can be working together in the same time (i.e. "in one unbroken interval") or at different times. Based on these categories, several types of groupware were assigned to each cell in Figure 2.1 . For example, collaborative activities developed in the same time and in the same place could benefit from tools to support meeting facilitation. Collaboration carried out in the same time and where collaborators are located in different places can be supported by video conferencing tools. In a similar line of reasoning, collaborative activities performed at different times from different locations can benefit from tools such as mail management systems.

		Time	
		Same	Different
Space	Same	Meeting facilitation	Work shifts Team rooms
	Different	Tele/video/desktop conferencing Interactive multicast seminars	E-mail Computer bulletin boards Collaborative writing Workflow

Figure 2.1: Groupware classification according to the two dimensions of a collaborative process: time and space [50]

With the growth of interest in the design and development of groupware technology, the technological landscape fitting Figure 2.1 became much richer. However, several considerations can be made at this point:

- Some cells benefit from more computer support than others.
- Most real work does not necessarily fall into one category or another.
- Technology designed to support activities in one cell can prove to be unproductive for activities in other cells.

—

Collaborative activities can be performed in two modes:

1. *Synchronous*: All collaborators are working in the same time on the same shared document. Information is being exchanged at the same time, such as in a face-to-face meeting. Examples of tools which support synchronous collaboration are instant messaging, and electronic whiteboarding. Synchronous collaboration is more interactive, resembling a face-to-face conversation between collaborators. The immediacy character of synchronous collaboration enables a more natural way of communicating and sharing information. One drawback of synchronous collaboration is that the tools to support it are not that well spread and used. Moreover, their diversity and capabilities are still poor. Another drawback brought by synchronous collaboration is its lack of flexibility, oftentimes being difficult for all the parties involved to be ready and willing to collaborate at the same given time.

2. *Asynchronous*: The collaborators do not have to be simultaneously active on a document. This implies that different people might receive the information at different times. Examples of tools which support asynchronous collaboration include e-mail, newsgroups, and discussion boards. Asynchronous collaboration enables flexibility, allowing each collaborator to digest the information received and not feel the pressure of an immediate response. Moreover, asynchronous collaborative tools are spread largely, e-mail management systems being an example in this direction. Asynchronous collaboration implies less immediate interaction,

sometimes a longer period of time being needed for a collaborator to receive a response or feedback on a shared document. The direct consequence of this fact may be that the information can be out of date by the time someone views it.

Sometimes, there is no clear delimitation between the two modes and/or there could happen that the two modes intertwine. However, specific contexts may ask for specific modes of collaboration. For example, an immediate decision making process might require that all the parties involved collaborate towards the decision making in the same time, whereas less urgent actions might allow the collaboration to be conducted in an asynchronous manner.

2.1.2 Design Challenges and Concerns

Back in 1989, Liam Bannon underlined three major issues for CSCW. First, "any cooperative effort involves a number of secondary tasks of mediating and controlling the association of individuals" [16]. This leads to the need of exploring ways for articulating cooperative work as a first core issue. A second core issue in CSCW as noted by Bannon is supporting a shared information space since cooperative work "may require the interaction of people with multiple goals of different scope and nature as well as different heuristics, conceptual frameworks". Lastly, fitting technology into the workplace is an acute issue for CSCW, a better understanding and control of "the interaction between technique and work organization" being needed. In 1994, Jonathan Grudin points out rightfully that "because individuals interact with a groupware application, it has all the interface design challenges of single-user applications, supplemented by a host of new challenges arising from its indirect involvement in group processes" [50]. As a result, he identifies eight challenges for developers in the design of computer systems to support cooperative work:

1. "A groupware application never provides the same benefits to every group member". Therefore, it is suggested to design processes that create benefits for all group members, along with the underlying technology.

2. "Most groupware is only useful if a high percentage of the group members use it". Therefore, a solution could be reducing the work required of all and building

in incentives of use.

3. ”Groupware can lead to activity that violates social taboos, threatens existing political structures or demotivates users crucial to its success”. Therefore, it is highly recommended to work with representative users whenever possible.

4. ”Groupware may not accommodate the wide range of exception handling and improvisation that characterizes much group activity”. Therefore, the author suggests providing flexibility by using tailorable systems.

5. ”Groupware features will fare better if integrated with features that support individual activity”. Therefore, it is suggested to design for accessibility.

6. ”Task analysis, design, and evaluation are much more difficult for multi-user applications.”

7. ”Good intuition for multiuser applications is unlikely to be found anywhere in a product development environment”

8. ”Groupware must be introduced very carefully, leaving little to chance.”

A review of the work done nowadays in identifying major challenges and concerns in CSCW adds to the list above several other finer-grained issues, identified through further observations and experience. Described below, there is a brief overview of the results of the review, including those challenges also covered by or closely related to the work described by this thesis.

2.1.2.1 Technology Supporting CSCW

Technical support of collaborative processes is a complex challenge because ”the interacting people usually have differing backgrounds, ways of thinking and of self-expression, and their collaboration is only weakly structured” [54]. Based on interviews with 13 professionals in CSCW, Herrmann et al. developed a set of design heuristics by categorizing and condensing the interviewees’ statements:

- Supporting the large picture, providing means for visualizing rich material. A typical example which is compliant to this design heuristic is an interactive large screen on which several "contributions can simultaneously be made visible and readable".
- Malleability of shared material and stimulation of variations.
- Support of convergence within evolutionary documentation. Examples of functions which support this heuristic are rating, voting, mind maps.
- Smooth transitions between the two different modes of collaboration (synchronous and asynchronous).
- Integration of communication with work on shared material. Counter examples are wikis which are examples of insufficient integration of communication.
- Support of role dynamics and varying mode of collaboration.

Co-located collaborative work on tabletop displays is the focus of [95] where a set of system guidelines for the design of applications targeting tabletop devices is identified using data sources such as: a) literature on existing digital tabletop systems, b) HCI and CSCW literature on design requirements, implications and guidelines for co-located CSCW systems, c) CSCW literature involving observational studies of co-located collaboration involving traditional media, d) literature from the social sciences on interpersonal communication and tabletop collaboration, and e) experience. The guidelines described included:

- Support interpersonal interaction.
- Support fluid transitions between activities.
- Support transitions between personal and group work.
- Support transitions between tabletop collaboration and external work.
- Support the use of physical objects.

- Provide shared access to physical and digital objects
- Consideration for the appropriate arrangements of users.
- Support simultaneous user actions.

In [36], the authors hypothesize that "shared displays within a group setting would impact the behaviour of a group". For evaluating the hypothesis, the authors compared the participation of the members of a group in conversation with and without the presence of a shared display used for continuously revealing how much each person had participated. The results indicated that the shared display of social information impacted the amount participants spoke as follows: a) "over-participators responded to the display by significantly decreasing the amount they spoke", b) "under-participators responded by not increasing the amount they spoke", and c) "subjects over-estimated their level of participation in a conversation in that under-participators rated the display as a less accurate reflection of their behaviour than over-participators did".

Shared mobile devices are a particular category of devices supporting collaboration, and they do share unique design concerns. Understanding of how the interfaces provided on a shared device influence mobile collaborative navigation activity is depicted in [82]. Two interfaces providing the same route descriptions, but differing in the way this information is combined were evaluated. Results reported that "text facilitated collaborating at a distance, while graphical depictions encouraged gathering around the device".

Single-display groupware systems enable their users to concurrently share and interact with a computer. On one hand, this leads to an increase in productivity and facilitates group communication [86]. On the other hand, it brings new challenges related to the ways users input information and the ways information is outputted for all users' access. In [12], a discussion on multiple mouse text-entry techniques is provided together with the description and the evaluation of 13 such techniques designed to be used in educational contexts. The authors considered several design factors for such techniques to be used by multiple students simultaneously in classroom settings, and these factors included: cost, screen footprint, scalability, leveraging multiple users, learning rate, speed, accuracy, and user preferences. The 13 techniques were classified as on-screen keyboard techniques, multi-letter keyboard techniques, scrolling techniques, collaborative techniques, and advanced techniques.

In [86], the authors introduce a system that allows four users to each receive sound from a private audio channel while using a shared tabletop device. Further on, an evaluation of the impact such a system has on the group productivity, communication and work strategies was performed. Results showed that: a) group members participated more equitably in the task when using the system as opposed to when not using private audio channels, b) they spoke to each other more frequently, and c) they managed the available time more effectively than when individual audio channels were not available.

2.1.2.2 Coordination and Conflict

Coordination is one major concern in CSCW. Even if it highly depends on the context of the cooperative work, several general considerations and results have been described in the literature.

As pointed out in [60], in cooperative work ”the informal norms once known by everyone need to be codified into explicit rules and enforced”. However, ”as communities grow, they encounter common coordination problems”. Three major coordination mechanisms are described:

1. Peer-to-peer communication is considered to be the most basic form of coordination within a group and it is especially needed in conditions of uncertainty.

2. Group structure consists of role differentiation, division of labor, and formal and informal management.

3. Shared mental models are defined as beliefs held in common among a set of collaborators ”about what should be done and who should be doing it.” [60]. Shared mental models can also be imposed through standards, guidelines, or policies.

The impact of each of these mechanisms was studied on a pool of wiki production groups, results showing that: a) "Policies and procedures were associated with less conflict when smaller numbers of editors worked on content, but were associated with increased conflict when many editors were involved", and b) "Communication and concentration were more associated with reduction of conflict as more editors were involved".

In [87], a coordination framework for group tabletop usage is presented. Three major types of conflicts are identified in collaboratively working around a tabletop: a) global conflicts which involve changes that affect the application as a whole, b) whole-element conflicts which involve access to a single object, and c) sub-element conflicts which occur when several users are editing the same item simultaneously and issue conflicting changes. To overcome these types of conflicts, three types of strategies are proposed:

1. Proactive initiative strategies which allow an element's owner or the initiator of a global change to control the outcome of the conflict.
2. Reactive initiative strategies which produce an effect based only on the actions of the other users.
3. Mixed-initiative strategies.

In addition to that, the authors describe several coordination policies for each type of conflict. As global conflicts coordination policies, they propose "No selections, no touches, no holding documents", "Voting", "Rank", "Privileged objects", "Anytime", whereas whole-element conflict coordination policies they propose "Public", "Private", "Duplicate" (view a read-only copy of the original document), "Personalized views", "Stalemate" (if a user tries to take a document someone else is working on, the document becomes temporarily inactive to both users), "Tear" (breaks the document into two pieces), "Rank" (a higher-ranking user can always take documents from a lower-ranking user), "Speed, force" (physical measurements determine the "winner" of a specific document), "Sharing", "Explicit" (a document's owner retains explicit control over which other users can access that document), "Dialog" (a popup dialog box appears for the owner of a document through which s/he gives access to another user to that document).

An investigation of the conditions ”under which conflict arises, and the effectiveness of coordination mechanisms in managing conflict different scales” in wiki production groups is described in [60]. Results suggest that the density of the information space is a key determinant of conflict, as opposed to the absolute number of contributors. The authors consider as measure of conflict the number of reverts in the wiki per month.

2.1.2.3 Communication

With respect to communication, there are several factors that may affect the dynamics of a conversation as well as participants' perceptions of one another [14]. Two of them are discussed in [14]: timing and responsiveness (i.e. the time until a person responds to communication) in semi-synchronous communication, such as instant messaging. The questions addressed by the authors are: a) How does the user's ongoing activity affect his or her responsiveness to incoming communication?, b) Will responsiveness vary based on who sent the message?, c) Will people respond at different speeds during different parts of the day?, d) How does the content of the communication affect the user's responsiveness to it?, e) Will responsiveness, when the communication is already ongoing, differ from responsiveness to attempts to initiate new communication? The results obtained lead to the identification of several design opportunities. First, the fact that ”open requests for help may be neglected in favor of new ones merely for being in the background”, asks for the need to ensure the prominence of ongoing help requests when new ones arrive. Moreover, in order to improve users' ability to discuss external resources (such as URLs), a seamless link between the conversation, the URL reference and the browser is needed. Lastly, distinguishing between task and sub-task boundaries across applications would allow avoiding the disruption of high-level tasks.

2.1.2.4 Notifications and Awareness

Questions such as ”how do notifications support users' need for information awareness?” and ”what effect does this need have on their focus on tasks that are interrupted by the arrival of notifications?" are addressed in [58] where a comparative study on users' reactions to notifications for email communications is described. The goal of the authors was to "understand how interruptions caused by notifications influence users' focus on ongoing tasks and to contrast this with task focus if notifications are turned off". Results showed that users react to only about a quarter of all mail notifications and that user focus on the primary task is largely unaffected if the notifications are disabled.

The effects of intelligent notification management systems on users and their tasks are described in [58] where the OASIS system is presented. OASIS was designed on two major considerations: 1) probe strategies for detecting interruptible moments leveraging the structure of users' tasks, and 2) build statistical models to detect interruptible moments during tasks. Therefore, the system: a) allows notifications to be deferred until breakpoints are reached during interactive tasks, b) permits the users to identify breakpoints and correlate with those detected, and c) generates notifications that are relevant to the user's ongoing activity.

Multi-synchronous authoring tools allow simultaneous work in isolation of members of the group and the subsequent integration of their contributions. In such a context, private workspaces are essential for collaboration since they give co-authors the possibility to "carry out polishing and revision of their contributions before communicating or including them in shared documents" [57]. An awareness mechanism to support users in filtering the amount of information about their changes to be delivered to their collaborators is described in [57]. The mechanism is based on user defined privacy levels.

2.1.2.5 Interruptions

In collaborative work, interruptions are frequent. Their effects at different moments within task execution are studied in [5]. The assumption the authors are verifying is that "poorly timed interruptions can adversely affect task performance and emotional state". Different computer-mediated strategies for notification on interruption are highlighted, including visual strategies, multimodal presentations, or appropriate timing. As result of their analysis, the authors identify several implications for the design of an attention manager system. First, several alternate modalities for interruption should be considered. Moreover, in order to develop an effective attention manager, one should either supply it with the task models or it must learn the task models over time.

A slightly different perspective on dealing with interruptions is presented in [51], where the authors underline the importance of an efficient mechanism for preventing and/or recovering from disconnections especially in synchronous collaborative processes. This is due to the fact that "a central assumption of synchronous groupware is that the members of the group are temporally present – that is, they are actively observing changes to the shared workspace and noticing new updates as they arrive". The authors describe an application-level framework for dealing with disconnection in synchronous groupware together with a classification of possible disconnections. They identify three types of disconnections: 1) delay-based interruptions, 2) network outages, and 3) explicit departures. The framework considers all types of disconnections through identification of absence, adaptive behavior during disconnection, and re-establishment of the synchronous interaction.

2.1.2.6 The Social Side of CSCW

The social aspects of CSCW and their influence on the effectiveness of collaborative processes are pointed out from early on in the CSCW literature [16]. The interest in these aspects has not faded overtime, some examples being described below.

Kalnikaite et al. investigate the utility of tagging to construct social summaries of complex multimedia material through a study during which students were allowed to apply time-indexed tags (such as handwritten annotations or photos) to different parts of the lecture in real-time [59]. Also, the students were presented with information about which tags are most frequently accessed by others. The authors addressed three questions: 1) Do users make greater use of systems offering social feedback?, 2) Which types of social tags (notes or photos) are most useful for retrieving lecture materials?, and 3) What are the benefits of using this type of system? Results showed that the benefits of the social feedback were clear, 49% of all tag accesses were for popular tags. Moreover, students preferred notes over photos, the authors explaining: "because they provided a finer granularity of access (recall that there were almost three times as many word tags as pictures)".

Gaming is yet another domain where one would expect to find benefits of social features in collaboration. As result, recommendations for the design and support of social activities within multiplayer games are described in [38], based on an analysis of the player-to-player interactions in a multiplayer game.

2.1.2.7 Annotations

Annotations are association of specially marked knowledge elements with specific documents or elements within them [33]. They support several cognitive functions as follows [24]:

- Remembering - by highlighting the most important parts of a document.
- Thinking - by supporting each user in adding his/her own ideas and feedback to the document.
- Clarifying – by rephrasing the content of a document on the words of its reader.

Pioneering initiatives in allowing annotation of web content (such as Annotea) force the user to disrupt his/her navigation activity in order to start an annotation application. To answer this drawback, Bottoni et al. focus on smoothly integrating tools which support annotation in existing browsers. Their results are presented in [24], where MADCOW - a digital annotation system "organized in a client-server architecture, where the client is a plug-in for a standard web browser and the servers are repositories of annotations to which different clients can login" - is presented.

An annotation model specifically targeting collaborative writing is described in [110]). The model provides the following advantages:

- Supports in-situ communication and decision making via threaded annotations.
- Improves cross-role awareness via "smooth transitions or interactions between work done by people of different roles, such as reviewer and author".
- Provides a rational version control mechanism by "capturing integrated version history and annotations leading to revisions".
- Improves the shared workspace and the group awareness.

Within the model, each annotation is defined in greater detail by the following 11 elements:

1. Context: the description of the object the annotation is attached to.
2. Message body: the actual text of the annotation.
3. Annotation creator: information about the author of the annotation.
4. Annotation recipient: information about those users to whom the annotation specifically addresses.
5. Annotation time: the creation time of the annotation.
6. Response deadline: the time by which the annotation should be addressed.
7. Responses: information useful for forming threaded discussions.
8. Status: indicates whether there are responses to an annotation, whether annotations are incorporated in new versions, etc.
9. Category: captures the characteristics of the problems brought up in an annotation.
10. Rating: label the annotation with either "positive feedback" or "negative feedback".
11. Urgency: sets up the priority for annotations.

A comprehensive group-memory aid for software developers which acts by pushing content and annotations form a link target (in this case, online documentation of commonly used APIs) to its source (in this case, commonly used IDEs such as Eclipse) is described in [33]. The authors' motivation found its roots in the belief that "readers of a document in a linked information space can benefit not only from the knowledge and experiences of its original authors but also from those of subsequent readers and contributors".

In a similar context, TagSEA [3] allows tagging locations of interest in Eclipse. The framework is developed using both social bookmarking techniques and geographic navigation methods and it supports tagging locations of interest by means of keywords, adding data and author metadata, filtering tags, navigating to places once tagged, creating both shared tags and private tags. The tags apply to any of the following elements: source code, breakpoints, tasks and resources.

2.1.2.8 Roles in CSCW

An in-depth discussion of the concepts of 'role' and 'role mechanism' is presented in [53] and the motivation for this is introduced as two-fold. On one hand, "social systems create and change themselves by reciprocal expectations towards behaviour" and on the other hand, "communities can only develop and build up, if the participants accept the conditions under which they can interact and the scope of options which determine their activities" [53]. The synthesised description of the concept of 'role' would include four characteristics: position, functions/tasks, behaviour/expectations, and social interaction. Seven role-mechanisms - "interaction patterns for role-taking and role-making" - are defined in [53]: role-assignment, role-taking, to allow someone's role-taking, role-change, role-making, role-definition, inter-role-conflict. The discussion takes a sociological stand and it is meant as a "foundation for building and supporting socio-technical community systems"[53].

Lukosch and Schummer define the concept of role as combining prototypical behavior, rights, capabilities, and obligations [68]. The authors address the problem of users structuring their interaction in the group and propose a set of design patterns for describing what the owner of the role is supposed to do. Moreover, a discussion on the tools to be used in order to reach the role's intended goal is provided. Some examples of applications which support the assignment of roles are:

- Blackboard is an software application used for educational purposes where each user has a specific role defined by a set of permission levels:
  - Course builder: has access to all the features except "Assessments" and "Course tools".
  - Grader: has access to the grade book and is able to edit "Assessments".
  - Instructor: has access to all of the course functions.
  - Student: has read-only access to all of the course content.
  - Teaching assistant: shares the same level of access as the instructor.
  - Administrator: assigns the roles to the users.
- SourceForge is an open source software development web site in which each project's members can have a specific role as developer, administrator, translator, and so on.

—

The subsection brings together some of the challenges to be faced when designing software applications for synchronous collaboration. Surely, an exhaustive list of such challenges could very well stand for the topic of a brand new thesis and it is for that reason that my interest mainly relates to those challenges covered by the work I present further on in the thesis. All of the above address the context in which groups of people work together - either remotely or co-located - in the same time, sharing the same resource. In that respect, technology is one key aspect since moving from individual to collaborative use of tools asks for new means of capturing and displaying feedback, innovative displays and communication channels. Socio-related factors such as coordination, conflict, communication, awareness, roles go hand in hand with any collaborative process. If designing for individual use had very little to do with these issues, shifting to collaborative use brings to light additional challenges and concerns such as: how to make sure that collaborators coordinate their effort?, how to deal with conflicts among them?, how to support the representation of role-structures similar to those existing in real-life?, how to make sure that all those collaborating are aware of each others contributions?, how to support common understanding among those working together?.

2.2 Design Patterns

2.2.1 History and Evolution

Patterns and pattern languages were introduced by the architect Christopher Alexander in the seventies [7] [9] [8] as tools for capturing and making available and communicable knowledge related to urban spaces. Alexander conceived urban spaces as artifacts, where "people enjoy living in" and which "have a certain, timeless 'Quality without a Name' that cannot be reduced to a single dimension" [9]. These environments must provide affordances which support "the patterns of events that frequently happen there" [8]. Patterns of events that frequently happen in a space and the relationships among them are not created by the architects themselves, but emerge from the interaction among their inhabitants and the space itself. Urban spaces are not designed in insulation but as a system: they refer to each other, smaller spaces being defined in the context of larger ones. Design becomes a process in which space is differentiated to create a complex solution. To design urban spaces of the desired quality, Alexander saw the necessity for architects to explain their views to their clients, to discuss within the architects community about the reached solutions of design problems and to have a repository of the knowledge created through the design activities performed by the community. This repository evolves in time, recording new solutions to the (possibly new) problems arising in design activities.

Alexander conceived documents to be used by architects: 1) as knowledge repositories about the solutions of often recurring urban design problems, 2) as means of communication of the solutions among the architect communities and, 3) as communication means between architects and their clients in the design of urban spaces. He called these documents ”patterns”. Alexander established a precise structure and layout of a design pattern: each pattern has a name, a descriptive entry, and some cross-references to other design patterns which support and contextualize the solution described. Each structural section is characterized by a specific graphical layout. The uniform format of presentation improves the usability of design patterns, because readers can develop reading patterns [56], adaptable to the different uses of the documents required by their activities. Design patterns are not independent but they constitute a network of inter-related documents, the ”pattern language”. A pattern language is a network of patterns which organizes good design practices within a domain. Alexander did not propose any formal definition of design patterns and design pattern languages but only informal guidelines for their development. His proposal is limited to the use of paper based documents organized in a hypertext fashion.

Alexander's approach had a wide impact in several domains, including Computer Science and HCI. Software engineering (SE) applied design patterns for expressing Object-Oriented software design experience. Software engineering patterns address mainly professional programmers and computer scientists and are not intended to a general audience. Moreover, the collection of design patterns and the relationships among them are not complete enough to form a pattern language in the Alexandrian sense [46]. The HCI community was attracted by the Alexander's approach in two directions. First, HCI designers adopted the metaphor which maps an interactive system to a space which offers affordances for humans to develop their activities and to face the variances which can affect them. Reenskaug coined the term habitable spaces to define these virtual spaces [81]. Secondly, many HCI designers adopted the design pattern and the design pattern language approach to document and describe ”the reasons for design decisions and the experience from past projects, to create a corporate memory of design knowledge” [22].

Borchers evolves Alexander's notions of design pattern and design pattern language while recognizing the HCI design as a complex process. He adheres to the view that the design of complex processes requires more knowledge than any single person can posses [42]. Therefore, he proposes a user centered approach to HCI design in which stakeholders from the application domain, HCI and SE collaborate to the design. This leads to the definition of three pattern languages: one for describing the problems met by stakeholders in the targeted domain, one for describing the problems in the HCI domain and one for describing the problems met by stakeholders in SE. These languages facilitate the communication among all the stakeholders involved in the design. Moreover, Borchers recognizes the importance of formalism as a support for reasoning and creation of software tools. Therefore, he introduced a graph based definition for design pattern languages, which he uses for developing a new way of visualization and access to design patterns and patterns language. However, the definition underlies the design pattern construction. To be usable by their users, patterns are presented as multimedia information, including images, sketches and graphical schema and not as formulae. Design patterns become Web documents (nodes of the graphs) and the pattern language is presented to users as a browsable map representing the graph and deploying the hyper-textual structure of the language in a way understandable by all stakeholders in the design team. To reach this result and exploit the affordances of the Web 2.0, the Pattern Language Mark-up Language (PLML) was defined for allowing the translation of the definition of a design pattern into an XML Web document and presents a sample authoring and browsing tool to work with pattern languages.

2.2.2 Design Pattern Collections

Several collections of design patterns have been published, the diversity of areas they are targeting growing in the past few years. Initially addressing urban and architecture design, the concept of design pattern has inspired and was adopted in design areas such as: software engineering, interaction and web design, collaborative applications, social interfaces, usability, ubiquitous computing, interactive exhibits, and accessibility. This section aims at providing various examples of such collections for: a) a comparative analysis of the concept across different domains and design areas and b) a walk-through of the evolution of the concept from its initial definition in architectural design till its current use in different other domains.

2.2.2.1 Urban and architectural design

"Towns and buildings will not be able to become alive, unless they are made by all the people in society, and unless these people share a common pattern language, within which to make these buildings, and unless this common pattern language is alive itself" [8]. This is how Christopher Alexander motivates his initiative of writing a collection of 253 inter-related design patterns (Appendix 7.3) to be used by architects in urban and architectural design. These patterns are ordered, beginning with more general ones – such as *INDEPENDENT REGIONS* or *MAGIC OF THE CITY* – and ending with more specific ones – such as *SITTING WALL* or *POOLS OF LIGHTS*. They "create a coherent picture of an entire region, with the power to generate such regions in a million forms, with infinite variety in all the details".

The first 94 patterns define a town or a community, the patterns relating to:

- The regional policies which protect the land and mark the limits of the city.
- The city policies which encourage the formation of the major structures which define a city.
- Self-governing communities which "exist as physically identifiable places".
- Connecting communities for encouraging the growth of the city network.
- Neighbourhood policies which help control "the character of the local environment".
- Boundaries which encourage the formation of local centers.
- Work communities.
- Public open land.

The second part of the collection comprises design patterns addressing groups of buildings and individual building in three dimensions. Some of the aspects discussed by these patterns are:

- The height and number of the building in a group of buildings.

- The entrances to the site.
- Main parking areas.
- Lines of movement through the complex.
- The position of individual buildings on the site, within the complex of buildings.
- The shaping of the volume of the buildings and of the space between the buildings, at the same time.
- The most important areas or rooms within a building.

Lastly, the last part of the collection includes design patterns targeting the design of specific details of construction such as:

- The exact positions for openings (doors and windows), and the frame of these openings.
- Surfaces, indoors and outdoors details.
- Ornament the building with lights and colors.

2.2.2.2 Software engineering

Software engineering adopted the concept of design pattern around 1987, but it was only on 1995 when Gamma et al. published a first collection of design patterns for object-oriented software design [46]. The collection contains three types of patterns:

1. Creational patterns address class instantiation and are further divided into class-creation patterns and object-creation patterns. The creational patterns are:
   - *Abstract* factory groups object factories which have a common theme.
   - *Builder* separates construction and representation of complex objects.
   - *Factory method* creates objects without specifying their class.
   - *Prototype* allows cloning existing objects.

- *Singleton* restricts the object creation for a class to only one instance.

2. Structural patterns use inheritance to compose objects and define ways to compose objects for obtaining new functionality. These patterns are:

   - *Adapter* wraps its interface around that of an existing class for allowing incompatible interfaces to work together.
   - *Bridge* decouples the abstraction from the implementation.
   - *Composite* composes several objects so that they can be used as one.
   - *Decorator* dynamically adds behavior in an existing method of an object.
   - *Facade* provides a simplified interface to a complex part of code.
   - *Flyweight* reduces the cost of manipulating a large number of similar objects.
   - *Proxy* provides a placeholder for another object to control the access to it.

3. Behavioral patterns address communication issues between objects and include:

   - *Chain of responsibility* delegates command to a chain of processing objects.
   - *Command* creates objects which encapsulate actions and parameters.
   - *Interpreter* implements a specialized language.
   - *Iterator* accesses the elements of a complex object sequentially without exposing the object's representation.
   - *Mediator* couples classes by knowing the details of their methods.
   - *Memento* provides undo capabilities for complex objects.
   - *Observer* allows a number of observer objects to be notified of an event.
   - *State* allows an object to modify its behaviour when its state changes.
   - *Strategy* allows the selection of one algorithm out of a family of algorithms to be selected for execution at runtime.
   - *Template* method defines the skeleton of an algorithm to be defined as an abstract class, which has further subclasses providing concrete behaviour.
   - *Visitor* separates an algorithm from an object structure.

2.2.2.3 Graphical User Interface (GUI) design

"Design engaging and usable interfaces with more confidence and less guesswork" is what Jennifer Tidwell is supporting in [103]. The book describes a collection of design patterns for interaction design, comprising desktop applications, web sites, web applications, and mobile devices. The collection of patterns comprises eleven categories of patterns as follows:

1. Patterns of users behaviour: *Safe Exploration*; *Instant Gratification*; *Satisficing*; *Changes in Midstream*; *Deferred Choices*; *Incremental Construction*; *Habituation*; *Microbreaks*; *Spatial Memory*; *Prospective Memory, Streamlined Repetition*; *Keyboard Only*; *Other People's Advice*; *Personal Recommendations.*

2. Patterns on information architecture and application structure: *Feature, Search, and Browse*; *News Stream*; *Picture Manager*; *Dashboard*; *Canvas Plus Palette*; *Wizard*; *Settings Editor*; *Alternative Views*; *Many Workspaces*; *Multi-Level Help.*

3. Patterns on navigation, signposts, and wayfinding: *Clear Entry Points*; *Menu Page*; *Pyramid*; *Modal Panel*; *Deep-linked State*; *Escape Hatch*; *Fat Menus*; *Sitemap Footer*; *Sign-in Tools*; *Sequence Map*; *Breadcrumbs*; *Annotated Scrollbar*; *Animated Transition.*

4. Patterns for organizing the page (i.e. the layout of the page elements): *Visual Framework*; *Center Stage*; *Grid of Equals*; *Titled Sections*; *Module Tabs*; *Accordion*; *Collapsible Panels*; *Movable Panels*; *Right/Left Alignment*; *Diagonal Balance*; *Responsive Disclosure*; *Responsive Enabling*; *Liquid Layout.*

5. Patterns for images, messages, search results: *Two-Panel Selector*; *One-Window Drilldown*; *List Inlay*; *Thumbnail Grid*; *Carousel*; *Row Striping*; *Pagination*; *Jump to Item*; *Alphabet Scroller*; *Cascading Lists*; *Tree Table*; *New-Item Row.*

6. Patterns on actions and commands: *Button Groups*; *Hover Tools*; *Action Panel*; *Prominent "Done" Button*; *Smart Menu Items*; *Preview*; *Progress Indicator*; *Cancelability*; *Multi-Level Undo*; *Command History*; *Macros.*

7. Patterns for showing complex data through trees, graphs and other information graphics: *Overview Plus Detail*; *Datatips*; *Data Spotlight*; *Dynamic Queries*;

Data Brushing; *Local Zooming*; *Sortable Table*; *Radial Table*; *Multi-Y Graph*; *Small Multiples*; *Treemap.*

8. Patterns for getting input from users: *Forgiving Format*; *Structured Format*; *Fill-in-the-Blanks*; *Input Hints*; *Input Prompt*; *Password Strength Meter*; *Autocompletion*; *Dropdown Chooser*; *List Builder*; *Good Defaults*; *Same-Page Error Messages.*

9. Patterns on using social media: *Editorial Mix*; *Personal Voices*; *Repost and Comment*; *Conversation Starters*; *Inverted Nano-Pyramid*; *Timing Strategy*; *Specialized Streams*; *Social Links*; *Sharing Widget*; *News Box*; *Content Leaderboard*; *Recent Chatter.*

10. Patterns for mobile devices: *Vertical Stack*; *Filmstrip*; *Touch Tools*; *Bottom Navigation*; *Thumbnail-and-Text List*; *Infinite List*; *Generous Borders*; *Text Clear Button*; *Loading Indicators*; *Richly Connected Apps*; *Streamlined Branding.*

11. Patterns on visual style and aesthetics: *Deep Background*; *Few Hues, Many Values*; *Corner Treatments*; *Borders that Echo Fonts*; *Hairlines*; *Contrasting Font Weights*; *Skins and Themes.*

Van Welie published online a separate collection of design patterns for interaction design and specifically for web design[1], the author classifying the patterns into:

- Patterns addressing the user's needs which include navigation, searching, dealing with data, making choices, giving input, personalizing, and other basic interactions.
- Patterns addressing the needs of the application are further classified into patterns for drawing attention, providing feedback, and simplifying interaction.
- Patterns related to the context of the design classify sites, experiences, and pages.

[1] A complete list of these patterns is available online at: http://www.welie.com/patterns/index.php.

A collection of design patterns for developing Silverlight applications is available at [2]. Several patterns included in this collection are similar to patterns in other collections. Some such examples are: *Undo*, *Visual Framework*, *Wizard*.

2.2.2.4 Collaborative applications

Design patterns in the field of computer-mediated interaction have been written by Stephan Lukosch and Till Schummer who, in their book ""Patterns for Computer-mediated Interaction" [94], described patterns for community building support, group interaction support, and base technology.

The patterns addressing community building target issues such as:

- How to arrive in a community: *QUICK REGISTRATION*, *LOGIN*, *WELCOME AREA*, *MENTOR*, *VIRTUAL ME*, *USER GALLERY*, and *BUDDY LIST*.
- How to deal with quality: *QUALITY INSPECTION*, *LETTER OF RECOMMENDATION*, *BIRDS OF A FEATHER*, *EXPERT FINDER*, *HALL OF FAME*, and *REWARD*.
- How to protect users: *RECIPROCITY*, *MASQUERADE*, *AVAILABILITY STATUS*, *ATTENTION SCREEN*, and *QUICK GOODBYE*.

The patterns targeting group interaction support address issues such as:

- How to modify shared material together: *GROUP*, *SHARED FILE REPOSITORY*, *SHARED BROWSING*, *VOTE*, *APPLICATION SHARING*, *SHARED EDITING*, and *FLOOR CONTROL*.
- How to create places for collaboration: *ROOM*, *ACTIVE MAP*, *INTERACTION DIRECTORY*, *BELL*, *INVITATION*, and *BLIND DATE*.
- How to support textual communication: *EMBEDDED CHAT*, *FORUM*, *THREADED DISCUSSIONS*, *FLAG*, *SHARED ANNOTATION*, *FEEDBACK LOOP*, *DIGITAL EMOTIONS*, and *FAQ*.

- How to provide synchronous group awareness: *USER LIST*, *SPONTANEOUS COLLABORATION*, *ACTIVE NEIGHBOR*, *INTERACTIVE USER INFO*, *REMOTE FIELD OF VISION*, *REMOTE SELECTION*, *REMOTE CURSOR*, *TELEPOINTER*, and *ACTIVITY INDICATOR*.
- How to maintain asynchronous group awareness: *ACTIVITY LOG*, *TIMELINE*, *PERIODIC REPORT*, *CHANGE INDICATOR*, *ALIVENESS INDICATOR*, and *AWAY MESSAGE*.

The group of patterns related to base technology address issues such as:

- How to handle sessions: *COLLABORATIVE SESSION*, *PERSISTENT SESSION*, *STATE TRANSFER*, and *REPLAY*.
- How systems manage common data: *CENTRALIZED OBJECTS*, *REMOTE SUBSCRIPTION*, *REPLICATED OBJECTS*, *NOMADIC OBJECTS*, *MEDIATED UPDATES*, *DECENTRALIZED UPDATES*, and *DISTRIBUTED COMMAND*.
- How systems ensure data consistency: *PESSIMISTIC LOCKING*, *OPTIMISTIC CONCURRENCY CONTROL*, *CONFLICT DETECTION*, *OPERATIONAL TRANSFORMATION*, *LOVELY BAGS*, and *IMMUTABLE VERSIONS*.

In addition to that, the same authors have proposed patterns for assigning roles in collaborative processes [68], and shared object management [67].

A collection of design patterns for cross-culture collaboration is identified by Nicole Schadewitz in [91] and comprises the following 11 patterns: *GRAND OPENING*, *COMMUNITY WATCH*, *INTERNATIONAL HOME*, *STRUCTURED CHAT*, *SUMMING UP*, *MOOD OF THE MOMENT*, *ANNOTATED DESIGN GALLERY*, *WHO WHEN WHAT*, *LOCAL VARIATIONS*, *GLOBAL RESOLUTION*, and *GRAND FINALE*.

2.2.2.5 Social interfaces

The rapid growth of interest in social networks led to the need of a knowledge base comprising best practices in designing social interfaces. Such a knowledge base is the collection of design patterns presented in [30]. The issues addressed by the collection are:

- Broadcasting and publishing.
- Identity and reputation.
- Sharing and collaboration.
- Community management.
- Communication and feedback.
- Social media.

The patterns related to collaboration target collaborative editing in wiki-based application, crowdsourcing, voting, and project management.

Design patterns for e-Government applications are presented in [79]. The authors initiate by listing down a set of user interface design golden rules, following to describe three types of design patterns:

- Basic components are recommendations such as: *CONDITIONAL ACTIVATION OF FIELDS*, *DOWNLINK LINK*, *MANDATORY FIELDS*, *NON TEXTUAL OBJECTS*, *PRE-FORMATTED FORM FIELDS*, and *TYPOGRAPHY*.
- Page level patterns include: *ACKNOWLEDGMENT OF RECEIPT*, *ADVANCEMENT BOX*, *CLEAR ENTRY POINTS*, *FILTER A LIST*, *OVERVIEW*, and *WIZARD STEP*.
- Screen flow level patterns comprise: *CONSULT AND MODIFY DATA*, *FILE MANAGEMENT*, *HUB AND SPOKE*, *INTEGRATION IN A PORTAL*, *MULTI-STEP WIZARD*, and *ROLE MANAGEMENT*.

A different stance to social aspects is described in Douglas Schuler's book, "Liberating voices: A Pattern Language for Communication Revolution" [93]. The collection of 136 patterns presented by the book address different behavioural patterns and relate to organizing principles, enabling systems, policy, collaboration, community and organizational building, self representation, tactics. The full collection is available online and it is open to further modifications from interested communities (http://www.publicsphereproject.org/patterns/pattern-table-of-contents.php).

2.2.2.6 Usability

In his book "A Pattern Language for Web Usability" [47], Ian Graham describes 79 design patterns organized as a collection and addressing usability of web applications. These patterns are categorized into:

- Patterns for getting started on a site design.
- Patterns for improving usability.
- Patterns for adding detail to a design to enhance usability even further.
- Patterns for dealing with workflow and security issues.

The full collection of patterns can be found in Appendix 7.3.

Patterns for usability have been also explored in [70], where four types of patterns have been identified:

- Patterns of tasks give developers "an insight into the functionality which should be provided and how it will be used".
- Patterns of users "can be used to explore the forces involved in the context of a particular kind of user accessing the system and to specify the user-interface accordingly".
- Patterns of user-interface elements "help detailed designers and programmers understand where it is appropriate to use a certain user-interface element, possibly as a replacement for traditional documentation on toolkit usage".
- Patterns of entire systems capture issues involved in their development.

2.2.2.7 Ubiquitous computing

The applications for ubiquitous computing are "systems that make use of sensors, computing devices in a variety of form factors, and wireless networking to assist us in all kinds of tasks" [29]. This is the target of the collection of pre-patterns described in [29]. These patterns describe:

- Ubiquitous computing genres: they provide a taxonomy of the emerging ubiquitous applications. Some examples of such patterns are *SMART HOMES*, and *PERSONAL MEMORY AIDS*.

- Physical-virtual spaces: they help designers understand the ways to improve the users' navigation through such spaces. Some examples of such patterns are *ACTIVE MAP*, *FIND A PLACE*, and *NOTIFIER*.

- Developing successful privacy: they address policy, systems, and interaction issues in designing privacy-sensitive systems. As examples of such patterns consider *PRIVACY MIRRORS*, and *INVISIBLE MODE*.

- Designing fluid interactions: they describe the ways to design applications which involve a large number of sensors and devices, allowing the users to feel in control. Some examples of such patterns are *SCALE OF INTERACTION*, *ACTIVE TEACHING*, and *KEEPING USERS IN CONTROL*.

2.2.2.8 Interactive exhibits

One of the promoters of the concept of design patterns in interaction design is Jan Borchers who describes a collection of patterns for the design of interactive exhibits in [22]. The collection includes the following 17 patterns: *ATTRACT-ENGAGE-DELIVER*, *ATTRACTION SPACE*, *COOPERATIVE EXPERIENCE*, *EASY HANDOVER*, *SIMPLE IMPRESSION*, *INCREMENTAL REVEALING*, *FLAT AND NARROW TREE*, *AUGMENTED REALITY*, *CLOSED LOOP*, *LANGUAGE INDEPENDENCE*, *DOMAIN-APPROPRIATE DEVICES*, *INNOVATIVE APPEARANCE*, *IMMERSIVE DISPLAY*, *INVISIBLE HARDWARE*, *DYNAMIC DESCRIPTOR*, *IN-*

FORMATION JUST IN TIME, and *ONE INPUT DEVICE*.

2.2.2.9 Accessibility

Design patterns aiming to provide "a structured model of the design knowledge in the accessibility domain" are described in [45]. The patterns are classified according to two dimensions:

1. Functionality, leading to patterns on presentation (such as *SEPARATE CONTENT FROM PRESENTATION*), navigation (such as *NAVIGATING HIERARCHICAL INFORMATION*), and interaction (such as *LIVE REGION*).
2. Level of abstraction, describing high level (*MULTI-FORMAL PRESENTATION*, *EASY NAVIGATION*, *INTERACTION AT USER'S PACE*), mid level (*MEANINGFUL STRUCTURE*, *WHERE IS THE FOCUS?*, *ALERT STOP AND RESUME*) and low level patterns (*COLOR AND CONTRAST*, *SELECTION ELEMENT*, *EXCLUSIVE CHOICE*).

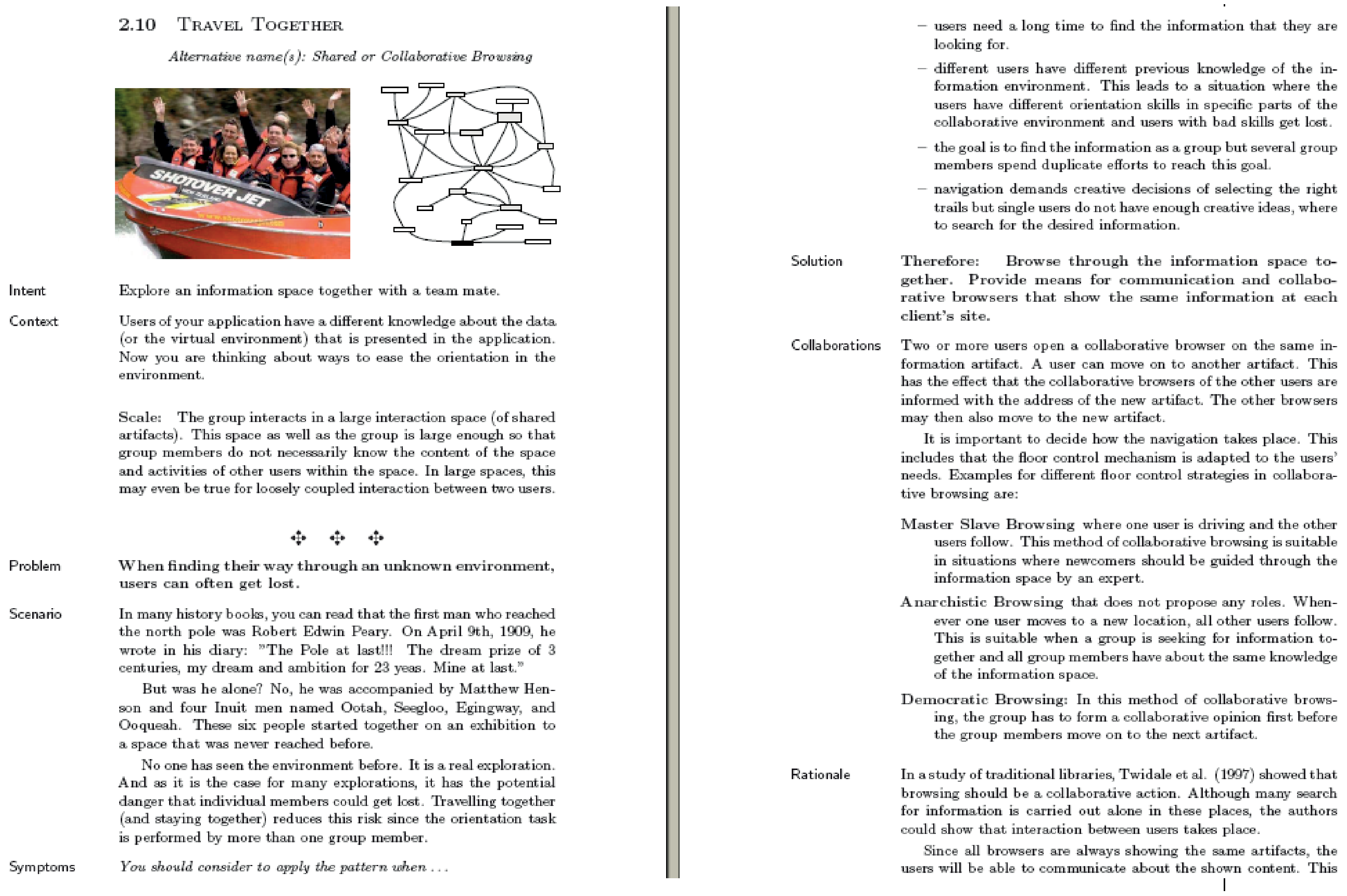
—

As illustrated above, the coverage of the existing collections of patterns is quite broad. Different areas and domains are subject to being documented through patterns. However, none of the collections specifically address synchronous collaboration and this thesis aims at filling in that particular gap. Moreover, at a closer look, these collections use different template definition for the patterns they comprise. This led to an investigation of the template definitions used across collections for a better understanding of what these elements refer to and in which way they help.

2.2.3 Template Definitions of Patterns

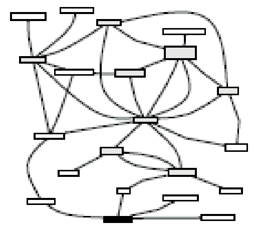
The set of all the defining elements used to describe the patterns in a specific collection is referred to as the template of definition of the patterns. Several different such templates have been proposed. They generally include the name of the design pattern, the description of the problem it addresses together with the forces that influence this problem, some examples of situations in which this problem can be met and a possible solution to tackle the problem [35]. The results of a survey of the used template definitions for design patterns are presented in [64], the authors identifying 4 major parts in the definition of patterns: the head, the body, additional information, and references. Each of these parts is further defined by a set of elements[1]. Figures 2.2 and 2.3 present an example of a design pattern, comprising a subset of the defining elements listed below.

[1] Each author decides on the elements to use for defining a design pattern, not all the elements being mandatory. The final goal is for the pattern to be understandable to its readers.



2.10 Travel Together

Alternative name(s): Shared or Collaborative Browsing



Intent — Explore an information space together with a team mate.

Context — Users of your application have a different knowledge about the data (or the virtual environment) that is presented in the application. Now you are thinking about ways to ease the orientation in the environment.

Scale: The group interacts in a large interaction space (of shared artifacts). This space as well as the group is large enough so that group members do not necessarily know the content of the space and activities of other users within the space. In large spaces, this may even be true for loosely coupled interaction between two users.

✣ ✣ ✣

Problem — **When finding their way through an unknown environment, users can often get lost.**

Scenario — In many history books, you can read that the first man who reached the north pole was Robert Edwin Peary. On April 9th, 1909, he wrote in his diary: "The Pole at last!!! The dream prize of 3 centuries, my dream and ambition for 23 yeas. Mine at last."

But was he alone? No, he was accompanied by Matthew Henson and four Inuit men named Ootah, Seegloo, Egingway, and Ooqueah. These six people started together on an exhibition to a space that was never reached before.

No one has seen the environment before. It is a real exploration. And as it is the case for many explorations, it has the potential danger that individual members could get lost. Travelling together (and staying together) reduces this risk since the orientation task is performed by more than one group member.

Symptoms — *You should consider to apply the pattern when . . .*

- users need a long time to find the information that they are looking for.
- different users have different previous knowledge of the information environment. This leads to a situation where the users have different orientation skills in specific parts of the collaborative environment and users with bad skills get lost.
- the goal is to find the information as a group but several group members spend duplicate efforts to reach this goal.
- navigation demands creative decisions of selecting the right trails but single users do not have enough creative ideas, where to search for the desired information.

Solution — **Therefore: Browse through the information space together. Provide means for communication and collaborative browsers that show the same information at each client's site.**

Collaborations — Two or more users open a collaborative browser on the same information artifact. A user can move on to another artifact. This has the effect that the collaborative browsers of the other users are informed with the address of the new artifact. The other browsers may then also move to the new artifact.

It is important to decide how the navigation takes place. This includes that the floor control mechanism is adapted to the users' needs. Examples for different floor control strategies in collaborative browsing are:

Master Slave Browsing where one user is driving and the other users follow. This method of collaborative browsing is suitable in situations where newcomers should be guided through the information space by an expert.

Anarchistic Browsing that does not propose any roles. Whenever one user moves to a new location, all other users follow. This is suitable when a group is seeking for information together and all group members have about the same knowledge of the information space.

Democratic Browsing: In this method of collaborative browsing, the group has to form a collaborative opinion first before the group members move on to the next artifact.

Rationale — In a study of traditional libraries, Twidale et al. (1997) showed that browsing should be a collaborative action. Although many search for information is carried out alone in these places, the authors could show that interaction between users takes place.

Since all browsers are always showing the same artifacts, the users will be able to communicate about the shown content. This

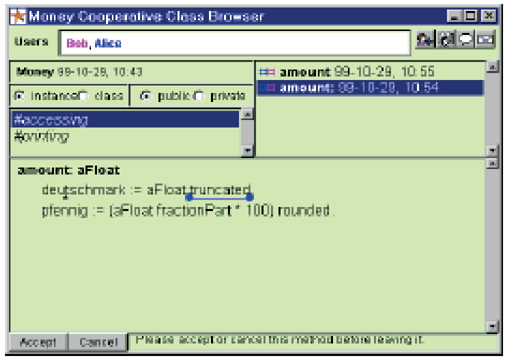
Figure 2.2: *Travel together* design pattern from [94] - part 1

helps them to better understand the artifacts. When a user navigates to another artifact, all other users follow, which ensures that the group remains focussed on the same artifact.

By discussing the route, the team will choose the most appropriate path. And since many users travelled together, there is a better history of the steps taken.

Danger Spots Since all browsers are coupled, the group will always travel at the same pace. This can be a problem if the comprehension speed for the browsed artifacts differs significantly. In this case, fast users can feel obstructed by the slower users.

Known Uses TUKAN (Schümmer 2001) is a collaborative software development environment. It informs the programmers of the presence of other programmers to support dynamic group formation. After users have met, they can navigate through the software system using a tightly coupled browser (cf. figure 21). The selection of the class, the protocol, and the method are shared by all users, thus they always see the same method. Within the method, the users can read independently (the scroll position of the text pane is not coupled).



```
amount: aFloat
    deutschmark := aFloat truncated.
    pfennig := (aFloat fractionPart * 100) rounded.
```

Figure 21: The Collaborative Class Browser in TUKAN.

During the collaborative exploration of the source code, the developers can communicate about the shown code fragments using an integrated chat tool.

CobWeb (Stotts et al. 1998) allows a group of users to browse web pages together. It uses two frames in a standard web browser: the content is shown in one frame of the browser while the other frame is used for controlling the browsing.

This frame also includes means for requesting the floor (requesting to driv).

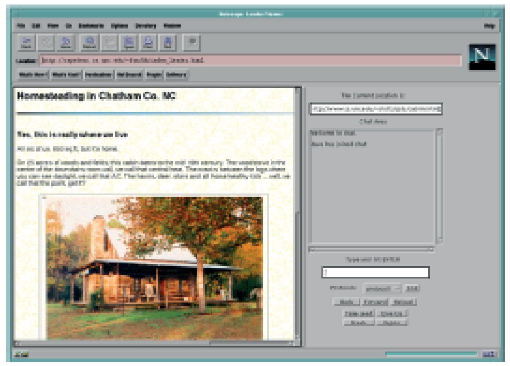

Homesteading in Chatham Co. NC
Yes, this is really where we live

Figure 22: Collaborative Web Browsing in CobWeb (Stotts et al. 1998).

One special feature of CobWeb is that it allows the system developer to model interaction models for collaborative browsing by means of a petri net. This approach provides flexible means for designing all interaction processes discussed above.

Comparable to the CobWeb system are GroupScape (Graham 1997) and CoWeb (Jacobs et al. 1996). Both systems also support collaboratibe synchronous browsing of the web.

✣ ✣ ✣

Related Patterns APPLICATION SHARING$_{\rightarrow 2}$ is the general concept of using an application together with another user and seeing exactly the same application state at each point in time.

SWARM AND COLLECT$_{\rightarrow 2}$ is comparable to collaborative browsing with respect to the fact that a group wants to explore a collaborative information space together. The difference is that each group member explores parts of the space individually and then shares the relevant results in the group.

TELL ME A STORY$_{\rightarrow 2}$ uses records of other users' browsing activities to simulate a collaborative browsing activity. The main differences are that only one user defines the path and that this user is not co-present.

Figure 2.3: *Travel together* design pattern from [94] - part 2

HEAD

The head in the definition of a design pattern provides an overview of the problem addressed by the pattern together with a set of metadata such as the name of the author, and the creation date of the pattern. A full list of elements which can be found in the head of a design pattern definition is provided below.

1. **Pattern number.** A pattern number is associated to a design pattern in order to uniquely identify it in a collection of patterns. This element supports the indexing and the referencing of patterns. Even if the pattern number is usually a number, it might happen that this identifier contains an additional code used for categorizing the pattern.

2. **Pattern name.** The pattern name suggests the main idea of the pattern. It is used mainly to create a vocabulary within the community using a collection of patterns (hence, it should be easy to remember) and it should provide a significant hint to the content described by the pattern.

3. **Alternative pattern name.** An alternative name for a design pattern can be provided for making it more understandable or for following a set of constraints.

4. **Rating/ranking.** Being used within communities of designers, design patterns are subject to the designers' feedback. Hence, they can be rated or ranked so that pattern users are supported in making informed decisions on what patterns to consider. However, such an element becomes relevant in time and only when intensively used within a specific community.

5. **Image.** An image suggesting the main idea of the pattern can be associated to it. Usually, images are suggestive and do not require a lot of time to grasp. It is for this reason that images associated to patterns prove to be helpful when collections of patterns are browsed by pattern users.

6. **Author name.** The author of the pattern is the person who wrote the pattern and made it available.

7. **Pattern classification.** Usually, pattern collections are classified according to some criteria (more on this in Section 2.2.2). The definition of a pattern should also provide information on the category the pattern belongs to.

8. **Creation date.** The creation date points to the date when the pattern was created.

9. **Last revision date.** Patterns are dynamic entities, subject to modification. Hence, each pattern may at any time be revised either by its author or by other users. In these situations, storing the date of the last revision proves to be helpful.

10. **Level.** The level of a pattern refers to the level of abstraction the pattern addresses. High-level patterns describe problems at an abstract level and point to medium or low level patterns which provide descriptions to more concrete problems. The level is a means of triggering a hierarchy within a collection of patterns.

BODY

The body definition of a design pattern details the information included in the head, adding to that other elements, relevant to the understandability of the problem addressed by the pattern.

11. **Context.** The context of a design pattern is regarded as a precondition which decides the applicability of a specific pattern. As opposed to guidelines, design patterns support their users in understanding the overall landscape to which a pattern adheres by specifying the context element in its definition.

12. **Problem description.** The description of the problem addressed by the pattern is one of the core defining elements of a pattern. This description presents the major point of the problem, being further supported by other defining elements comprised in the body.

13. **Forces.** The forces illustrate the tradeoffs to be considered when applying the pattern. Some implications the pattern has might be conflicting and might generate or ask for specific consideration. This is the kind of information provided by the forces.

14. **Solution.** The solution element describes proven solutions to the problem described by the patterns. As opposed to guidelines, the pattern solution is not a step-by-step list of instructions to be followed to get to one specific solution. The pattern solution can be used "a million times over, without ever doing it the same way twice" [9].

15. **Rationale.** The rationale element provides a proof of concept, showing why the pattern works and how the forces described are balanced by the application of the solution.

16. **Diagram.** The diagram is a graphical representation of the solution proposed by the pattern. Depending on the level of the pattern and on the domain the problem belongs to, this diagram can be a sketch, or even a more complex representation such as an UML diagram.

17. **Resulting context (consequences).** The application of the pattern triggers a set of consequences described by the resulting context element. These consequences might ask for the application of other patterns or might require further design considerations and decisions.

18. **Examples.** The examples are illustrating applications of the solution proposed by the pattern. They are usually accompanied by illustrative screenshots and links to the working instances of the solution described by the pattern.

19. **Known uses.** Quite similar to the examples, the known uses point to applications of the design pattern in already implemented applications. They are meant to support the pattern users in understanding the pattern by examples.

20. **Counter examples.** As opposed to the examples, the counter examples show poor design in the context of the design pattern described. The counter examples are meant to show the consequences of either not applying the pattern in a given context or applying it wrongly.

ADDITIONAL INFORMATION

Considering the problem and the solution as being the core elements in the definition of a design pattern, their understandability is enhanced by any other additional information. Some of these additional elements have been described above and are comprised in the body of the pattern. Others are included in the additional information part.

21. **Related literature.** References to literature which discusses the mechanisms and concepts described by the pattern may be added to the 'Related literature' element.

22. **References to implementations.** Patterns may also be described by a set of references to their implementations.

23. **Code example.** Design patterns for software engineering use the code element as a core defining element. However, since HCI design patterns address a different audience and have a slightly different goal, the code element is seldom met in the definition of HCI design patterns.

REFERENCES

The 'References' part contains information on design patterns related to the one described.

24. **Related patterns.** Patterns related to the one described (either from the same collection or from different collections) are described as references. As result, the pattern users are redirected to other patterns, describing related and possibly influencing problems to the one described by the current pattern.

Table 2.1 identifies the pattern defining elements used in the definitions of the design pattern collections described in Section 2.2.3.

Within a collection of design patterns, authors use one single template for defining all the patterns in the collection. This is meant to support the pattern users in developing reading patterns and easily finding relevant information within the collection.

Defining Element	(Alexander, 1977) [9]	(Gamma, 1995) [46]	(Borchers, 2001) [22]	(Schuler, 2002) [93]	(Graham, 2003) [47]	(van Welie, 2003) [105]	(Tidwell, 2005) [103]	(Schummer, 2007) [94]	(Crumlish, 2009) [30]
Number				X	X			X	
Name	X	X	X	X	X	X	X	X	X
AltName		X			X			X	
Rank	X		X		X				
Image	X		X		X	X	X	X	
Author				X	X				X
Class		X							
Creation									
Revision									
Level									
Context	X		X	X	X		X	X	
Problem	X	X	X	X	X	X	X	X	X
Forces	X	X	X	X	X				
Solution	X	X	X	X	X	X	X	X	X
Rationale		X		X		X	X	X	X
Diagram	X		X	X	X				
Conseq.		X			X		X		
Examples		X				X	X	X	
Uses		X					X	X	X
CounterEx									
Literature						X			
Implemen.						X			
Code		X							
Related	X	X	X	X		X	X	X	X

Table 2.1: Templates of definition for various design pattern collections

2.2.4 Design Patterns versus Guidelines

There have been several parallels made between design patterns and guidelines, since guidelines have been used to capture and describe ergonomic knowledge, as well. Following some of these parallels' reasoning, guidelines prove to be very versatile, being employed during several phases of the development process such as design and evaluation [79]. Due to this fact, guidelines are often ambiguous and can only be applied correctly by experts. Moreover, experts may experience difficulties in selecting and applying guidelines since they are sometimes conflicting with one another and there is a wide gap between the recommendation of the guideline and its application.

As opposed to guidelines, design patterns "focus on the context of a very specific problem at a time and provide a solution that not only includes the ergonomic knowledge but also guides the designers to apply it in a practical way" [79]. This overcomes the versatility and ambiguity of guidelines. Since design patterns are easier to apply than guidelines, the number of patterns required for covering a specific design area is higher. Moreover, due to their semi-formal template of definition, design patterns allow designers to form reading patterns, guiding them in browsing entire collections [56].

According to Borchers, "style guidelines, guidelines, and standards are the forms of expressing HCI design experience that are close to HCI design patterns"[22]. However, one of the major differences he identifies between design patterns and guidelines resides in the fact that design patterns are primarily constructive in that they suggest how a problem could be solved, whereas guidelines are mainly descriptive, "merely stating desirable general features of a good finished interactive system". In the author's words:

"Patterns can improve these forms through their structured inclusion of existing examples and an insightful explanation not only of the solution, but also of the problem context in which this solution can be used, and the structured way in which individual patterns are integrated into the hierarchical network of a pattern language, similar to the distinction between general, category-specific, and product-specific guidelines."

2.2.5 Design Pattern Mining Methods

Often, design patterns are identified by experts in the field of application of the patterns [37] and the process these experts follow is seldom described. However, literature documents two types of methods for design pattern mining: a). inductive methods which start by observing the specifics of a context and move towards generalizations, and b). deductive methods which start from generalizations and move towards identifying the specifics of a context [15], [111].

Inductive methods include:

- ad-hoc *discussions* among experts in fields such as computer gaming where elements of game design or narratives are discussed.
- *structural analysis* and *play testing* in fields such as game design.
- *multi-disciplinary descriptions* and validations through which collaborative learning patterns are identified by collaborative learning practitioners, and validated by pedagogy experts [111].
- *systematic pattern development cycles* targeting the design of e-learning systems. Such a cycle is proposed in [83], where a 4-phase pattern development process based on the reverse engineering of e-learning systems which embed good designs is described.

Deductive methods include:

- drawing *mind maps* for:
  - free exploration of a central topic,
  - exploration of the five questions of a scenario: who, why, how, when, where,
  - map of things learned through the years [15]
- describing *metaphors* such that "the attributes of one general environment and the functions of these attributes are translated towards functions of another type of environment" [15].

- experts' *experience* in confronting themselves with a recurrent problem.

- discussions held during the *PLoP workshops* where patterns are discussed in small groups, feedback being provided in the form of a face-to-face peer review.

- *shepherding* [111] which is essentially a reviewing process organized within PLoP conferences. Shepherds are individuals, with experience in pattern writing, assigned to an author's paper with the expressed interest in helping the author improve the pattern.

- *open calls* for patterns [93] soliciting patterns in the field of civic participation. From the 170 such patterns that have been received from contributors through the web, a committee of 34 members selected 64 for presentation and further refinement.

- using preexisting organizational *ontologies* as formal specifications of shared semantics [90].

Each of these methods is suited for specific domains of use; nonetheless, often times both inductive and deductive methods are used during pattern mining processes [90].

2.2.6 Documented Uses of Design Patterns

Teaching. Findings of teaching and evaluating Computer Science courses that dealt with HCI design patterns are summarized in [23]. They suggest that HCI design patterns are useful tools to teach HCI design principles as well as to support students in formulating their own design experiences. Two major ways of using design patterns for teaching have been identified. Firstly, taught as a method, they should be considered "as a segment of a larger advanced class in HCI design methodologies". Secondly, design patterns serve well as a tool and format for teaching HCI design principles.

A specific added value for the use of design patterns is identified in [63] as being the support in acquiring design skills and domain knowledge. The three case studies described in [63] support the following propositions: a) "Novices will faster gain understanding in problem solving and design skills, when they learn to design with the design patterns approach first, before they learn to understand entire systems", b) "Experienced designers will not experience a learning effect from the use of design patterns, but might find them useful in other ways", and c) "Training novices with the use of design patterns will increase the quality of the schemas they build to represent a system".

Design. An initial and emerging collection of 45 pre-patterns for ubiquitous computing have been described and evaluated in [29]. Sixteen (16) pairs of designers used the pre-patterns for designing location-enhanced applications. The pre-patterns were emailed to the participants prior to the 90-minute design sessions. The design sessions were directly observed, results showing evidence of the following: a) pre-patterns helped novice designers, b) pre-patterns helped designers with the unfamiliar domain, and c) pre-patterns helped designers avoid some design problems.

An extension to this work is presented in [89] where the same collection of pre-patterns was evaluated by 22 pairs of professional designers. Half of the pairs performed a 120-minute design creation task without any external aid, while the other half was given access to the pre-pattern collection via a browser for performing the same task. Results show that the pre-patterns were mostly effective in supporting designers generate design ideas, and allowing them to go back to the pre-patterns to get clarifications on open issues.

A slightly different approach to evaluating design patterns is described in [32] where the contribution that a collection of interrelated patterns could make to the user participation in the design of interactive systems is investigated. A designer-facilitator worked with a user to develop the design of either a travel website or a web-based learning resource using a collection of design patterns addressing web design. Direct observations revealed that users made extensive reference to the patterns' illustrations, often without referring to the text of the patterns. Also, it proved to be important that only a small number of patterns were presented to the users at the same time. The patterns were also used as a checklist to ensure that all the issues have been discussed.

If design patterns emphasize capturing a problem-solution pair in a certain context, design claims focus on describing the positive and the negative implications of a design decision [4]. The case study described in [4] evaluates the benefits of structuring design advice in a pattern or a claim form and, instead of declaring the pattern or the claim structure as a clear winner, proposes a hybrid structure for sharing design advice. The paper also underlines an under-appreciated contribution of design patterns which is their ability to offer "a way to capture and share successful design trade-offs in context" [4].

2.2.7 From Patterns to Pattern Languages

A collection of design patterns together with all the relationships identified between the patterns was defined by Alexander as a pattern language [9]. Each of the patterns described by Alexander is connected to other patterns, so that the entire collection is grasped as a whole, as a language, "within which you can create an infinite variety of combinations". Adopting the idea of a pattern language, Jan Borchers notes: "pattern languages essentially aim to provide laymen with a vocabulary to express their ideas and designs and to discuss them with professionals". He identifies inter-related pattern languages able to define three levels of an application design: software engineering, interaction design, and the target domain of the application. In [22], the three languages address interactive exhibits and are for software engineering, interaction design of interactive exhibits, and music composition.

The advantages of pattern languages over less structured collections of patterns are many. First, patterns are written with the purpose of capturing knowledge. Grasping and understanding this knowledge asks for unfolding the individual concepts and the relationships between them. Hence, traversing the collection supported by an underlying logic, i.e. the relationships between the patterns, helps creating representations of the area addressed by the collection. Secondly, patterns are written with the purpose of describing recurring problems and proven solutions to tackle them. Hence, their goal is to support designers in reusing proven solutions and not having to reinvent the wheel. Using a solution, however, does not always limit to that and often asks for considering related problems as well. Support in finding such related problems is brought by the relationships between the patterns documenting them. Lastly, patterns are written with the purpose of sharing knowledge helping to build an integrated repository of knowledge. Adding to such a repository asks for connecting newly acquired knowledge to the existing base, such connections needing to be explicit. Pattern languages are by definition such structures, allowing the identification of relationships between different bodies of knowledge.

Common to all the authors proposing pattern languages is the lack of a method for identifying the relationships between the patterns. The only contribution in this direction is the description of some possible relationships between two patterns, X and Y [105]:

- X uses Y in its solution
- X is similar to Y
- X can be combined to Y
- X is a sub-pattern of Y
- X is related to Y

Similarly, in [114], a classification of possible relationships between software design patterns is provided and this includes the following types: X uses Y in its solution, X is similar to Y, X can be combined to Y. Based in these relationships, the authors relate the software patterns in [46], forming a pattern language. Moreover, these patterns are separated in 3 different layers: design patterns specific to an application domain, design patterns for typical software patterns, and basic design patterns and techniques. However, no method for actually identifying such relationships between patterns has been described.

The goal in [25] is to provide a "language independent formalization of the notion of pattern, so that it allows its application to different modelling languages and tools, as well as generic methods to enable pattern discovery, instantiation, composition, and conflict analysis". The goal is met by proposing a mechanism for suggesting model transformations in a way that models become consistent with the patterns. Even if the authors specifically target software engineering patterns, the approach has proven powerful enough to formalize patterns from other domains, including interaction patterns.

All in all, there is very little documented support in generating a pattern language out of an existing collection. Surely, classifying possible relationships between patterns is a valuable contribution, but then this area is lacking a method to support relating (apparently) independent patterns. Interesting enough, most of the pattern authors present their patterns in the form of pattern languages, being aware of all the advantages of a language over a collection. However, they are solely supported by experience, valuable but not easy to replicate and evaluate as a method. Deriving from this, one of the gaps this work is aiming to fill in is precisely describing a method which would support the semi-automation of a pattern language generation.

—

Even if the concept of design pattern has been around since the '70s, there is still work to be done in supporting research in this field by means of methods and tools. A broad spectrum of domains are subject to being documented by patterns and most of the collections of patterns addressing these domains are further structured as pattern languages. However, the methodological landscape for identifying and relating design patterns is quite scarce, most of the authors basing their judgement mainly on experience. Moreover, even if communities are formed around the existing collections of patterns - their role being to manage and use such repositories - their is still little understanding of the ways these collections are used. Having as starting point the findings described above, I aim to bring further methodological support in the areas of identifying and relating patterns. Moreover, inspired by the studies described above for understanding the use of design patterns in teaching and design, I look into the use of design patterns specifically in collaborative interaction design processes.

2.3 Creativity in Software Design

Software is rarely associated with creativity. However, software design, as any type of design, is a highly creative endeavor [71]. It implies the key steps of any design process - problem finding and problem solving, understanding and defining problems, balancing forces and coming up with creative solutions. For that reason, it could highly benefit from the techniques and concepts revolving around creativity.

2.3.1 The Creative Process - History and Evolution

The history of the creative process definition dates way back to the late 19th century, beginning of the 20th century. A first attempt of a creative process model definition was proposed in 1926 by Graham Wallas [108] who identifies four phases in a creative process: i). *preparation* as the phase in which the problem to be solved is clarified and understood, ii). *incubation* when one no longer consciously considers the problem, iii). *illumination* as the phase in which the creative insight occurs, and iv). *verification*, the last phase during which it is verified that the creative insight is indeed a solution for the problem to be solved. Osborn [77] refines this definition and proposes a two-phase model for defining a creative process. The first phase consists of the *idea generation*, being followed by a second phase called *idea evaluation*. Idea generation is developed in two sub-phases. First, the problem at hand is being clearly defined and understood (*fact finding* phase), and secondly, new ideas are being produced through the combination of already existing ones (*idea finding* phase). Idea evaluation implies assessing the ideas generated in order to identify creative solutions. Another similar model was proposed by Amabile [10] which defines the creative process based on four phases: i). *problem presentation*, ii). *preparation*, iii). *response generation*, and iv). *response verification*. The preparation phase implies building up knowledge about the problem and researching what a potential solution may necessitate. During the last two phases, possible solutions are generated for the problem and then verified in the given context.

The particularity of all these models of creativity is that they look at one individual or a small group of individuals solving a problem, creating the image of designers working in isolation. This, however, is rarely the case today due to the exponential complexity of design problems and to the expanding scale of design projects. The concept of community became central to design and this brought new challenges related to the way knowledge is being created, shared and reused within a specific community. In [98], Schneiderman proposes the genex framework for defining creative collaborative processes, identifying four phases of a creative process: i). *collect*, ii). *relate*, iii). *create*, and iv). *donate*. During the first phase, previous similar work is studied and learned from. The relate phase brings people together in order for them to consult with each other and exchange knowledge on the problem at hand. The create phase assumes the exploration, composition and evaluation of possible solutions to the problem at hand. The last phase allows the dissemination of the solutions reached, contributing in this way to the global knowledge within a community.

The dynamic and complex nature of collaborative design problems certainly involves more comprehensive knowledge than a single person can possess [42]. The knowledge associated with design problems is tacitly distributed among the various individuals or communities involved in the collaborative design process [43]. Thus, involving all the stakeholders in problem solving collaboration is necessary and leads to social creativity. Social creativity arises not in one individual's mind, but from "the interaction between a person's thoughts and a socio-cultural context" [42]. It exploits the creativity of groups of minds while their interaction through tools and artifacts. There are three major principles of social creativity: i). *knowledge creation*, ii). *knowledge integration*, and iii). *knowledge dissemination*. Knowledge creation is defined as the process of externalization of an individual's tacit knowledge. As a further step, knowledge integration assumes merging the information that is collaboratively constructed into the problem-solving context. Lastly, knowledge dissemination not only asks for efficient techniques for knowledge sharing, but also requires the possibility of providing the 'right' information at the 'right' time and in the 'right' way [42]. One of the challenges that arise in collaborative design processes comes from the need to support social creativity through tools and techniques appropriate to each stakeholder involved in the process. Supporting the interaction, communication and common reasoning [39] on the problem at hand is, nowadays, a must in the context of collaborative design.

2.3.2 Creative Techniques in Software Design

There are several creative techniques documented in the literature; of interest to this work are some of those mostly fitting the context of software design. The subsection will introduce scenario-based design, sketches, and mockups.

2.3.2.1 Scenario-based Design

”Software design is fundamentally about envisioning and facilitating new ways of doing things and new things to do.” [28]. The complexity of software design problems exceeds one’s individual ability to tackle them and asks for the collaboration of stakeholders with different expertise and backgrounds. Graphic designers, software engineers, programmers, human-computer interaction specialists, and marketing people come together and collaborate in the design of software applications [43]. In addition to that, it is often the case that software addresses the problems of clients (often, the users) which may not be (and are not willing to be) experts in software design. However, they need to communicate with the software designers which are not experts in the domain of the clients/users. Therefore, one of the challenges in designing software applications is finding ways to communicate design ideas and interaction representations. One way to do that is by describing as in a story the actors (the potential users of the application) and the activities (the actions the application supports) they would perform when engaged in the interaction with the application under design. Such descriptions are called scenarios. They are stories which describe people in action, their goals, and motivation, the concrete descriptions of activities that engage the user when performing a specific task [104]. Scenarios prove to be powerful design tools in that:

- they are easily understandable by all those involved in the design process – software designers, programmers, clients/users,
- they allow reasoning about situations of use even before those situations actually exist [28],
- they support software designers in understanding the requirements expressed by the client/user, providing ”a description sufficiently detailed so that design implications can be inferred and transformed into actual models” [104],

- they constitute a bridge between the specialized language of the client/user and the specialized language of the designer,
- they provide insight into the ways to tackle usability aspects, since they provide "snapshots" of the application in use.

Defining a scenario should answer a set of pre-requisites. First, any scenario should have a narrative character; it should sound like a story. This not only supports the communication and common understanding among the stakeholders involved in the design process, but it also supports the dialog of the designers with the clients/users and vice versa. In addition to that, any scenario should answer a set of questions, such as: who are the users?, what are their goals?, what is their motivation to use the software application?, how could they use the application, and when and where can the application be used?. scenario-based design – i.e. the technique of using scenarios during design processes – has been applied in various stages of the software development cycle, such as requirements analysis, user-designer communication, design rationale, documentation and training, evaluation, abstraction and team building [104].

2.3.2.2 Sketches

"Designers explore new ideas and concepts at various stages of their design cycle using different material artifacts" [107]. Examples of such artifacts are sketches. They are tools for capturing preliminary observations and ideas [106]. In addition to supporting externalization processes, sketches have been documented to enhance social interaction [96], coordination among collaborating designers [18], and introspection [92]. Design processes often times require moving from abstract, non-structured ideas to concrete, well-defined concepts. This move may not always be linear, requiring the designers to "initiate, explore, combine, transform, refine, and reject different ideas" [71]. In that respect, sketches are tools which support designers in the exploration of the design space, in the same time allowing them not to commit to design ideas too early in the process. sketches can be of several types; they can be concrete or abstract, representational or symbolic, improvisational or rehearsed [106]. Moreover, they are used in various domains and contexts, of interest to this work being their use in software design processes, and more particular during GUI design.

The initial phases of GUI design are governed by uncertainty. Objects belonging to the GUI may have "uncertain types, sizes, shapes, and positions" [65]. It is because of this uncertainty that designers do not feel the burden of deciding on details such as colors, alignments, and fonts which should be decided on during later phases of the process. Sketching GUIs gives the process certain fluidity, since sketches are rough representations and may be modified at any time based on the continuous exploration of the design space of the application. Sketches have been documented to encourage and enhance creativity in GUI design in that "when the designers generated a new idea in a freehand sketch, they quickly followed it with several variations." [65]. The more ideas get generated during design processes, "the greater the probability of achieving an efficient solution" [109]. Hence, the more creative the design process is, "the greater the probability of designing useful and usable software applications and computer systems" [109].

2.3.2.3 Mockups

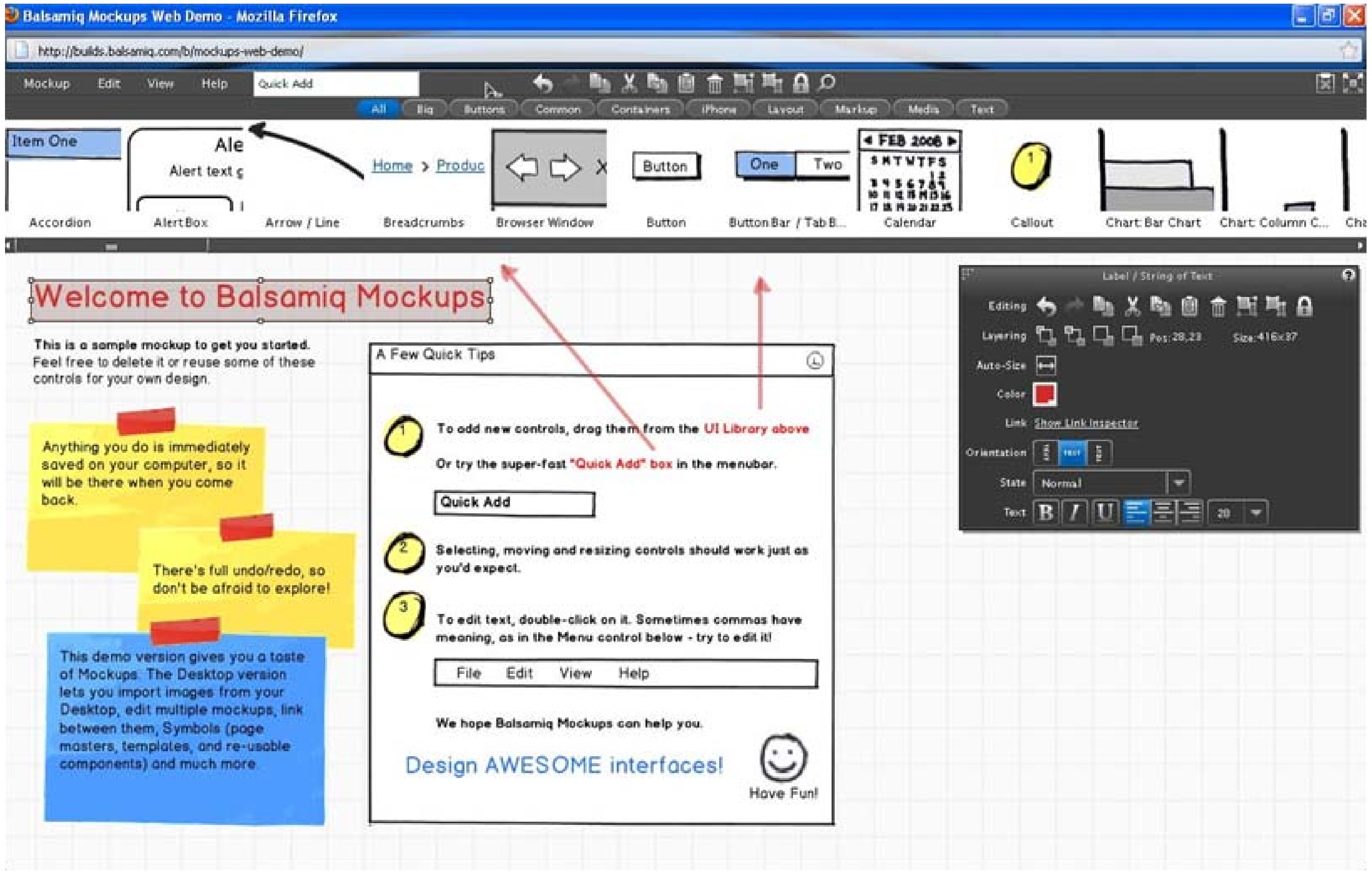
Mockups are 'very early prototypes' made of cardboard or otherwise low-fidelity materials. They resemble the final product, but only at a surface level, having little of the eventual functionality [48]. Mockups help designers negotiate on UI design related aspects and may be easily created by anyone involved in the process. Several tools exist for creating mockups, the most popular being Balsamiq (Figure 2.4) and Mockingbird (Figure 2.5).

Balsamiq promotes the idea according to which mockups reproduce the experience of sketching interfaces on a whiteboard, only that on a computer. That implies mockups are easier to share, modify, and store. Moreover, Balsamiq supports the following:

- Quick idea generation.
- Sooner review and iteration processes.
- Improved communication between designers, developers and product managers.

- Real-time iterations involving all parties.

Mockingbird is a fully web-based mockup editor which allows dragging and dropping UI elements to the page. These elements may be rearranged and resized. Several mockups can be linked together obtaining a flow of the application as a whole. Each mockup can be shared through a link which, if made available to other users, allows its real-time edit. Final mockups can be exported to commonly used formats for visualization purposes such as pdf or png.



Welcome to Balsamiq Mockups

This is a sample mockup to get you started. Feel free to delete it or reuse some of these controls for your own design.

Anything you do is immediately saved on your computer, so it will be there when you come back.

There's full undo/redo, so don't be afraid to explore!

This demo version gives you a taste of Mockups. The Desktop version lets you import images from your Desktop, edit multiple mockups, link between them, Symbols (page masters, templates, and re-usable components) and much more.

A Few Quick Tips

1. To add new controls, drag them from the UI Library above

Or try the super-fast "Quick Add" box in the menubar.

2. Selecting, moving and resizing controls should work just as you'd expect.

3. To edit text, double-click on it. Sometimes commas have meaning, as in the Menu control below - try to edit it!

File Edit View Help

We hope Balsamiq Mockups can help you.

Design AWESOME interfaces!

Have Fun!

Figure 2.4: Snapshot of Balsamiq - a tool supporting mockup creation

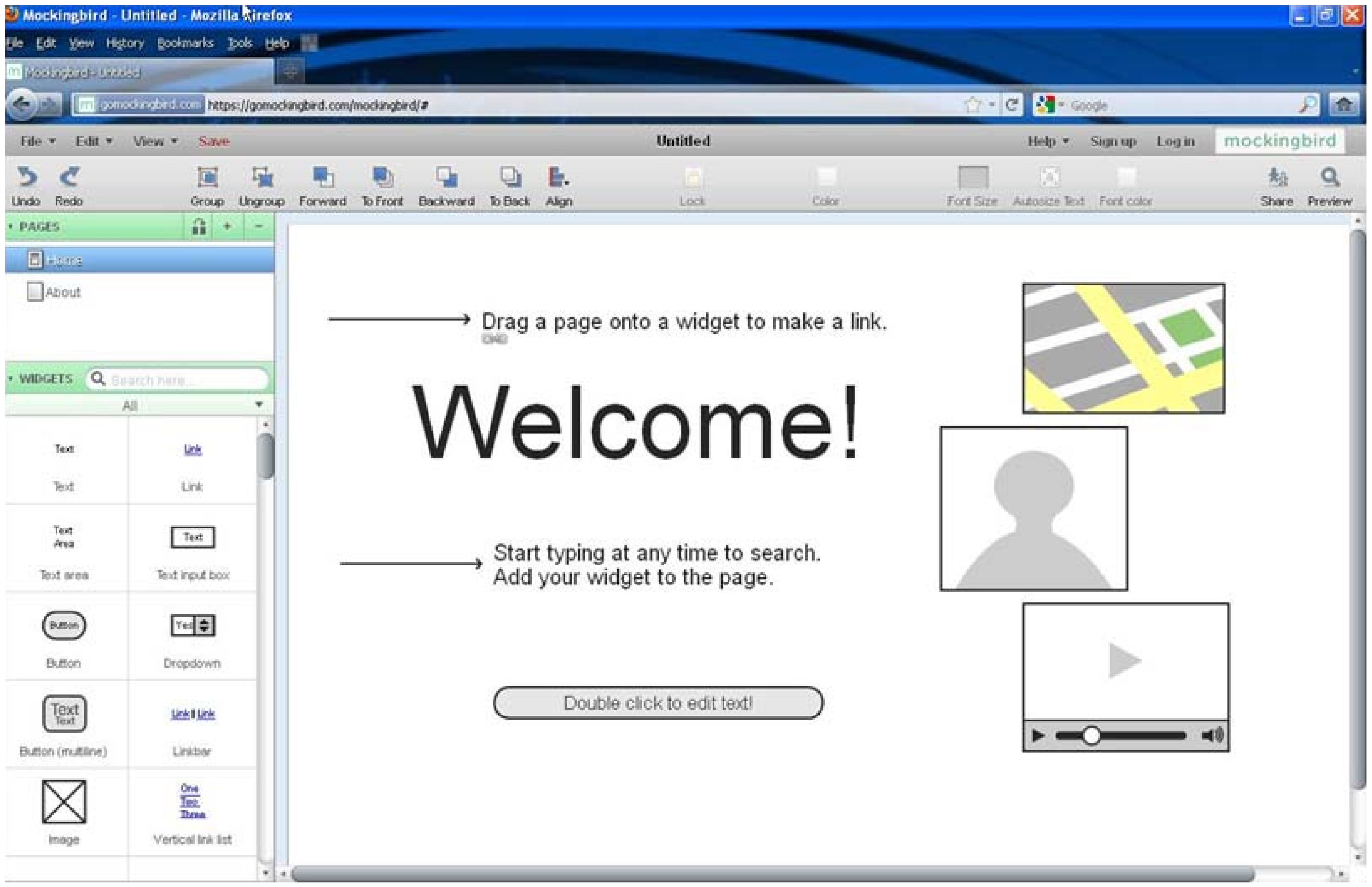

Mockingbird - Untitled - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://gomockingbird.com/mockingbird/#
File Edit View Save
Untitled
Help Sign up Log in mockingbird
Undo Redo Group Ungroup Forward To Front Backward To Back Align Lock Color Font Size Autosize Text Font color Share Preview
PAGES
Home
About
WIDGETS
Search here
All
Text
Link
Text area
Text input box
Button
Dropdown
Button (multiline)
Linkbar
Image
Vertical link list
Drag a page onto a widget to make a link.
Welcome!
Start typing at any time to search.
Add your widget to the page.
Double click to edit text!

Figure 2.5: Snapshot of Mockingbird - a web-based tool supporting mockup creation

—

In this chapter, I looked into three areas related to the topic of this thesis, aiming at underlying present work done and gaps overlooked. First, I described a set of documented challenges and concerns in the design of synchronous collaborative applications. The purpose of this was to get an overview of what is there to expect when initiating such designing efforts. Surely, the survey is not exhaustive, but it aims at supporting the findings further described in this thesis.

Secondly, I looked into design pattern research only to discover that even if there is a large number of collections of patterns available today, there is very little documented methodological support in terms of identifying such patterns. In addition to that, most pattern authors present their collections in the form of pattern languages, but base their reasoning mainly on experience and very little methodology relies behind. I also looked into studies documenting how patterns are used and these studies target mainly the use of patterns in teaching and generally in design. However, patterns are usually used in collaborative contexts since communities are formed around existing collections, so there is the need of investigating how patterns are used by collaborating designers working in teams.

Lastly, I described some of the creative techniques used throughout this work. Even if not directly related to the results and findings described by this thesis, these techniques helped shape the methodology defined and proposed below.

Chapter 3

Synchronous Processes: Motivation and Tool Support

Most research and development of technology to support collaboration has been directed towards asynchronous collaboration contexts [72]. Therefore, today, there is a growing interest in supporting synchronous interaction, as well. Several software applications, developed as either research projects or commercial products, exist and are used today in synchronous collaborative settings in domains such as drawing, searching, text editing, and game solving. The aim of this chapter is to describe a collection of such applications, providing a few examples of tool support and helping frame the landscape this work is looking into.

3.1 Collaborative Drawing

Collaborative drawing is common to several design-oriented domains such as architecture, engineering, and graphic design [80]. It is often that architects or designers gather around a table to draw together, sketching their ideas and communicate based on them. Their processes are characterized by: a) a shared drawing surface accessible to each member of the team, and b) the possibility to mark up their drawings and comment on them. Switching from the real to the virtual and allowing designers to work together in the same time using software tools should not limit their freedom of expression and their creativity. Education is another area in which synchronous collaboration is common. Moreover, distance learning has triggered the development and use of software applications to support such collaborative processes.

3.1.1 Synergo

Synergo [72] is a synchronous collaborative tool used for building several types of diagrammatic representations. Libraries for building flowcharts, entity-relationship diagrams, concept maps, data flow diagrams have been built in the tool; however, its use can go beyond that, allowing different extensions to be added. *Synergo* is targeted to small groups of students, but it also provides analyzing and supervision tools for the teachers. The tool has a built-in chat feature, allowing all team members to exchange ideas and discuss their results. A color-scheme is used to differentiate each collaborator's messages. *Synergo* produces log files containing the actions and the messages exchanged by the members of the group. Based on these log files, the tool allows the playback of the activities performed by the collaborators. Also, log files can be viewed, commented, and annotated. At any time during the collaboration, the tool is able to measure the state of collaboration, defined as "a combination of machine-learning and statistical techniques".

3.1.2 NetDraw

NetDraw [80] is a Java application which provides 2D collaborative drawing features in a client-server architecture (Figure 3.1). It is trying to bridge the gap between complex drawing applications which require users to buy into an entire CAD package and simple whiteboard programs with simple diagramming tools. *NetDraw* has a thin client, suitable for running on any device that supports Java. Any number of users can log in to the server and use their browsers to observe and participate in a synchronous drawing session. The server notifies all the clients of any drawing and editing actions. Drawing and editing actions include: a) group and ungroup objects, b) gesture command for marking the drawing, c) providing objects with different appearances and behaviors. The tool includes an instant messaging feature and it also records snapshots of the drawing in progress. These snapshots may then be played back for reviewing purposes. Drawn objects can be annotated and, during a collaborative session, any of the collaborators may leave or join the process at any time. Notifications of their absence or appearance are sent to those logged in to the session. Objects drawn can be linked to annotations consisting of descriptive text. Gesture objects, which are only temporary, can be drawn as well for marking parts of the actual drawing; they help in clarifying issues during the collaborative process by pointing to those parts. Based on their unique identifiers, the objects drawn are subject to a locking mechanism for ensuring proper coordination among collaborators. Each object's lock status is shown with a color scheme, marking three possible states of an object: 1) waiting to be granted a lock from the server, 2) being locked, and 3) being unlocked.

3.1.3 CO2DE

CO2DE [73] is a collaborative drawing tool which supports the creation of diagrammatic representations, including UML diagrams (Figure 3.2). The application includes a versioning mechanism, providing the collaborators with support in reverting changes. Moreover, whenever a new user joins the collaboration, he is able to navigate through the versioning structure being aware of and comparing different versions. He can analyze contributions and changes made to the shared document, but also messages exchanged among the collaborators and annotations made on the document. Telepointers are another feature of *CO2DE* which allow collaborators to see in which part of the shared workspace other participants are located. Editing the shared document is only permitted after it is locked. At all times, a list with all the users logged in to the application (hence, available for collaboration) is displayed to all the other users. The user who initiates one collaborative session is assigned the role of coordinator throughout the process. An instant messaging feature is included in the application and all the objects collaboratively created may be annotated. All the editing operations of one user are instantly reflected in the other's shared workspace.

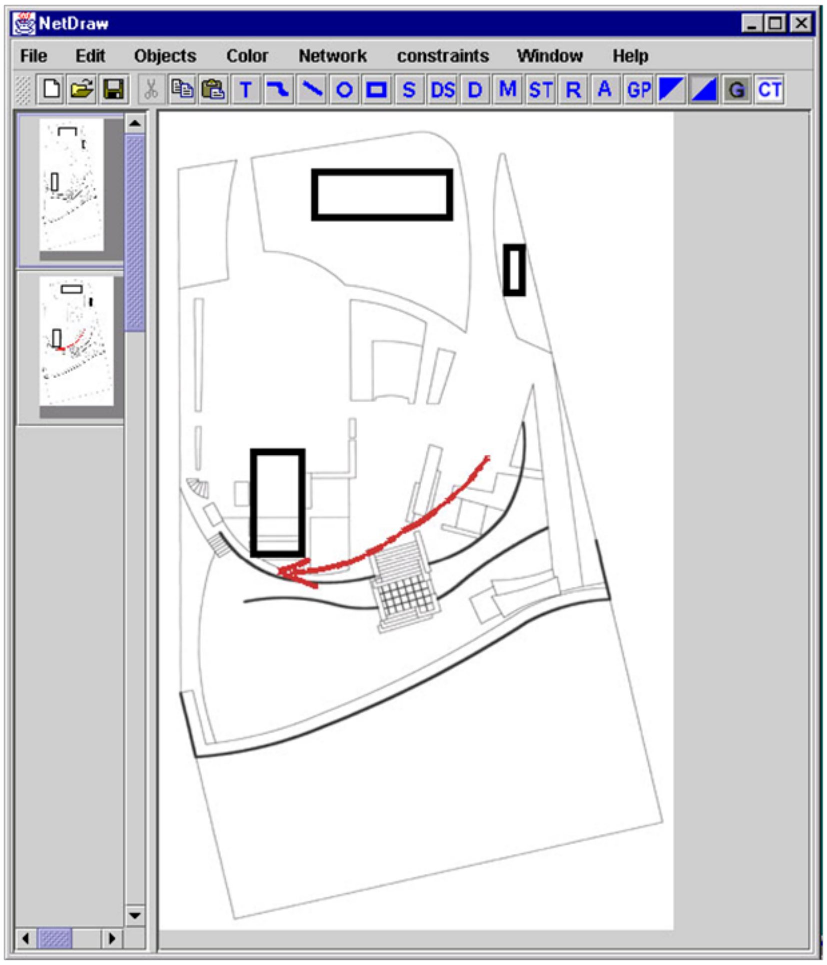
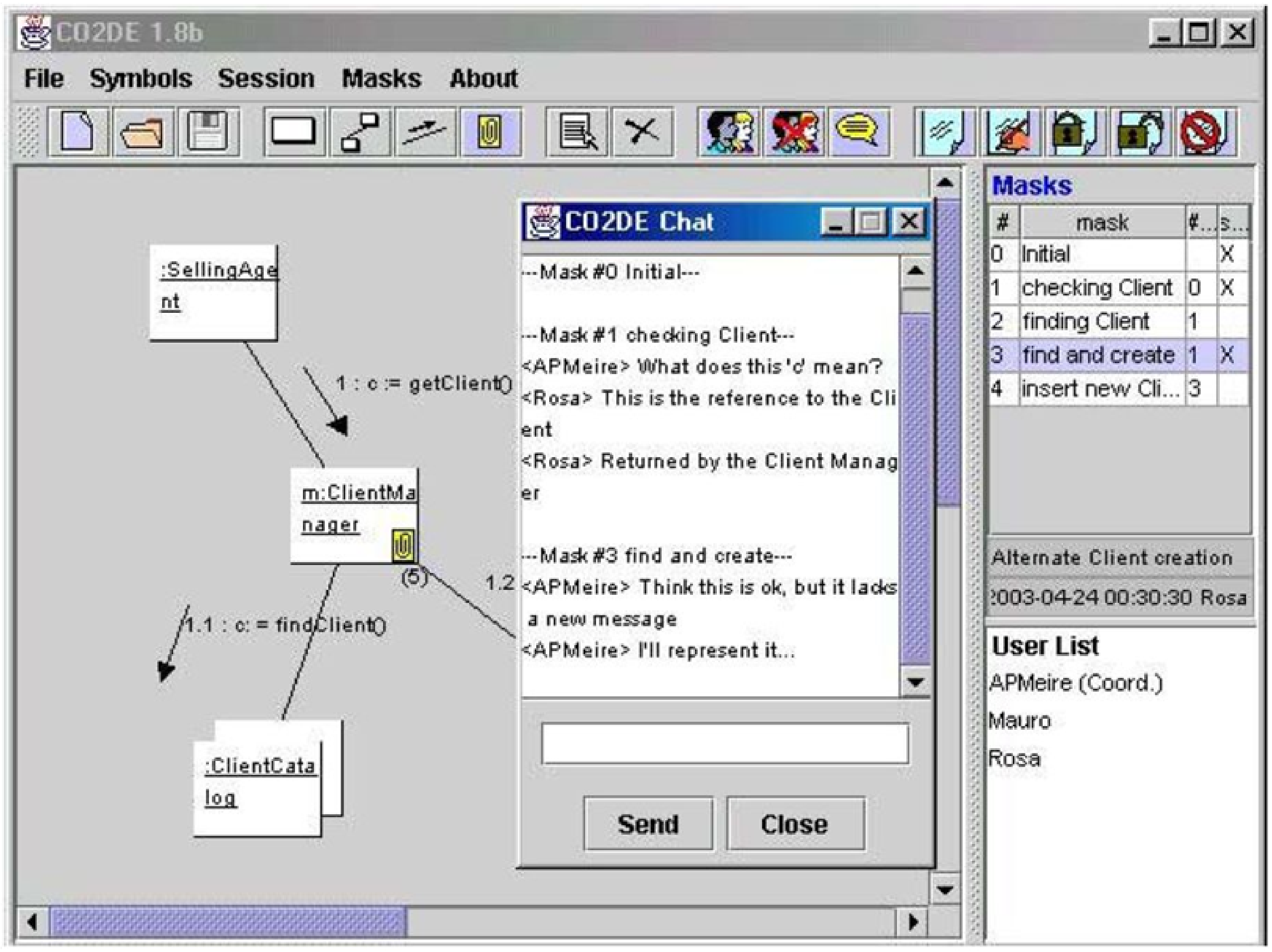


Figure 3.1: Synchronous drawing in NetDraw - a Java application providing 2D collaborative drawing features in a client-server architecture [80]



---Mask #0 Initial---

---Mask #1 checking Client---
<APMeire> What does this 'c' mean?
<Rosa> This is the reference to the Client
<Rosa> Returned by the Client Manager

---Mask #3 find and create---
<APMeire> Think this is ok, but it lacks a new message
<APMeire> I'll represent it...

#	mask	#...	s...
0	Initial		X
1	checking Client	0	X
2	finding Client	1	
3	find and create	1	X
4	insert new Cli...	3	

Figure 3.2: Synchronous drawing in CO2DE - a collaborative drawing tool which supports the creation of diagrammatic representations [73]

3.1.4 LucidChart

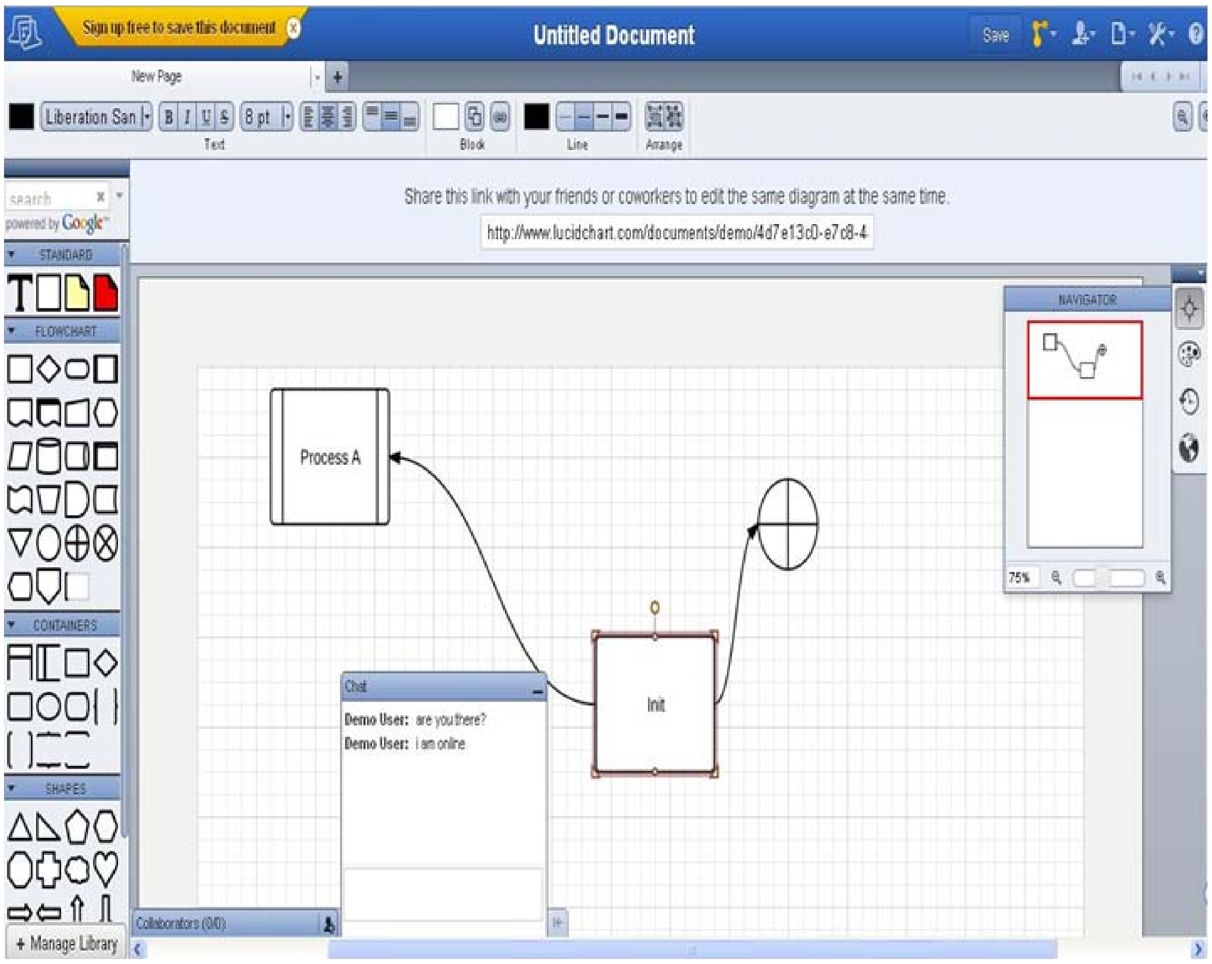
LucidChart (LucidChart, 2008) is a web tool released in 2008 which supports the collaborative drawing of diagrams such as UML diagrams, and flow diagrams (Figure 3.3). Documents collaboratively created may be shared online, published as a web page or shared on social networks. Any user may invite the people s/he chooses to work with and all the changes made by one collaborator are instantly reflected in the others' workspace. The tool includes a revision history feature which supports reverting changes, or starting a new document from a previous version of an existing one. *LucidChart* is available on any device that supports browsing and it allows the export of the documents created to other formats such as pdf, or jpg. Annotations are available for any object collaboratively created. Moreover, groups of collaborators which form a community in itself may choose to create and share a repository of templates in a community library.

3.1.5 DeTransDraw

DeTransDraw (DeTransDraw) is a decentralized collaborative graphical editor which provides the collaborators with a shared drawing area. The tool provides graphic support for notifications about other users' activities. Moreover, collaborators may join and leave the application (hence, the collaborative process) at any time without relying on any central point of control.

3.2 Collaborative Searching

Even if web search is usually considered a solitary activity, there are several real-life scenarios in either educational contexts or among knowledge workers where people collaborate in search tasks [84]. Some of these scenarios include planning travel or events, researching medical conditions, or finding information related to a specific project. As a concrete example, Amershi et al. conclude in [11] that there are many situations in which teachers, librarians, and researchers gather around a single computer to jointly search for information online. On one hand, resource constraints are often a factor,



Share this link with your friends or coworkers to edit the same diagram at the same time.

http://www.lucidchart.com/documents/demo/4d7e13c0-e7c8-4

Demo User: are you there?
Demo User: i am online

Figure 3.3: Synchronous drawing in LucidChart - a web tool which supports the collaborative drawing of diagrams such as UML diagrams, and flow diagrams

since the ratio of, for example, students to computers in public schools is most of the times skewed. On the other hand, even when resource constraints are looser, "the social and pedagogical benefits of face-to-face collaboration and shared viewing of information can be a compelling reason for collaborators to share a single computer" [11]. The results of a survey described in [84] show that often respondents instant-message other people to coordinate real-time Web search or divide responsibility for parts of a search task and then share the results. Moreover, in the absence of a tool to support their collaborative search, respondents have developed their own strategies for that purpose (emailing links back and forth, instant messaging).

3.2.1 CoSearch

CoSearch [11] is a searching tool which supports co-located collaborative Web search (Figure 3.4). Groups of users gather around a single computer, and each user has access to his/her own mouse controlling unique cursor (distinguished by color). For identification purposes, each user is associated with a name and a color. At any time, there are two roles each user might have: 1) driver – actually performing the search, and 2) observer – overlooking the results. These roles are, however, inter-changeable, a user acting as a driver being able to switch to the role of observer and vice versa. Since *CoSearch* allows co-located collaboration, each collaborator can visualize and be aware of what the others are searching for. Notes may be added to each of the websites reached through the search. At any time during a search session, users can choose to create a shared summary of their search results, keeping track of the shared sessions' findings. This also permits tracking the history of a collaborative search process for further analysis. The summaries contain information on the pages' URLs, and the notes associated to each page. *CoSearch* can be used on both desktops and mobile devices. Collaborative features for mobile devices are not fewer, users being able to use the application in a similar way and even collaborate with others using the desktop version of it. However, due to the technical constraints imposed by a mobile device, such as a phone, the interaction flow between the user and the application is simplified. Users have access to a global menu through which they can: a) send a query to the browser, b) get and share the search results, c) get the tabs of other search running in parallel, and d) get the summaries of the other searches performed.

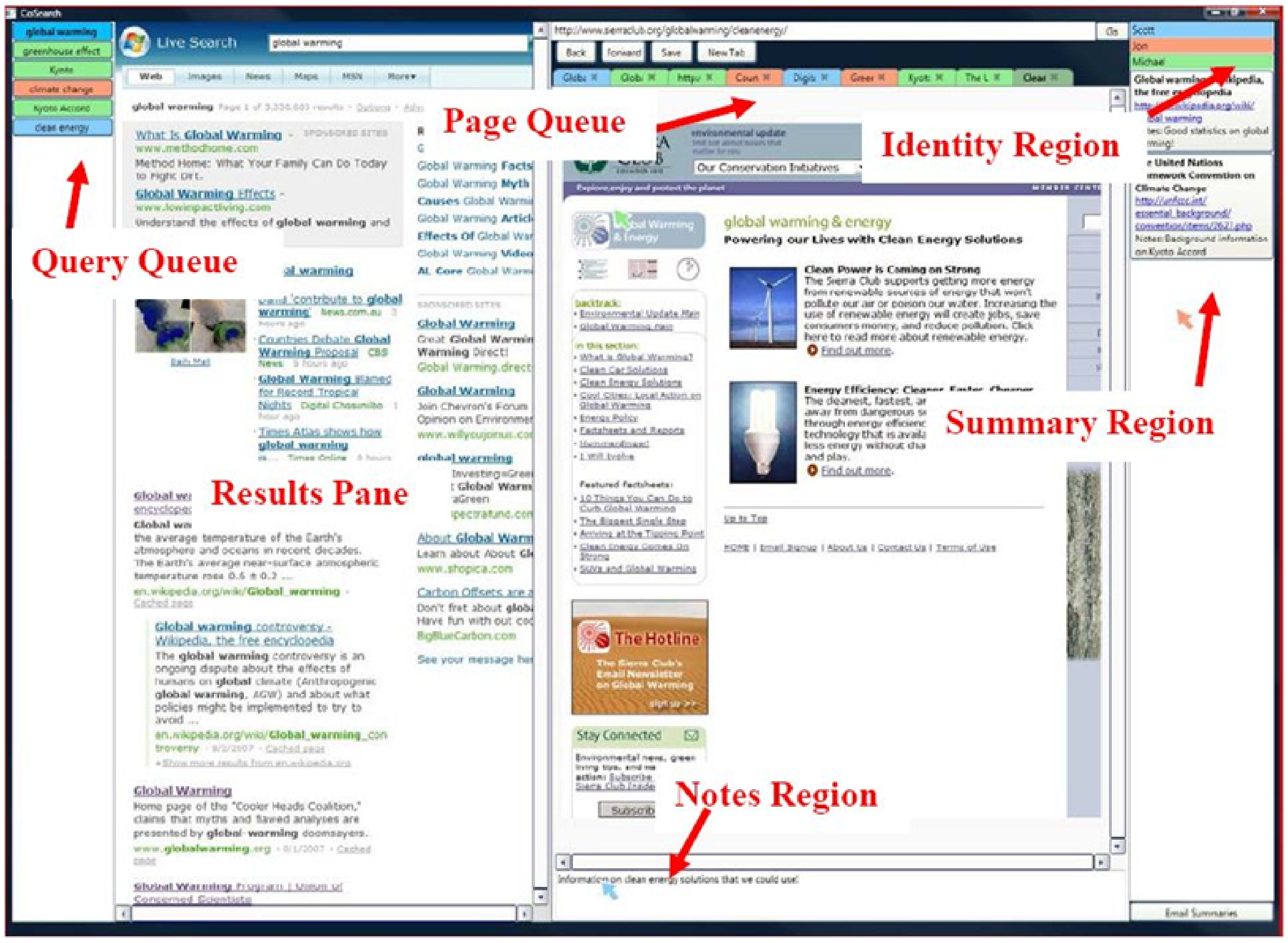
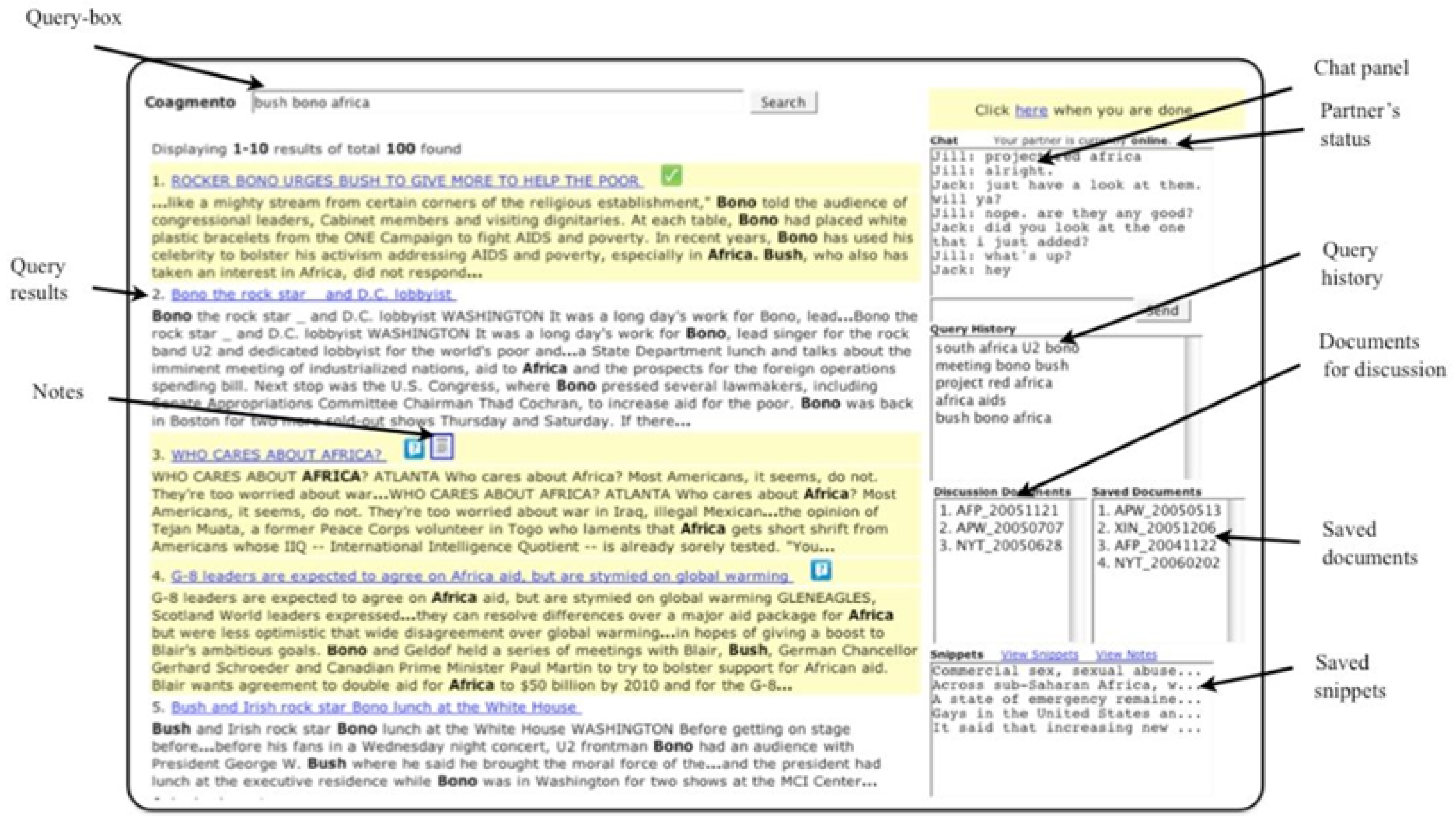


Figure 3.4: Synchronous searching in CoSearch - a searching tool which supports co-located collaborative Web search [11]

3.2.2 Coagmento

Having in mind several documented conditions for a successful collaboration, Shah et al. designed *Coagmento* [97], a tool able to support collaborative information seeking on the Web (Figure 3.5). These conditions include diversity of opinion, independence, decentralization, aggregation, awareness, division of labor, and persistence [101] [84]. *Coagmento* allows two people – either co-located or from remote locations – to work together for seeking information. However, the tool can be used in a collaborative manner as well as individually. The application provides its users with a chat feature for communication. It also displays the collaborators' information, making all collaborators aware of the actions the others are performing. For example, if one document reached through the search is viewed by one of the collaborators, the document is highlighted for both collaborators.

All queries of a session are logged, and documents obtained through the query may be saved or flagged. Therefore, at any time during the search, one user may save a document, flag it as candidate for further discussion and assign it a note. Even if notes proved particularly useful, users would also benefit from simply highlighting and saving portions of a page of a document. In answer to that, *Coagmento* provides ways for users to "snip" passages of documents. Search session states are preserved in *Coagmento*, meaning that if a user leaves the session, he will find it as it was upon return. Changes made by the other collaborator would be updated to the session and at each moment of the collaborative process each user is aware of the presence status of his/her collaborator.



Coagmento bush bono africa Search

Displaying **1-10** results of total **100** found

1. ROCKER BONO URGES BUSH TO GIVE MORE TO HELP THE POOR

...like a mighty stream from certain corners of the religious establishment," **Bono** told the audience of congressional leaders, Cabinet members and visiting dignitaries. At each table, **Bono** had placed white plastic bracelets from the ONE Campaign to fight AIDS and poverty. In recent years, **Bono** has used his celebrity to bolster his activism addressing AIDS and poverty, especially in **Africa. Bush**, who also has taken an interest in Africa, did not respond...

2. Bono the rock star _ and D.C. lobbyist

Bono the rock star _ and D.C. lobbyist WASHINGTON It was a long day's work for Bono, lead...Bono the rock star _ and D.C. lobbyist WASHINGTON It was a long day's work for **Bono**, lead singer for the rock band U2 and dedicated lobbyist for the world's poor and...a State Department lunch and talks about the imminent meeting of industrialized nations, aid to **Africa** and the prospects for the foreign operations spending bill. Next stop was the U.S. Congress, where **Bono** pressed several lawmakers, including Senate Appropriations Committee Chairman Thad Cochran, to increase aid for the poor. **Bono** was back in Boston for two more sold-out shows Thursday and Saturday. If there...

3. WHO CARES ABOUT AFRICA?

WHO CARES ABOUT **AFRICA**? ATLANTA Who cares about Africa? Most Americans, it seems, do not. They're too worried about war...WHO CARES ABOUT AFRICA? ATLANTA Who cares about **Africa**? Most Americans, it seems, do not. They're too worried about war in Iraq, illegal Mexican...the opinion of Tejan Muata, a former Peace Corps volunteer in Togo who laments that **Africa** gets short shrift from Americans whose IIQ -- International Intelligence Quotient -- is already sorely tested. "You...

4. G-8 leaders are expected to agree on Africa aid, but are stymied on global warming

G-8 leaders are expected to agree on **Africa** aid, but are stymied on global warming GLENEAGLES, Scotland World leaders expressed...they can resolve differences over a major aid package for **Africa** but were less optimistic that wide disagreement over global warming...in hopes of giving a boost to Blair's ambitious goals. **Bono** and Geldof held a series of meetings with Blair, **Bush**, German Chancellor Gerhard Schroeder and Canadian Prime Minister Paul Martin to try to bolster support for African aid. Blair wants agreement to double aid for **Africa** to $50 billion by 2010 and for the G-8...

5. Bush and Irish rock star Bono lunch at the White House

Bush and Irish rock star **Bono** lunch at the White House WASHINGTON Before getting on stage before...before his fans in a Wednesday night concert, U2 frontman **Bono** had an audience with President George W. **Bush** where he said he brought the moral force of the...and the president had lunch at the executive residence while **Bono** was in Washington for two shows at the MCI Center...

Click here when you are done

Chat Your partner is currently **online**.

```
Jill: projected africa
Jill: alright.
Jack: just have a look at them.
will ya?
Jill: nope. are they any good?
Jack: did you look at the one
that i just added?
Jill: what's up?
Jack: hey
```

Send

Query History

south africa U2 bono
meeting bono bush
project red africa
africa aids
bush bono africa

Discussion Documents	**Saved Documents**
1. AFP_20051121	1. APW_20050513
2. APW_20050707	2. XIN_20051206
3. NYT_20050628	3. AFP_20041122
	4. NYT_20060202

Snippets View Snippets View Notes

```
Commercial sex, sexual abuse...
Across sub-Saharan Africa, w...
A state of emergency remaine...
Gays in the United States an...
It said that increasing new ...
```

Figure 3.5: Synchronous searching in Coagmento - a tool able to support collaborative information seeking on the Web [97]

3.2.3 SearchTogether

SearchTogether [84] was designed to enable both synchronous and asynchronous remote collaboration in web search (Figure 3.6). The application has a client-server architecture, the server having two major roles: 1) sending shared state among clients, and 2) storing session data in order to enable session persistence. On the client side, each time a user executes a search, the query terms are associated to him/her and this history is synchronized across all group members' clients, creating awareness of the other users' activities. This history is also interactive, allowing each user to click on any of the query terms in order to view the results it produced. Users can rate and comment on the pages searched for. Moreover, each page searched for is associated with a set of metadata which includes information on the visitation (such as the date and time, and the id of the visitor), ratings, and comments. Every time a user views a webpage in the *SearchTogether*'s browser, the page is associated with the date, the time, and the identity of the visitor. Moreover, this information is visible to others, so that if one page has already been visited by a member of the group the others are aware of that and avoid visiting the same page several times. An instant messaging feature able to allow collaborators to discuss the current task and coordinate their efforts is provided by the application. All the conversations are stored and made available to all collaborators for later review. Another collaboration enabling mechanism provided by *SearchToghether* is a recommendation mechanism. Any user may recommend a webpage to a group or another individual. Search results can be obtained in *SearchTogether* in three ways: a) Standard search: a user's query is run and the results are displayed to all the collaborators logged in the session, b) Split search: a user's query is run and the results are divided up among all online group members in a robin-round fashion, facilitating parallelization and avoiding the duplication of efforts, c) Multi-engine search: a user's query is sent to n different search engines, where n is the number of online collaborators. All aspects of a search session in SearchTogether are persistent, including instant messaging conversations, query histories, recommendation queues, and page specific metadata. Therefore, whenever a user pauses his/her work, s/he is able to resume it without losing any information. In addition to that, at the end of any search session, collaborators are allowed to create a shared summary of it. The content of such a summary can be decided by its creator based on predefined criteria.

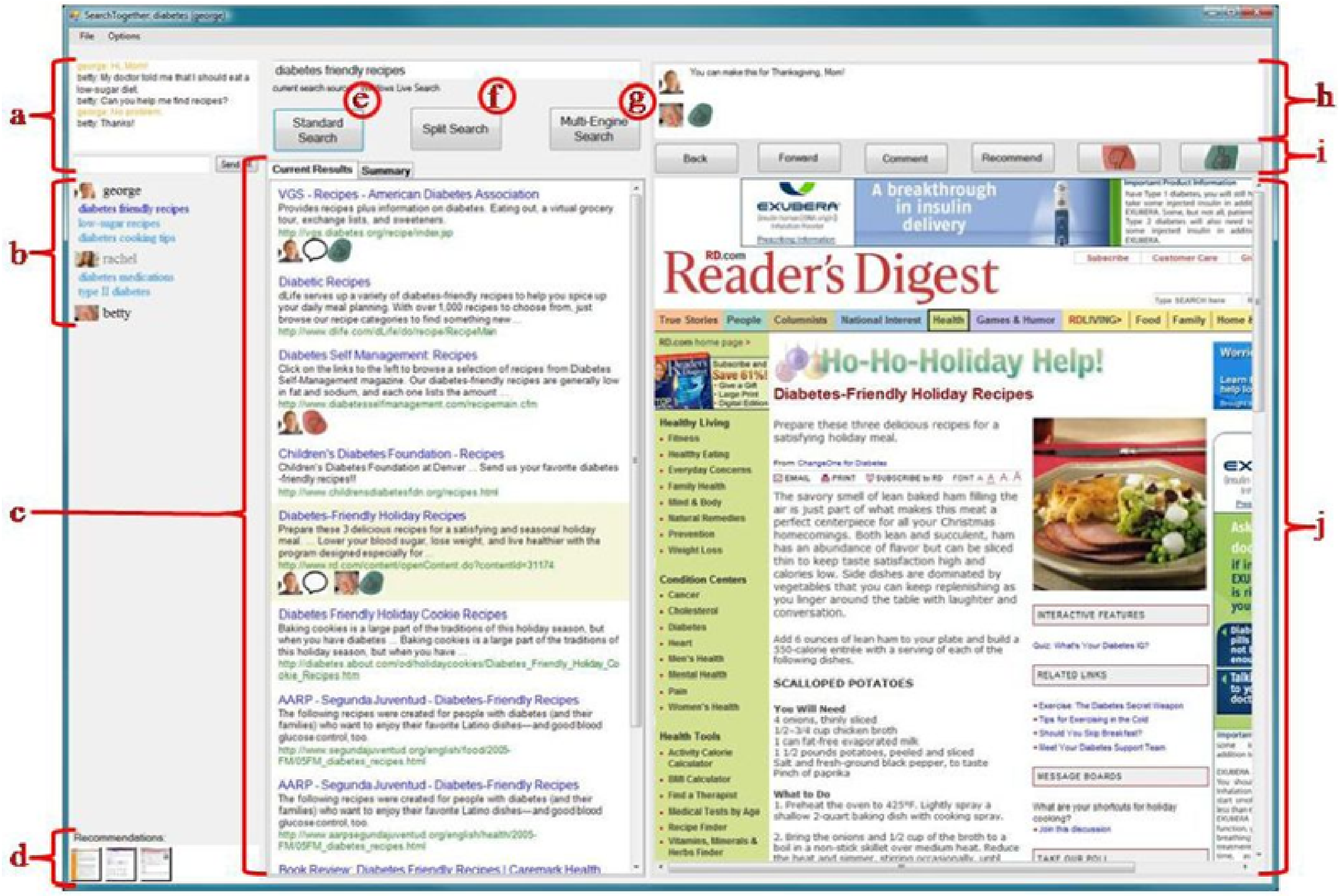


Figure 3.6: Synchronous searching in SearchTogether - a tool which enables both synchronous and asynchronous remote collaboration in web search [84]

3.2.4 Cerchiamo

Cerchiamo [40] is a "collaborative exploratory search system that allows teams of searchers to explore document collections synchronously" in both co-located and remote settings. The system provides each user with a dedicated search interface s/he can use independently of other users. The search process is customized, collaborators having different roles in the process:

- Prospector: the user who issues a query, discovering directions for exploration.
- Miner: the user who browses the results for making relevance judgements on them.

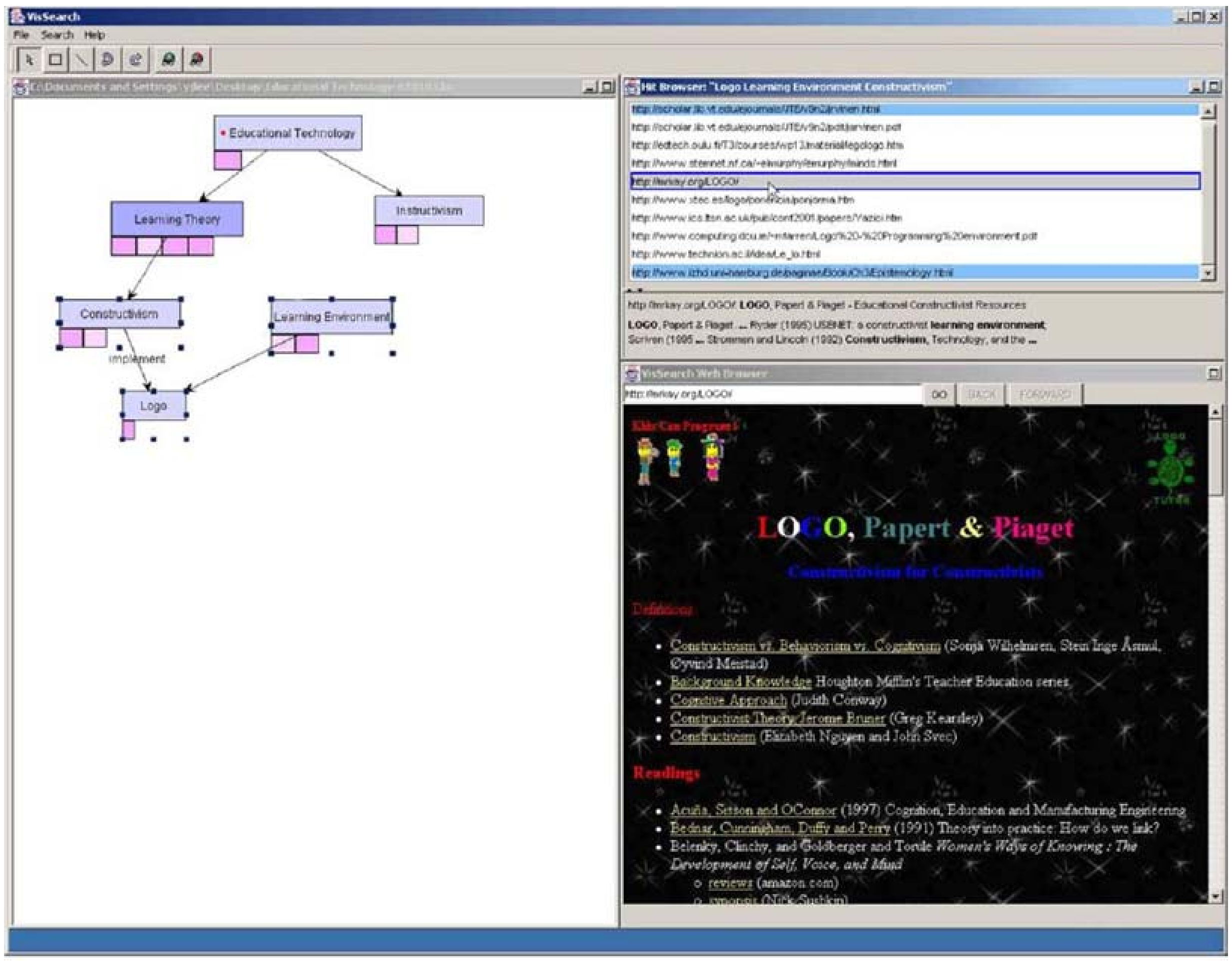
Both prospectors and miners may visualize and interact with a shared display containing information relevant to the progress of the search session.

3.2.5 VisSearch

VisSearch [112] is a collaborative Web searching application which supports sharing Web search results among people with similar interests (Figure 3.7). Searching sessions are evolving collaborative processes which allow collaborators to go back to previous search results and use these results as further queries. In *VisSearch*, this is allowed by representing a search session as a graph where nodes represent search queries and relation links (i.e. relations between pairs of Web search query nodes). Such graphs can be saved and restored, allowing users to track and replay their search process.

Search results can be bookmarked and commented on, text notes being associated to the page. *VisSearch* incorporates a recommendation mechanism able to recommend two types of information:

- Search queries associated with either a Web search query or a URL of a useful Web site.
- URLs and a useful Web site.



http://scholar.lib.vt.edu/ejournals/JTE/v9n2/jarvinen.html
http://scholar.lib.vt.edu/ejournals/JTE/v9n2/pdf/jarvinen.pdf
http://edtech.oulu.fi/T3/courses/wp1.3/materials/legologo.htm
http://www.stemnet.nf.ca/~elmurphy/emurphy/kinds.html
http://erikay.org/LOGO/
http://www.xtec.es/logo/ponencias/ponjornia.htm
http://www.ics.ltsn.ac.uk/pub/conf2001/papers/Yazici.htm
http://www.computing.dcu.ie/~mfarren/Logo%20-%20Programming%20environment.pdf
http://www.technion.ac.il/ideal/Le_jo.html
http://www.izhd.uni-hamburg.de/paginae/Book/Ch3/Epistemology.html

http://erikay.org/LOGO/ **LOGO**, Papert & Piaget - Educational Constructivist Resources

LOGO, Papert & Piaget ... Ryder (1995) USENET: a constructivist **learning environment**, Scriven (1995 ... Stromnen and Lincoln (1992) **Constructivism**, Technology, and the ...

http://erikay.org/LOGO/

Kids Can Program!

LOGO, Papert & Piaget

Constructivism for Constructivists

Definitions

- Constructivism vs. Behaviorism vs. Cognitivism (Sonja Wilhelmsen, Stein Inge Åsmul, Øyvind Meistad)
- Background Knowledge Houghton Mifflin's Teacher Education series
- Cognitive Approach (Judith Conway)
- Constructivist Theory/Jerome Bruner (Greg Kearsley)
- Constructivism (Elizabeth Nguyen and John Svec)

Readings

- Acuña, Sisson and OConnor (1997) Cognition, Education and Manufacturing Engineering
- Bednar, Cunningham, Duffy and Perry (1991) Theory into practice: How do we link?
- Belenky, Clinchy, and Goldberger and Torule *Women's Ways of Knowing : The Development of Self, Voice, and Mind*
  - reviews (amazon.com)
  - synopsis (Nick Sushkin)

Figure 3.7: Synchronous searching in VisSearch - a collaborative Web searching application which supports sharing Web search results among people with similar interests [112]

3.2.6 AntWorld

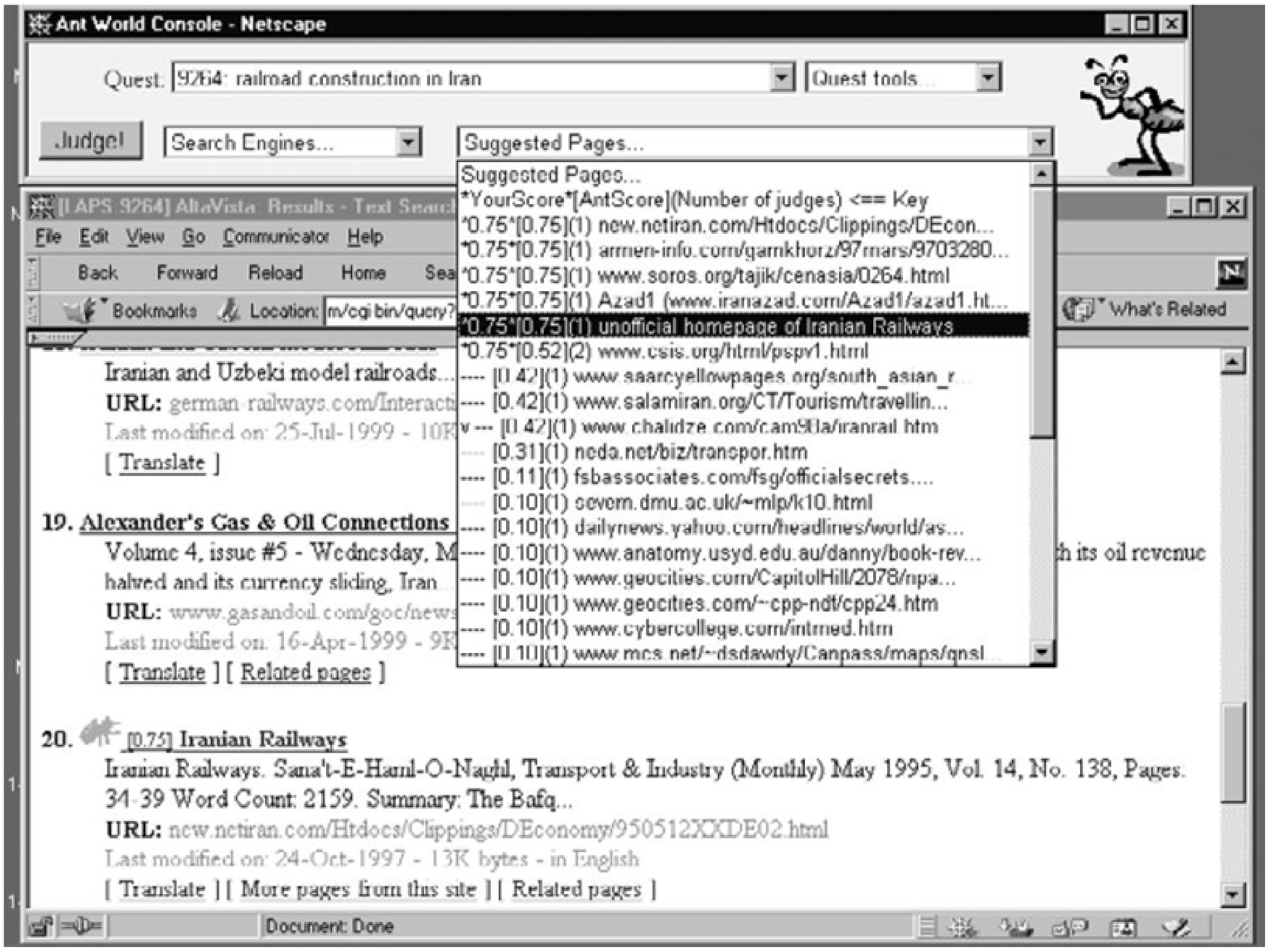
AntWorld [74] is an application developed in 2000 which has as goal making it easier "for the members of a common-interest user group to collaborate in searching the Web" (Figure 3.8). The application has a client-server architecture, where:

- On the client side, each user is provided with ranked lists of suggested web sites, and marked lists to suggested pages from the page the user is currently viewing. Moreover, each user is allowed to provide feedback on the pages visited through ranks, comments and annotations.
- On the server side, database profiles of all *AntWorld* quests ever run by the members of a group are stored. Whenever a user submits a comment on a webpage, the profile of the associated quest is updated and all the other clients are notified by the change.

AntWorld's user interface is designed as a "browser assistant", allowing a user to view suggested pages, provide feedback on a specific page, view similar quests, and edit his/her current quest. Other uses envisioned for *AntWorld* include bookmark and search management, directory creation (lists of links on a specific topic may be created), query refinement or generation.

3.2.7 WeSearch

WeSearch [85] is a tabletop application which supports collaborative Web search among groups of up to 4 collaborating users (Figure 3.9). Each collaborator is associated with a color. Such a color scheme is designed to support the identification of each collaborator's contribution to the search. The application supports touch-based interactions such as moving, rotating, scaling the browser. Pages searched for can be tagged and associated with metadata containing information on the user who reached the page, the type of its content, the URL of the page, and the query keywords used to find the page. A search session may be saved in formats accessible on other devices as well. This allows the visualization of a search session's results on other devices (such as mobile ones). Moreover, the current state of a search session can be stored, allowing


Ant World Console - Netscape
Quest: 9264: railroad construction in Iran
Quest tools...
Judge!
Search Engines...
Suggested Pages...
[LAPS 9264] AltaVista Results - Text Search
File Edit View Go Communicator Help
Back Forward Reload Home Sea
Bookmarks Location: m/cgi bin/query?
What's Related
Document: Done

```
Suggested Pages...
*YourScore*[AntScore](Number of judges) <== Key
*0.75*[0.75](1) new.netiran.com/Htdocs/Clippings/DEcon...
*0.75*[0.75](1) armen-info.com/garnkhorz/97mars/9703280...
*0.75*[0.75](1) www.soros.org/tajik/cenasia/0264.html
*0.75*[0.75](1) Azad1 (www.iranazad.com/Azad1/azad1.ht...
*0.75*[0.75](1) unofficial homepage of Iranian Railways
*0.75*[0.52](2) www.csis.org/html/pspv1.html
---- [0.42](1) www.saarcyellowpages.org/south_asian_r...
---- [0.42](1) www.salamiran.org/CT/Tourism/travellin...
v --- [0.42](1) www.chalidze.com/cam98a/iranrail.htm
---- [0.31](1) neda.net/biz/transpor.htm
---- [0.11](1) fsbassociates.com/fsg/officialsecrets....
---- [0.10](1) severn.dmu.ac.uk/~mlp/k10.html
---- [0.10](1) dailynews.yahoo.com/headlines/world/as...
---- [0.10](1) www.anatomy.usyd.edu.au/danny/book-rev...
---- [0.10](1) www.geocities.com/CapitolHill/2078/npa...
---- [0.10](1) www.geocities.com/~cpp-ndt/cpp24.htm
---- [0.10](1) www.cybercollege.com/intmed.htm
---- [0.10](1) www.mcs.net/~dsdawdy/Canpass/maps/gnsl...
```

Iranian and Uzbeki model railroads...
URL: german-railways.com/Interacti
Last modified on: 25-Jul-1999 - 10K
[Translate]

19. **Alexander's Gas & Oil Connections**
Volume 4, issue #5 - Wednesday, M ... th its oil revenue halved and its currency sliding, Iran
URL: www.gasandoil.com/goc/news
Last modified on: 16-Apr-1999 - 9K
[Translate] [Related pages]

20. [0.75] **Iranian Railways**
Iranian Railways. Sana't-E-Haml-O-Naghl, Transport & Industry (Monthly) May 1995, Vol. 14, No. 138, Pages: 34-39 Word Count: 2159. Summary: The Bafq...
URL: new.netiran.com/Htdocs/Clippings/DEconomy/950512XXDE02.html
Last modified on: 24-Oct-1997 - 13K bytes - in English
[Translate] [More pages from this site] [Related pages]

Figure 3.8: Synchronous searching in AntWorld - an application which has as goal making it easier "for the members of a common-interest user group to collaborate in searching the Web" [74]

collaborators to pause it and resume it later on.

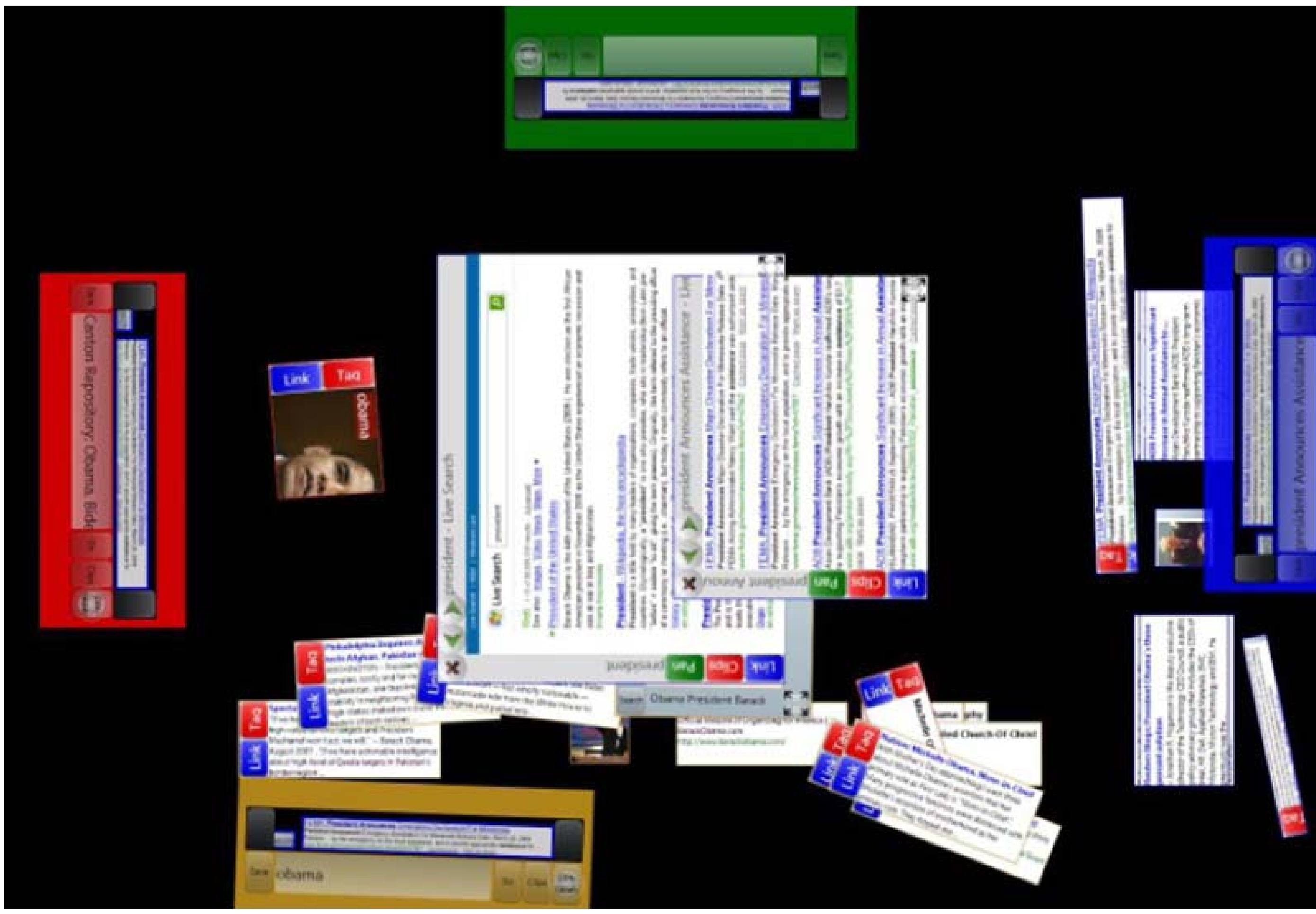


Figure 3.9: Synchronous searching in WeSearch - a tabletop application which supports collaborative Web search among groups of up to 4 collaborating users [85]

3.3 Collaborative Text Editing

Another frequent area of synchronous collaboration is text editing, contexts where more collaborators edit together the same document being met often [6]. Such editing processes go through various stages of refinement, and asynchronous tools such as e-mail prove to be too cumbersome and inefficient. On one hand, multiple copies of the same document can lead to confusion. On the other hand, users might need to collaboratively decide on changes to be performed on the document and discuss this changes in real-time.

Twelve challenges in the design of synchronous collaborative editing software are identified in [6]:

- Time and space: synchronous applications need to take into account issues such as the physical distribution of the potential users.
- Awareness: various methods for promoting awareness have been developed and used. Some of them include: using different colors for each collaborator's input, sending a notification whenever a document is modified, showing each collaborator's cursor in a synchronous system, or showing the status of each collaborator.
- Communication: synchronous applications are particular candidates for being integrated with communication tools (such as instant messaging systems). However, there are both advantages and disadvantages to be considered for such integrations. On one hand, they place the conversation in the right context, supporting the common understanding among collaborators. On the other hand, messages from other collaborators may be seen as a source of disruption.
- Private and shared work spaces: the issue of privacy is particularly important in synchronous applications where users can see each other's contributions immediately. Therefore, users might need to keep part of their work private. This, however, may influence their commitment to using the system collaboratively.
- Intellectual property: seldom discussed as an issue, intellectual property is becoming one in the context of collaborative applications where artifacts are being shared and collaboratively created. Labeling and tracking artifacts for which the authors have assigned special rights should be possible.
- Simultaneity and locking: synchronous collaboration often leads to the situation in which more collaborators edit one shared document in the same time. Mutual exclusion allows only one collaborator to edit a shared artifact at any time, while

locking allows several people to work on a document at the same time by making the part of the document that one person is working on unavailable to others.

- Protection: protecting the work of all collaborators is a major concern in synchronous applications, since deleting one's contributions may delay the collaborative process and lead to conflict. Some of the protection mechanisms used in text editing synchronous applications include undo, change tracking, and version control.
- Workflow: in situations where the collaborative development of documents is in itself a managerial activity, organizations need assistance in correctly controlling workflow processes. A concrete example of such a situation is a document needing the signatures of A and B before being approved by C.
- File format: file format issues can introduce significant difficulties for collaborative applications, which must either store information in a binary format, or convert contents to and from these formats.
- Platform independence: two classes of collaborative applications can be identified: 1) network-based collaborative tools which allow users to use different computing platforms, and 2) web-based systems which use web browsers with different capabilities.
- User benefit: collaborative tools only prove successful if they provide their users with an overall benefit, whether this is ease of use, flexibility, or any other gain. Making a tool beneficial for its users so that they find enough motivation to be part collaborative processes is a challenge addressed often in the literature.

3.3.1 TellTable

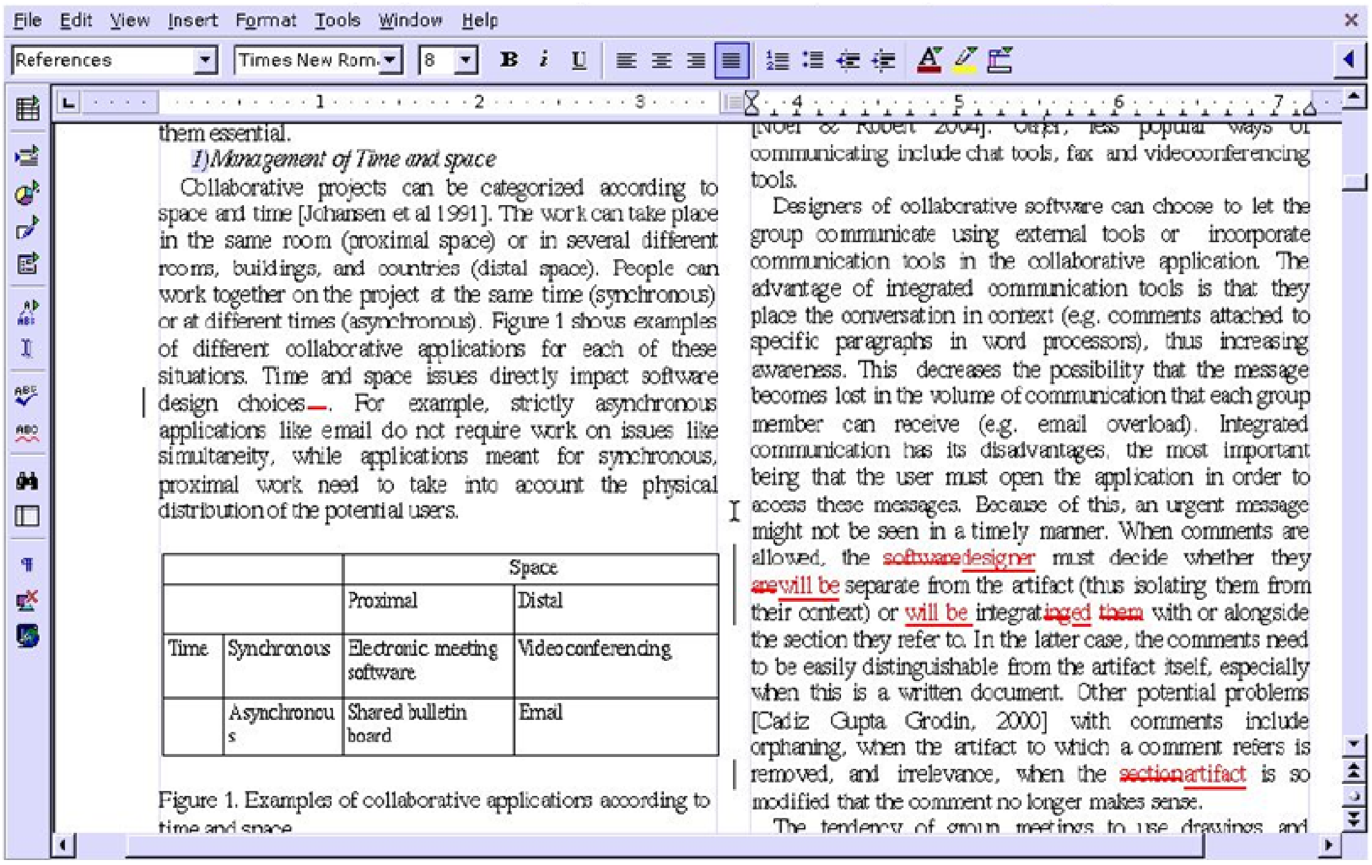
TellTable [6] tries to answer some of the challenges described above, being a "single user application to be used collaboratively by running the application on a server and allowing users to access it from a java-enabled web browser" (Figure 3.10). Upon logging in, users are provided with a list of files to edit. When a file is selected for being edited, it is opened on the server (using OpenOffice), and the display is exported to a browser via a network protocol (VNC). Further on, the user's browser loads a java VNC viewer, and all keyboard and mouse activity in the applet are sent to the server. The full history of the development of a document is being maintained by a versioning system (CVS), but there is no communication tool integrated in *TellTable*. Coordination is supported in two ways: a). by completely locking the document once a user starts editing it, or b). by allowing a writer to share the document edited with others and agreeing as a group on a form of coordination. *TellTable* supports awareness by displaying the current version number and giving access to past versions or change tracking. By placing the application on a central server, *TellTable* ensures that all users use the same version of the application.

3.3.2 CodoxWord

Released in 2007, *CodoxWord* (CodoxWord, 2010) is a real time sharing and group editing tool (Figure 3.11). Upon installing *CodoxWord*, Microsoft Word is enhanced with real-time collaborative capabilities, the sharing of documents being possible in a Peer2Peer mode. Instant updates of the others' activities are available and these updates are either visual or audio. Coloring schemes help highlight concurrent edits on the shared document. The application embeds a versioning system which supports recovering from errors, backtracking to previous versions, and reverting changes made to a document without affecting the work of the other collaborators. *CodoxWord* automatically merges all the simultaneous edits, no matter where they take place in the shared document, which asks for the collaborators to coordinate their edits in a consistent manner. However, the tool provides several conflict resolution policies and strategies. Different roles and rights may be assigned to each collaborator, so the collaboration process is subject to customization. For example, some users may be assigned reading only rights, while others may also edit the content of the document.

3.3.3 EtherPad

EtherPad (EtherPad, 2008) is a web-based collaborative real-time editor (Figure 3.12). Each user is assigned a name and a color to support the identification of individual


File Edit View Insert Format Tools Window Help

References | Times New Rom. | 8

them essential.

1) Management of Time and space

Collaborative projects can be categorized according to space and time [Johansen et al 1991]. The work can take place in the same room (proximal space) or in several different rooms, buildings, and countries (distal space). People can work together on the project at the same time (synchronous) or at different times (asynchronous). Figure 1 shows examples of different collaborative applications for each of these situations. Time and space issues directly impact software design choices. For example, strictly asynchronous applications like email do not require work on issues like simultaneity, while applications meant for synchronous, proximal work need to take into account the physical distribution of the potential users.

		Space	
		Proximal	Distal
Time	Synchronous	Electronic meeting software	Videoconferencing
	Asynchronous	Shared bulletin board	Email

Figure 1. Examples of collaborative applications according to time and space

[Noel & Robert 2004]. Other, less popular ways of communicating include chat tools, fax and videoconferencing tools.

Designers of collaborative software can choose to let the group communicate using external tools or incorporate communication tools in the collaborative application. The advantage of integrated communication tools is that they place the conversation in context (e.g. comments attached to specific paragraphs in word processors), thus increasing awareness. This decreases the possibility that the message becomes lost in the volume of communication that each group member can receive (e.g. email overload). Integrated communication has its disadvantages, the most important being that the user must open the application in order to access these messages. Because of this, an urgent message might not be seen in a timely manner. When comments are allowed, the ~~software~~designer must decide whether they ~~are~~will be separate from the artifact (thus isolating them from their context) or will be integrat~~ing~~ed ~~them~~ with or alongside the section they refer to. In the latter case, the comments need to be easily distinguishable from the artifact itself, especially when this is a written document. Other potential problems [Cadiz Gupta Grodin, 2000] with comments include orphaning, when the artifact to which a comment refers is removed, and irrelevance, when the ~~section~~artifact is so modified that the comment no longer makes sense.

The tendency of group meetings to use drawings and

Figure 3.10: Synchronous text editing in TellTable - a collaborative text editing tool used by running the application on a server and allowing users to access it from a java-enabled web browser [6]

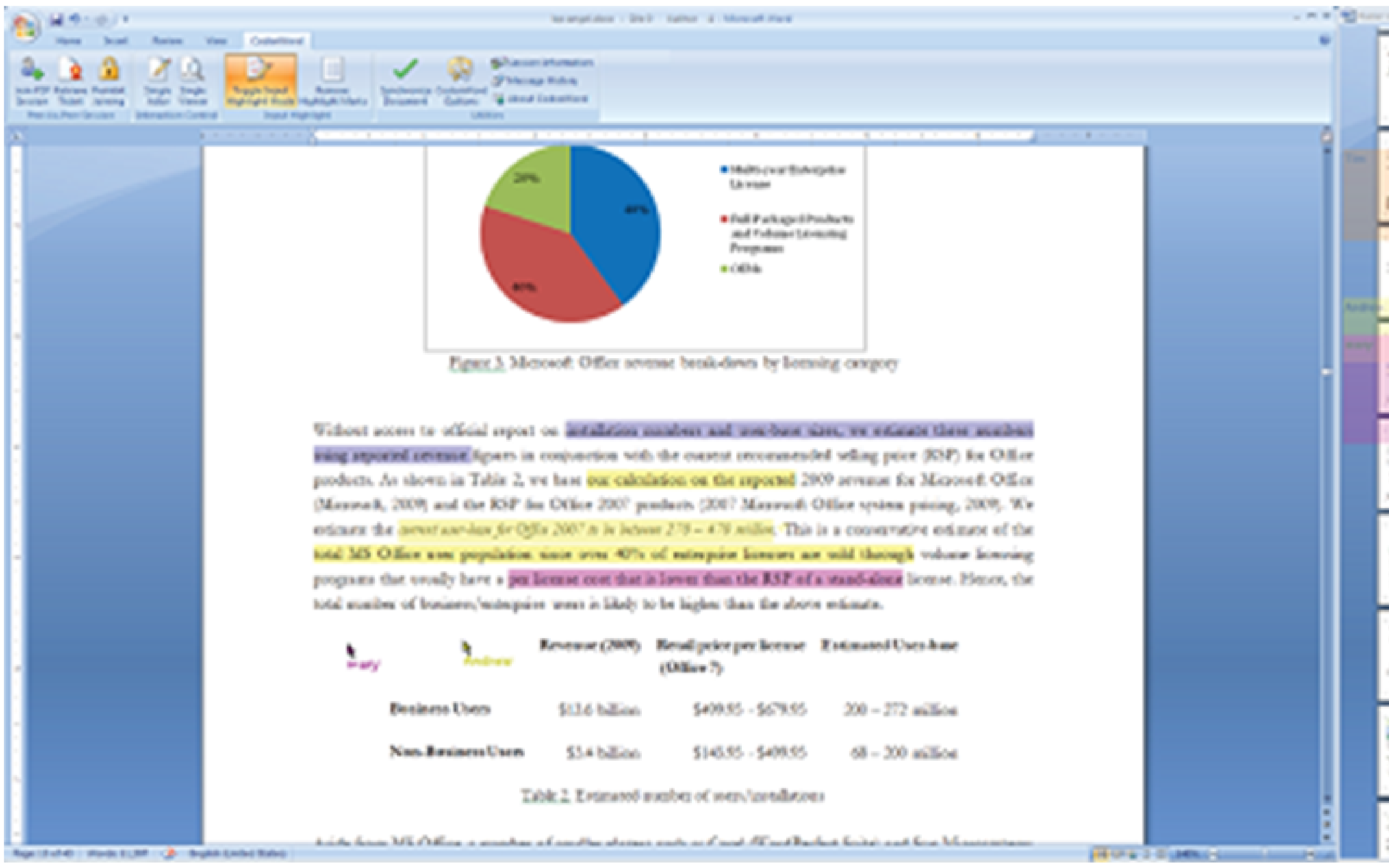


Figure 3. Microsoft Office revenue break-down by licensing category

Without access to official report on installation numbers and user-base sizes, we estimate these numbers using reported revenue figures in conjunction with the current recommended selling price (RSP) for Office products. As shown in Table 2, we base our calculation on the reported 2009 revenue for Microsoft Office (Microsoft, 2009) and the RSP for Office 2007 products (2007 Microsoft Office system pricing, 2009). We estimate the *current user-base for Office 2007 to be between 270 – 470 million*. This is a conservative estimate of the total MS Office user population since over 40% of enterprise licenses are sold through volume licensing programs that usually have a per license cost that is lower than the RSP of a stand-alone license. Hence, the total number of business/enterprise users is likely to be higher than the above estimate.

	Revenue (2009)	Retail price per license (Office 7)	Estimated Users-base
Business Users	$13.6 billion	$409.95 - $679.95	200 – 272 million
Non-Business Users	$3.4 billion	$145.95 - $409.95	68 – 200 million

Table 2. Estimated number of users/installations

Figure 3.11: Synchronous text editing in CodoxWord - a real time sharing and group editing tool

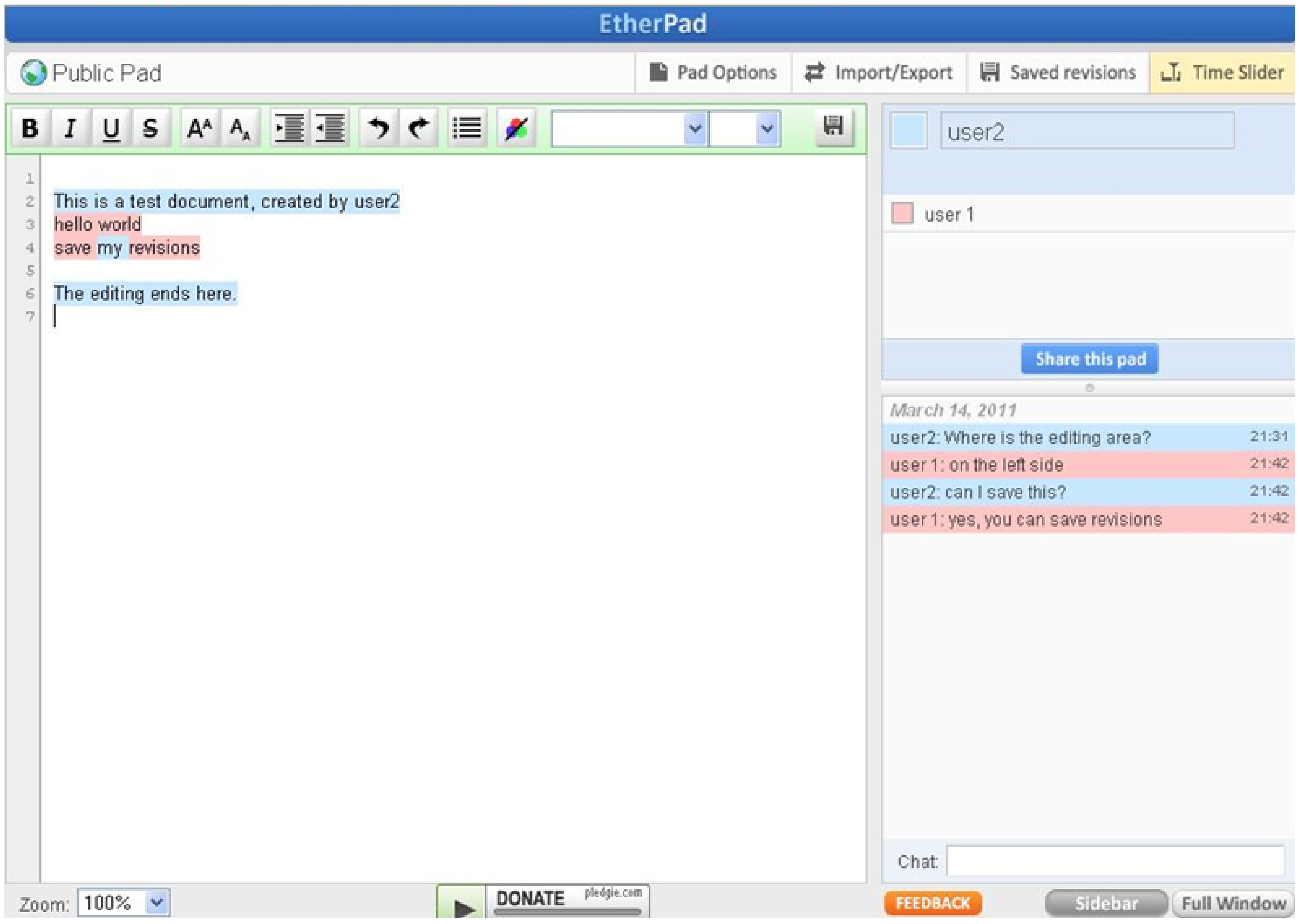
contributions to the document edited. Communication among collaborators is made available through an instant messaging feature included in the application. Any user may invite the collaborators s/he chooses to work with through e-mail invites. The document edited may be exported to other formats such as html, txt, pdf, doc and the editing process may be customized. As result of that, all the collaboration options (such as visualization settings) set by one collaborator affect the view of all the other collaborators. *EtherPad* provides the collaborators with the possibility of saving the revisions made on a document, so that the history of the collaborative process is tracked. Previous versions of a document may either be viewed on a time slider or restored. Moreover, any user may see at all times the list of all users logged in to the application.

3.3.4 GoogleDocs

Synchronous collaborative text editing is supported also by *GoogleDocs* (GoogleDocs). The tool has a chat feature integrated and it allows each participant to invite his/her collaborators. Revisions show at any time who changed what and when within the document edited. *GoogleDocs* supports web based collaboration, allowing each user to set the level of privacy desired. In this way, the tool allows collaborative text editing as well as individual text editing.

3.4 Collaborative Game Solving

Games have been played for literally thousands of years, being an important part of human life. Technology has brought forth a multitude of gaming alternatives, some targeting multiplayer use and focusing on collaborative problem solving. Even more, games create a social situation characterized by players sitting together around the same table, looking at each other to interpret mimics and gestures which may help them understand the others' actions [69]. Besides their social character, games have proven to be useful in developing "educational and rehabilitation tools to support learning" [20]. However, the positive effects of using technology for such learning purposes



This is a test document, created by user2
hello world
save my revisions

The editing ends here.

March 14, 2011
user2: Where is the editing area? 21:31
user 1: on the left side 21:42
user2: can I save this? 21:42
user 1: yes, you can save revisions 21:42

Figure 3.12: Synchronous text editing in EtherPad - a web-based collaborative real-time editor

are somehow neutralized by the fact that most such games are designed for one player working directly with the application, not supporting interaction between more players. Even if not yet fully explored, a solution to this issue is represented by computerized systems that can support co-located interaction of multiple players.

3.4.1 Mystery at the Museum

"*Mystery at the Museum*" (M@M) [61] is a synchronous collaborative game meant to engage visitors in museum exhibits and to encourage them to collaborate in solving a detective problem, i.e. a band of thieves had left their calling card in an exhibit case indicating that they had stolen a priceless object from the museum and replaced it with a replica (Figure 3.13). The players have been brought in as a team of experts to try to solve the crime, apprehend the criminals, and identify and retrieve the stolen artifact. Players take one of three possible roles during the problem solving: a technologist, a biologist, and a detective. The game was completed when players had accumulated enough evidence to obtain a virtual warrant for the arrest of the culprits. The system is adapted to Pocket PCs. It is able to track the individual contribution of each collaborator to the problem solving and it provides a chat feature.

3.4.2 Collaborative Puzzle Game

Collaborative Puzzle Game (CPG) [20] is a tabletop interactive game addressing children with Autism Spectrum Disorder (ASD) (Figure 3.14). Two players can work together on the same display, dragging and dropping pieces of the puzzle in an area of the screen designated for solving the puzzle. The coordination of the players is enforced in that a piece of the puzzle can only be moved when both players drag it. Any player can, at any time, visualize what his/her collaborator is doing. Feedback on the players' actions is provided through both visual and auditory media. For example, during the game, an animation is executed when a player tries to release a puzzle piece over another piece that is already anchored to the solution area (the piece being released jumps away with a spring sound and moves to a random position on the surface). Moreover, when a piece is released in an incorrect position on the solution

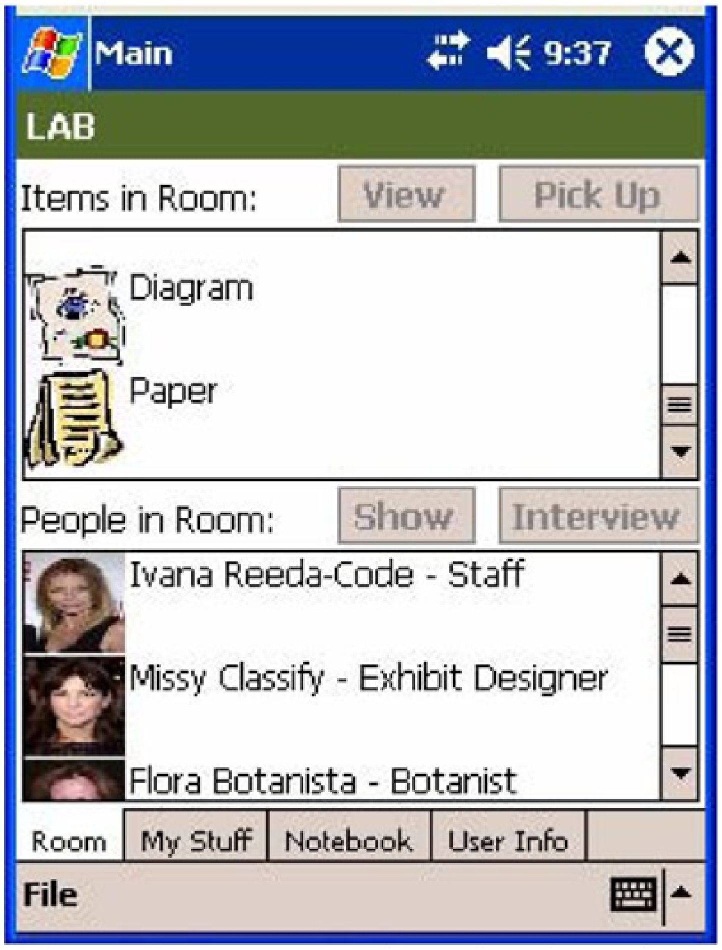


Figure 3.13: Synchronous game solving in M@M - a synchronous collaborative game meant to engage visitors in museum exhibits and to encourage them to collaborate in solving a detective problem [61]

area, an unpleasant buzz is played and a red halo surrounds the piece until it is removed.

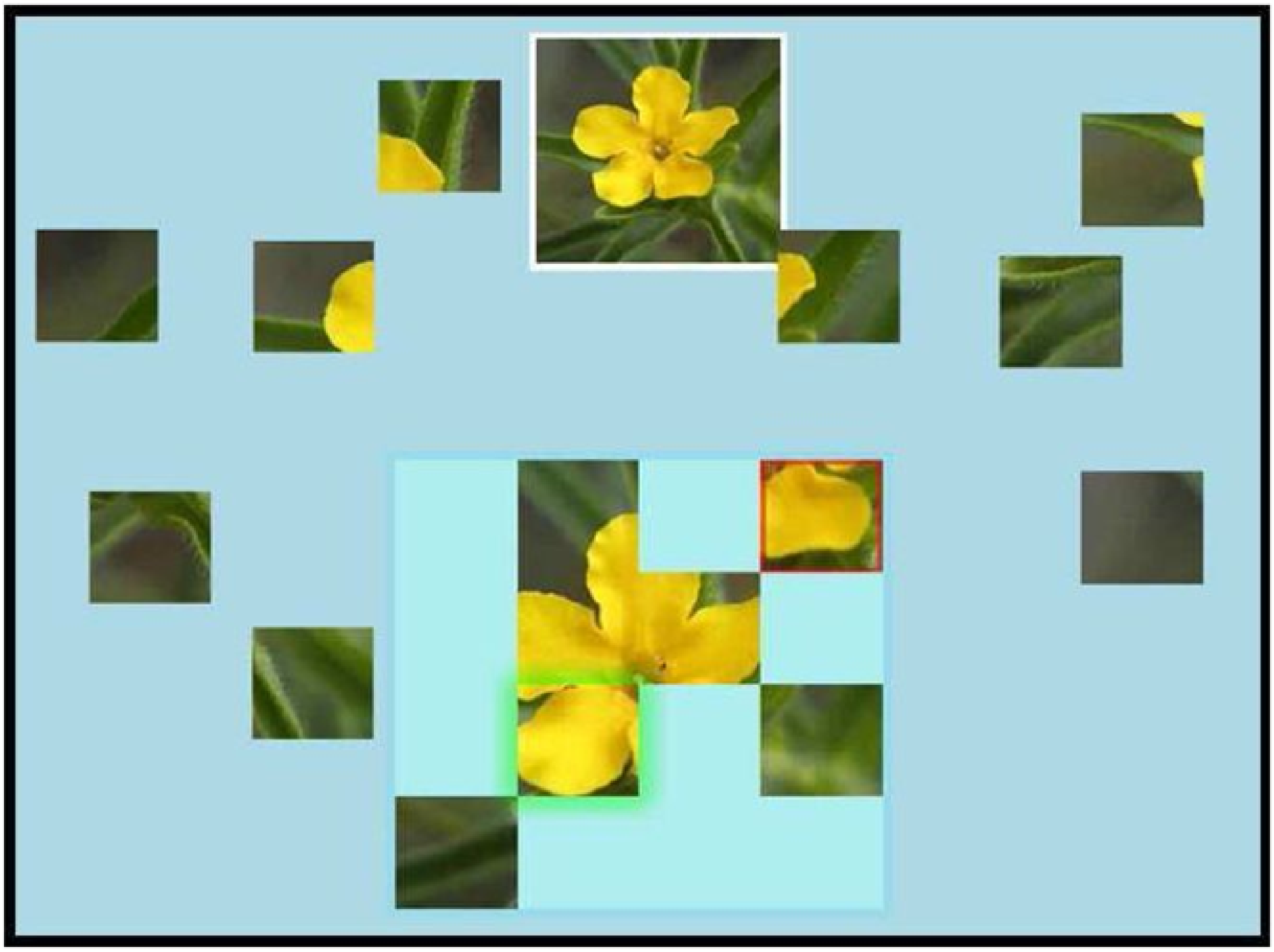


Figure 3.14: Synchronous game solving in CPG - a tabletop interactive game addressing children with Autism Spectrum Disorder[20]

3.4.3 SIDES

Sides [78] is a collaborative multi-player tabletop puzzle-style computer game that "encourages meaningful application of group work skills" (Figure 3.15). At the beginning of a game, each player receives nine square tiles with arrows. Arrows are divided among participants and they are asked to work together to build a path with their pieces to allow a frog" to travel from the start lily pad to the finish lily pad. Each player has a control panel containing information on the state of the game, voting buttons to test the solution of the game, reset, or quit the game. The coordination mechanism embedded in the application ensures the participation of all the players by allowing them to move to a new state in the game solely after all the players have given their vote on the current state of the game. The game does not enforce rules such as turn taking or piece ownership.

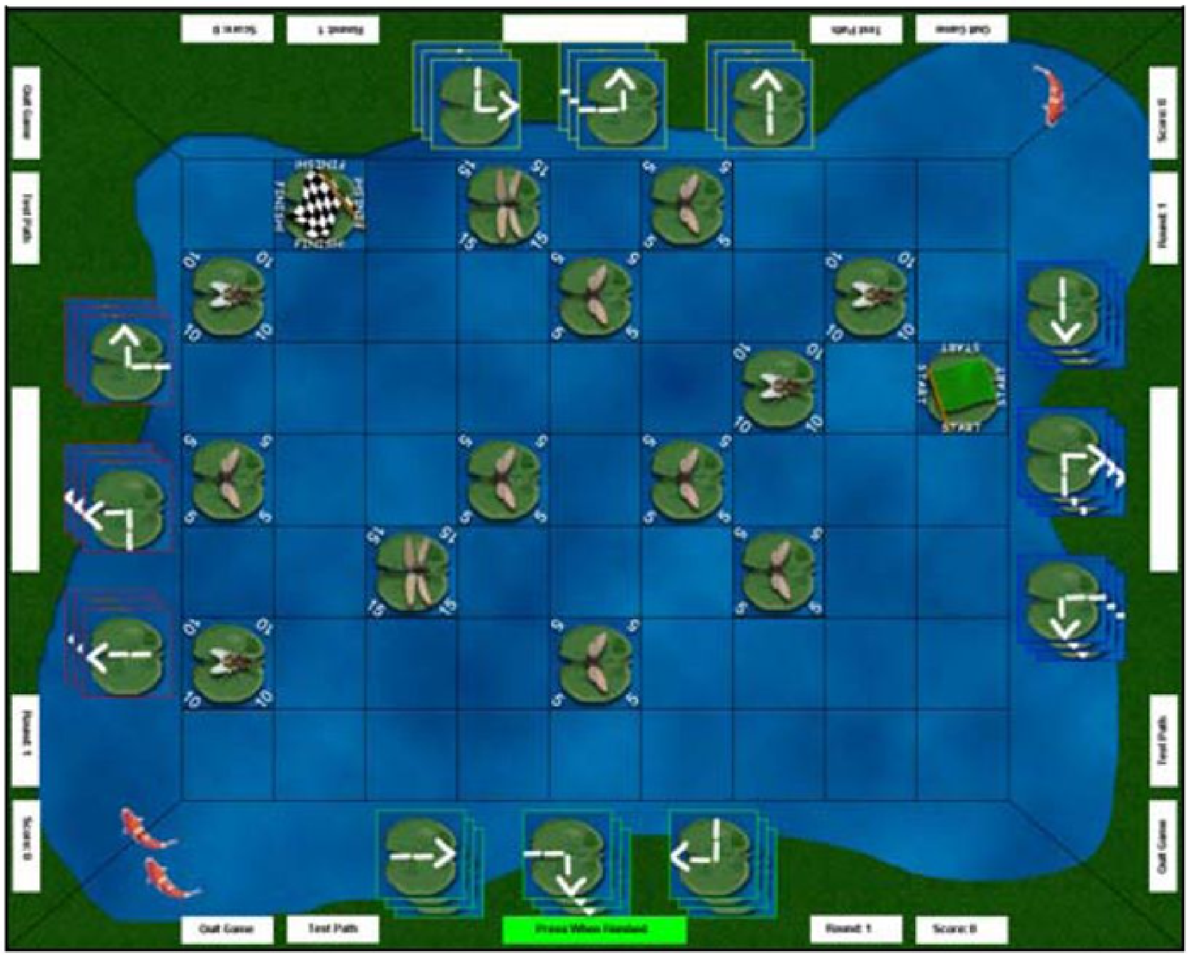


Figure 3.15: Synchronous game solving in Sides [78] - a collaborative multi-player tabletop puzzle-style computer game

3.4.4 STARS

STARS [69] is a software platform used for realizing computer augmented tabletop games. The platform supports the players in coordinating themselves as a community and it considers persistency in that it record the game events and allows the creation of a game history. Also, players are supported in customizing the game. Both private and public communication is available through instant messaging features. Two games have been developed using the platform: Monopoly and KnightMage. The first one embeds private communication features and mechanisms to support competition among players. The second one uses both audio and visual channels for capturing and transmitting the information.

—

In this chapter, I aimed at describing a few examples of synchronous collaborative applications which are used today in domains such as drawing, searching, text editing, and game solving. These applications form the basis of further investigations described later on in Chapter 4. For now, though, these descriptions are aimed to help the reader get a better picture of the type of applications this thesis is focusing on and on the overall technological landscape it addresses.

Chapter 4

Identifying Patterns: A Collection of Patterns for the Design of Synchronous Applications

Although there are lots of collections of design patterns described and available, there is very little indication on the methods the authors of these collections used to reach the patterns. Surely, as most of them explain, most of the times they base their judgement on experience and derive and document from past projects best practices to be considered. However, experience as a method in itself, can not be evaluated or replicated (easily); hence the need of methodological support able to allow such evaluations and replication processes. The aim of this chapter is just that - to describe a method designated for identifying design patterns in interaction design. The application of the method addresses the design of synchronous collaborative applications and it is described in this chapter together with the results derived form this application - a collection of 15 design patterns to be used in the design of applications which support synchronous collaboration.

4.1 Definition of a Design Pattern

Several different templates for defining design patterns have been proposed. A complete description of these templates is provided in Chapter 2, Section 2.2.3. They generally include the name of the design pattern, the description of the problem it addresses together with the forces that influence this problem, some examples of situations in which this problem can be met and a possible solution to tackle the problem.

For the scope of this thesis, a design pattern is defined by the tuple:

$$\mathrm{P} = (id,\ n,\ c,\ pb,\ F,\ E,\ SYM,\ CON,\ d,\ K,\ s,\ R,\ IN,\ OUT)$$

The description of these elements is presented below:

- The *identifier*, id is a string of characters that uniquely identifies a pattern. In the example 2.2, the identifier of the pattern is 2.10.
  Questions answered by this element: *What is the unique identifier of the pattern?*

- The *name*, n of the pattern is a string of characters which helps refer to the central idea of the pattern. In the example 2.2, the name of the pattern is 'TRAVEL TOGETHER'.
  Questions answered by this element: *What is the name of the pattern? How should one refer to the pattern?*

- The *context*, c is the description of the design context in which the pattern can be used. In the example 2.2, the context of the pattern is the second textual element.
  Question answered by this element: *When is the problem addressed by the pattern identified?*

- The *problem*, pb is the description of the major issue the pattern is trying to solve. It may embed textual, graphical, audio and video content. In the example 2.2, the problem is '*When finding their way through an unknown environment,*

users can often get lost'.
Questions answered by this element: *What is the recurring problem addressed?*

- The set of *forces*, F = f_1, ... , f_i is a set of the tradeoffs to be considered when applying the pattern. In the example 2.2, the forces are not described.
  Questions answered by this element: *What should one give up in the design process? What would be the benefits and the loses for eliminating or adding an element to the design process? What additional factors are involved in the problem?*

- The set of *examples*, E = e_1, ... , e_j presents the description of a set of existing applications in whose design the problem described by the pattern arose. In the example 2.3, the examples are described as Known Uses.
  Questions answered by this element: *Where has the pattern been applied?*

- The set of *symptoms*, SYM = sym_1, ... , sym_p is the set of preconditions which ask for the application of the pattern. In the example 2.2, the symptoms initiate with 'You should consider to apply the pattern when...'.
  Questions answered by this element: *What goes wrong before the identification of the problem?*

- The set of *consequences*, CON = con_1, ... , con_q is the set of benefits and liabilities resulting from the application of the pattern. In the example 2.2, the consequences are not described.
  Questions answered by this element: *What is the result of applying the pattern? What is being won? What is being lost?*

- The *diagram*, d is a graphical illustration of the pattern. No restriction on the type of the diagram are imposed, its goal being to transmit visually the main idea of the pattern. In the example 2.2, the diagram is presented at the very beginning together with the picture illustrating the place of the pattern in the

overall collection of patterns.
Questions answered by this element: *What is the visual clue associated to the pattern?*

- The set of *keywords*, K = k_1, ... , k_t is introduced to list the keywords (strings of characters) associated to the pattern, which may be either part of an existing glossary or new (with respect to the glossary) concepts related to the pattern. In this way, the keywords are the kernel for the creation of a glossary to be used for indexing and managing the pattern's description elements. In the example 2.2, no keywords are associated with the pattern.
  Questions answered by this element: *What are the words mostly associated to the pattern?*

- The *solution*, s is the description of a method or process for solving the problem addressed by the pattern. In the example 2.2, the solution is described as: '*Therefore: Browse through the information space together. Provide means for communication and collaborative browsers that show the same information at each client's side*'.
  Questions answered by this element: *How can the problem be solved?*

- The *references*, R = r_1, ... , r_u set is a set of literature references related to the pattern. In the example 2.2, no references are described for the pattern.
  Questions answered by this element: *What has been written about the pattern?*, *What similar patterns have been written by other authors?*

- The *input*, IN = in_1, ... , in_a is the set of identifiers of the design patterns which define a context for the pattern, i.e. the design situations in which the pattern can be used. In the example 2.3, related patterns are associated to the pattern described, without specifying the type of relationship existing between the patterns.

Questions answered by this element: *What are the patterns pointing to the pattern?* [1]

- The *output*, OUT = out_1, ... , out_b is the set of identifiers of the design patterns which define the design situations that refine the one in which the pattern is used. In the example 2.3, related patterns are associated to the pattern described, without specifying the type of relationship existing between the patterns.
  Questions answered by this element: *What are the patterns to which the pattern points?* [2]

The general reading template for each of the patterns is: *The pattern n addresses the problem pb which is generally met in the context c. You should consider applying the pattern when SYM. When applying the pattern you should consider F. Applying the pattern translates into applying the proposed solution s which has as consequences CON. Literature references related to the pattern are found in R and some examples of the pattern's application are E.*

4.2 Design Pattern Mining Method

Methods for identifying design patterns are scarcely described in the literature, the few examples existing being presented in Section 2.2.5. Most of the times, experienced designers write collections of patterns without any reference to the process they followed to reach them. On one hand, identifying patterns involves a great deal of experience in the area addressed by the patterns, so experience in itself can be a method for such mining processes. On the other hand, without a structured described method, the validation and the replication of such processes are hard to accomplish. Therefore, through my work, I propose a structured method for identifying design patterns in interaction design. The core idea of the method is inspired by the definition of a design pattern (”a proven solution to a recurring design problem” [22]) and by the related literature

[1]More on this in 5.
[2]More on this in 5.

in the field (see Section 2.2.5 for details).

The method consists in running a series of workshops during which teams of designers are asked to design the GUI and the interaction process of an application in the design area targeted by the pattern mining process. The design process of each team follows a set of steps and uses several creative techniques such as scenario-based design [28], free associations, sketching, and mockup creation. Each team is observed by a facilitator and the designers are encouraged to externalize the design ideas, problems, concerns, solutions they might find useful, and any issue relevant to the design of the application. These design issues form the basis of the pattern identification process. The most recurring design issues throughout the workshops point to potential candidates for design patterns in the field of the application under design.

To support the results of the workshops, a set of software applications in the area of the mining process are analysed in order to identify in what measure the design issues discussed during the workshops are considered in the implementation of existing applications. Such analysis implies getting familiar with the application, using its features, and studying its documentation in order to identify those design issues considered in each application's implementation.

The method has a two-fold focus. On one hand, it investigates issues designers address in designing applications from scratch with a clear focus on those issues mostly recurring. On the other hand, it looks at already developed and used applications for pointing out those design considerations mostly considered in their design. There are several advantages to this. First of all, the method uses several sources of data to validate its results. It does not rely only on the results designers provide and which do not go through any cycle of evaluation or only on the existing applications which do not leave much room for any additional considerations (since they are already running and used). Partial results from both the workshops and the applications analysis are correlated to lead to the final results. The overlap between the two sets of data is specifically interesting with respect to the final results. In addition to that, results obtained from the workshops and not identified during the application analysis open the door for possible innovative ideas worth being further explored.

4.2.1 Design Workshops

A design workshop provides a team of 3-5 designers (the participants) with a set of problems. These problems are chosen from the area of interest for the design pattern mining process. For example, for identifying design patterns for the design of systems to support synchronous collaboration, the problems would address domains of synchronous collaboration such as collaborative drawing or collaborative text editing (eg. design an application which would allow a group of users to draw together in the same time one drawing). Participants are asked to design the GUI and the interaction process for an application to tackle one of the problems proposed (concrete examples of this further on in Section 4.3.1.1). A design workshop brings together the team of designers and a facilitator. The role of the facilitator is to: a). describe the problems to the participants, b). walk the participants through each phase of the workshop, c). take notes of the participants' conversations, and d). observe the participants throughout the workshop and support them if needed. Each workshop lasts for approximately 2 hours and has 3 phases, adapted from the definition of a creative process as proposed by Wallas [108]. The model proposed by Wallas best fits the time length considered for a workshop and this is one of the motivation for choosing this as underlying model.

- During the first phase - the *preparation phase* - participants are encouraged to choose one problem from the set and to define as many scenarios as they can consider for software solutions (applications) to tackle the problem. In defining a scenario, they would consider answering the questions: a). who are the users?, b). what are they allowed to do through the application?, c). how could they achieve their goals using the application?, d). what is their motivation for using the application?, and e). when and where could the application be used? [28].

- The second phase - the *incubation phase* - asks the participants to choose another problem from the list and to find similarities and differences between the two problems (the one chosen during the first phase and the one chosen during the second phase). The purpose of this exercise is to identify commonalities and major differences between applications addressing different domains. Similarities would indicate the possibility of abstracting design details related to the two domains, while differences would suggest that similar design problems would require different design solutions for the two domains compared.

- Lastly, during the *illumination phase*, participants are asked to design the GUI and the interaction process of the application related to the problem they initially chose during the first phase. For that, they are strongly encouraged to sketch their ideas, express all the design problems they encounter and, possibly, create a mock up of their overall design.

4.2.2 Mining Method

A first step in the design pattern mining process is running a set of design workshops as described in Section 4.2.1. Examples of such concrete workshops are presented later on in Section 4.3.1. Throughout the design processes followed during the workshops, participants are encouraged to externalize any design problem, solution, or decision they consider relevant to their design process. In addition to that, the facilitator takes notes of their conversations. This leads to a list of design issues, a design issue being defined as any idea (problem, decision, solution, consequence, secondary effect) containing relevant information or concepts about the design of the application considered.

After each of the workshops conducted, all the design issues provided by the participants are collected. This sequence of steps (i.e. running a workshop and collecting the design issues discussed by the participants) is repeated until a fairly large (more than 100) number of different design issues are collected. Further on, for each of the design issues collected, its *degree of recurrence* with respect to the workshops (DoR_w) is computed as:

$$DoR_w(d_i) = \frac{numberOfOccurences(d_i)}{numberOfWorkshops} * 100 \tag{4.1}$$

, where *numberOfOccurrences* represents the number of workshops during which the design issue d_i has been discussed and *numberOfWorkshops* is the total number of workshops conducted. The list of design issues is sorted based on the computed DoR_ws.

As a second step in the design pattern mining, a set of existing software applications in the area of the mining process is analyzed. The analysis consists in walking through a scenario for each application and in collecting the design issues - relevant to the interaction affordances provided - considered in the application's implementation. The scenario should cover all the features provided by the application and should be tailored to each application in particular. For identifying the features provided, design documentation and/or requirements specification are used. Scenarios are written independently of the design workshops.

The goal of the software application analysis is identifying in what measure the design issues discussed during the workshops are considered in the implementation of existing applications. Moreover, the list of issues could be extended in the event that the analysis brings to light design issues not addressed throughout the workshops. For each of the design issues, its DoR_s is computed with respect to the software analysis as:

$$DoR_s(d_i) = \frac{numberOfOccurences(d_i)}{numberOfApplications} * 100 \tag{4.2}$$

, where *numberOfOccurrences* represents the number of applications in whose implementations the design issue d_i was considered and *numberOfApplications* is the total number of applications analyzed.

Candidates for being documented as design patterns are those design issues with a higher *degree of recurrence* with respect to both the workshops' results and the software analysis.

4.3 The Method Applied

4.3.1 Mining through Design Workshops

A number of 9 design workshops have been held with graduate and undergraduate students and professional designers. The list of problems they were provided with included collaborative drawing, collaborative text editing, collaborative searching, and collaborative game solving. The common requirement for all of the problems was the design of a software application to support one of the above activities in a synchronous manner. Each workshop was structured according to the description provided in Section 4.2.1 and a facilitator was present throughout each workshop, collecting all the design issues discussed by the participants. Those design issues mostly recurring were considered for further analysis.

4.3.1.1 Problems

The list of problems proposed during the workshops included collaborative drawing, collaborative text editing, collaborative searching, and collaborative game solving.

- The problem of collaborative drawing asked for the design of a software application which would allow painters, graphic designers and/or visual artists to collaboratively create one diagrammatic representation, working together in the same time.
- The problem of collaborative text editing required participants to design an application which would allow a group of users to create a summary of a written text in a synchronous collaborative fashion.
- The problem of collaborative searching required that more users are able to perform one search in the same time either from remote locations or being co-located. The problem had a concrete context of application: movie database searching. The requirements for this application asked that several users (possibly a group of friends) would be able to create through visual virtual tools a query on a database containing information about movies.

- The set of games considered for collaborative solving consisted in puzzles and crosswords. Both of these games have in common several aspects:
  - Each game can be seen as a problem with only one possible solution.
  - Reaching the solution asks for trials and decision making and marks the end of the game.

  The common requirement for both was that more users solve one game in the same time.

Overall, the motivation for choosing these four problems included the following aspects:

- As described in Chapter 3, the four activities considered are widely met in various contexts and domain.
- The identification of requirements for applications to support such activities does not necessarily require precise domain expertise. For example, defining the requirements for an application to support collaborative diagnosing requires strong medical expertise (hence the involvement of physicians). This is not necessarily the case for the activities considered throughout this work.
- All four activities are subject to synchronous collaboration.

The long-term goal of this work is to extend the results of the mining process across other activities subject to synchronous collaboration, as well. For the purpose of this thesis, however, the above mentioned activities are the target of the mining process.

4.3.1.2 Participants

The total number of participants in the workshops was 50, out of which 20% were female, and 80% were male[1]. They worked in 13 teams as follows:

[1] There could be gender differences at play in the results obtained; however, these differences are beyond the scope of this thesis to explore.

Workshop	Design	HCI	DB	TC	Male	Female	Total	%
1	4	0	0	0	1	3	4	8
2	0	0	19	0	14	15	19	38
3	0	0	4	0	4	0	4	8
4	0	0	4	0	4	0	4	8
5	0	5	0	0	3	2	5	10
6	0	0	4	0	4	0	4	8
7	0	0	0	3	3	0	3	6
8	4	0	0	4	4	0	4	8
9	0	0	0	3	3	0	3	6
Total	4	17	19	10	40	10	50	100

Table 4.1: Participants distribution across workshops

- 19 participants (38%) were Master students in a ”Multimedia databases” class (DB), so they were divided in five teams and worked simultaneously[1] on the collaborative database searching problem.
- 17 of the participants (34%) were Master students in a ”Human Computer Interaction” class (HCI). They worked in 4 teams, each team working on one of the following problems: drawing, puzzle solving, text editing, and crosswords solving.
- 10 participants (20%) were undergraduate students in a course on ”Technologies for Collaboration” (TC). They were divided into 3 teams, and they worked on the following problems: drawing, puzzle solving, and crosswords solving.
- 4 of the participants (8%) were professional designers (Design) with more than 5 years experience in graphic design. They worked as a team in designing an application for collaborative drawing.

Out of the 13 teams, 9 of them had 4 members, 3 had 3 members, and one team was constituted of 5 members. However, the slight difference between the number of team members had little (if any) impact on the collaborative design processes followed by the participants (judging by the number of design issues collected from each team).

[1] The 19 participants were divided into 5 teams, independent one of another. All the teams participated in one workshop, working in parallel.

Problem	Design	HCI	DB	TC	Male	Female	Total	%
Drawing	4	4	0	3	8	3	11	12
Searching	0	0	19	0	14	15	19	38
Text editing	0	4	0	0	4	0	4	8
Puzzle solving	0	4	0	3	7	0	7	14
Crossword solving	0	5	0	4	7	2	9	18
Total	4	17	19	10	40	10	50	100

Table 4.2: Participants distribution across problems

A more elaborate reasoning on this fact is provided further on, in subsection 4.3.1.3. Table 4.1 describes the distribution of the participants across workshops considering their background and gender.

As for the distribution of the participants across the problems addressed by the workshops (Table 4.2), 22% of them (three teams) worked on the problem of collaborative drawing, 38% (five teams) worked on the problem of collaborative searching, and 8% (one team) chose the problem of collaborative text editing. Moreover, two teams comprising 14% of the participants worked on the problem of collaborative puzzle solving and two other teams (18% of the participants) chose the problem of collaborative crossword solving.

4.3.1.3 Procedure and Results

The workshops were organized according to the description provided in Section 4.2.1, each workshop having three phases and lasting two hours. During each phase, the participants were encouraged to use different creative techniques for exploring the problem chosen. The participants were provided with postits, paper cards and other such means and they were encourage to put their thoughts down on them. After each phase, each team displayed the postits and the paper cards they wrote and discussed them with the facilitator. For sketching, they were provided with paper sheets and they were en-

Problem	Number of teams	Number of ideas
Drawing	3	45
Searching	5	72
Text editing	1	19
Puzzle solving	2	32
Crossword solving	2	47
Total	13	143

Table 4.3: Number of teams and scenarios generated for each problem

couraged to describe their ideas and results at the end of the session. Those interested in creating mockups were encouraged to use either Mockingbird or Balsamiq tools (see Section 2.3.2.3). At the end of the two-hour workshops, all such material was collected for further analysis.

Phase 1: Preparation

The first phase asked the participants to choose a problem from the list provided and to generate as many scenarios as they can consider for software solutions (applications) to tackle the problem. In defining a scenario, they would consider answering the questions: a). who are the users?, b). what are they allowed to do through the application?, c). how could they achieve their goals using the application?, d). what is their motivation for using the application?, and e).when and where could the application be used?. The total number of scenario generated was 143 (Table 4.3).

Three teams worked on the problem of designing a software application for collaborative drawing, generating a total of 45 ideas. The first team generated 26 scenarios, including: a). networks of friends come together and draw collaboratively as in playing a game, b). drawing collaboratively and projecting the drawing in different parts of the world, c). creating a city event which brings citizens together and providing them with a recording wall for drawing, d). create an online gallery and see it as a recruiting place. The second team generated 4 scenarios. Some of the ideas they generated were common to those coming from the first team. One example of common idea is allowing the application to revolve around a city event where people come together and, using different drawing techniques, draw collaboratively. Another idea proposed was seeing the overall drawing as the composition of individual drawings that each user could create in a private area of the application. Lastly, the third team proposed 15 scenarios. The recurring idea for the users' synchronous collaboration was allowing each collaborator to draw separately and compose the individual drawings into a collaboratively created drawing.

One team worked on the problem of collaborative text editing and 19 scenarios were generated. They included: a). groups of students collaboratively editing a document while discussing through an instant messaging feature, b). having a group of students take notes collaboratively during a lecture on a tablet PC, c). groups of users basing their collaboration on social features such as ranking, tagging, annotating, commenting, d). allowing each user to identify his/her contribution to the document or the contributions of others, e). allowing users to visualize the evolution of the shared resource throughout the synchronous collaborative process.

Four teams worked on the problem of the two collaborative games – puzzle solving and crosswords solving. They generated 79 ideas, including: a). the puzzle can be seen as a game or as an artistic act which brings people together, b). the puzzle can be used with medical goals such as helping elderly people in remembering things, d). allowing users to create personalized crosswords, e). supporting users in answering crossword questions in a round, each user's answer being timed.

The five teams working on the collaborative search problem generated 72 scenarios. These included: a). a group of friends trying to decide on a movie to watch, each having a criteria for their choice, b). several users remember scenes from a given movie, but can't recall the title of it, c). users from remote locations are trying to find the closest cinema to all of them.

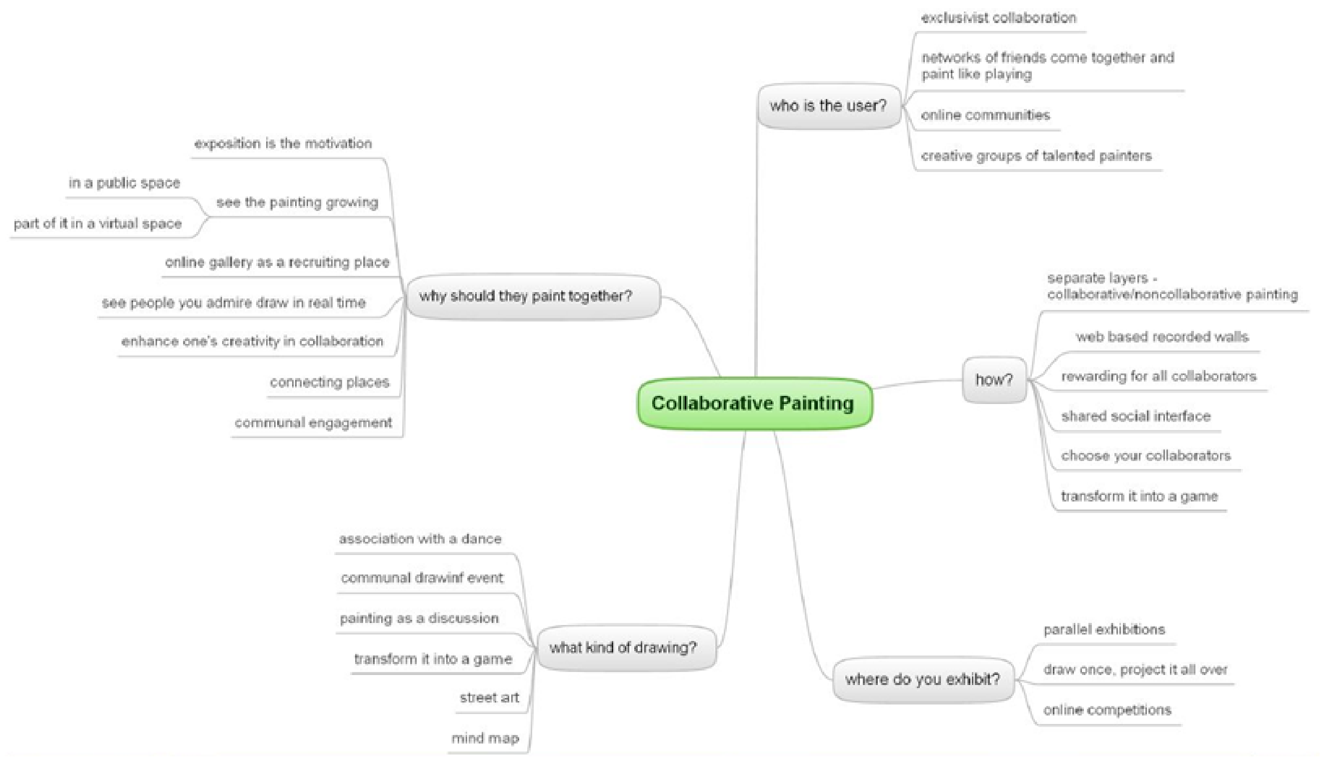

Collaborative Painting
who is the user?
exclusivist collaboration
networks of friends come together and paint like playing
online communities
creative groups of talented painters
how?
separate layers - collaborative/noncollaborative painting
web based recorded walls
rewarding for all collaborators
shared social interface
choose your collaborators
transform it into a game
where do you exhibit?
parallel exhibitions
draw once, project it all over
online competitions
what kind of drawing?
association with a dance
communal drawinf event
painting as a discussion
transform it into a game
street art
mind map
why should they paint together?
exposition is the motivation
see the painting growing
in a public space
part of it in a virtual space
online gallery as a recruiting place
see people you admire draw in real time
enhance one's creativity in collaboration
connecting places
communal engagement

Figure 4.1: Synthesis map of the scenarios collected from the first workshop

Problem	Users	Goals	Usage	Motivation	Location	Time
Drawing	10	12	9	10	4	0
Text editing	1	4	13	1	0	0
Puzzle solving	5	9	13	1	1	3
Crossword solving	4	15	20	5	2	1
Total	14%	28%	38%	12%	5%	3%

Table 4.4: Results of preparation phase - scenarios classification

A rough classification of these scenarios grouped them into issues related to users, their goals and motivation, the actual features of the application and the affordances these features provide, location, and time (Table 4.4). Moreover, the ideas expressed by each team were represented through mind maps to facilitate their further analysis. An example of such a mind map is presented in Figure 4.1.

This first phase aimed at allowing the teams to explore possible solutions for the problems they were looking into. The large number of ideas generated is an indicator of the fact that the teams found the design space exploration not only useful for finding possible solutions, but also a means of understanding the other members' backgrounds and expertise.

Phase 2: Incubation

The second phase asked the participants to choose another problem from the list and to find similarities and differences between the two problems (the one chosen during the first phase and the one chosen during the second phase). Incubation is defined as the phase during which the problem to be solved is no more conscientiously considered, but it is still running in the background of one's mind. Moving the focus from that particular problem to a similar one supports incubation in two ways. On one hand, exploring the second problem allows the identification of associations meant to help in reaching a solution for the problem initially considered (especially since both problems are in many respects similar). On the other hand, not thinking about the problem initially chosen allows the exploration of different reasoning thoughts which might provide valuable insight useful also for clarifying the initial problem.

Problem	**Similarities/Difference**
Drawing vs. Text Editing	• Similarities: – For both, parts of the document drawn/edited can be modified independently of each other. – Instant messaging features can be integrated in both types of applications. – Social features such as ranking, tagging, comments can be included in both types of applications. • Differences – Searching in an edited document is performed differently than in a drawing. – An editing document supports the creation of a table of contents.

Table 4.5 – continued from previous page

Drawing vs. Puzzle Solving	• Similarities: – Both types of applications can be used for medical purposes. – Both types of applications support team work and competitions. • Differences – There is one right solution in a puzzle game and the game imposes a set of rules. – Drawing may not follow any rules and there is no right or wrong solution to a drawing.
Crosswords vs. Puzzle Solving	• Similarities: – Both of them are games, so they support competitions among teams and may have different levels of difficulty. – Both have one final solution. – Both games can be personalized. • Differences – Puzzles work with images, while crosswords use a language, hence collaborating players must speak the same language. – Puzzles offer a preview of the solutions, while crosswords don't.

Table 4.5 – continued from previous page

Text editing vs. Crosswords Solving	• Similarities: – Both can be used with educational goals. – Both types of applications can be used on various devices such as mobile phones, laptops. – Both require a mechanism of notification of changes made to the edited document/board game. • Differences – Crosswords solving is an activity with a precisely defined end, while text editing does not have a defined end (a document may be edited at any time, while a game is over when the board is completely filled in).

Table 4.5: Results of incubation phase - Similarities and differences between two distinct problem domains

The total number of ideas generated (these including both similarities and differences) was 110, out of which 44 were similarities between 2 different problem spaces and 66 were differences. The teams explored the similarities and differences between applications addressing drawing and text editing, drawing and puzzle solving, crosswords solving and puzzle solving, text editing and crosswords. Some of their ideas are depicted in Table 4.5. After exploring possible solutions for the problem they chose, the participants shifted their focus on a different problem. This supported them in switching to a different context and creating associations between contexts. As a consequence of that, the participants were encouraged to understand the secondary implications in their design processes and to get a different perspective.

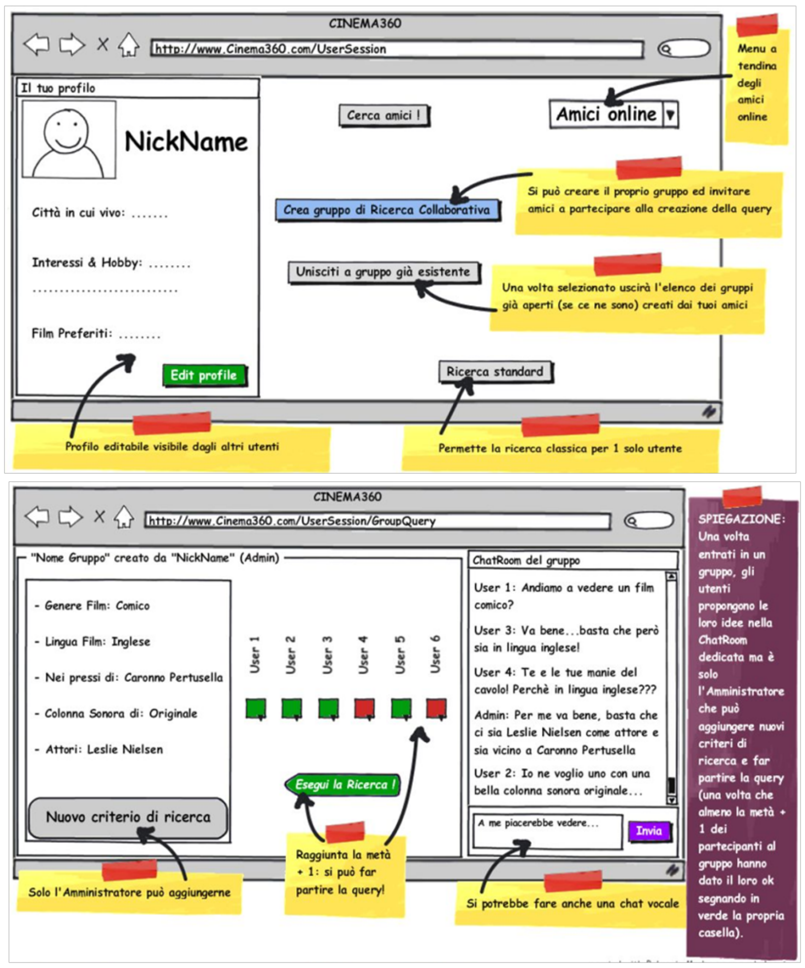
Phase 3: Illumination

During the last phase, the participants were asked to design the GUI and the interaction process of the application related to the problem they initially chose during the first phase. Their final results consisted in sketches or mockups, an example of such a design being depicted in Figure 4.2. Both the preparation phase and the incubation phase impacted this phase. On one hand, a large number of the scenarios the teams discussed during the first phase were considered for being part of their final designs. On the other hand, associations made during the incubation phase were referenced and considered in the design process. During sketching/mockup creation, the teams would constantly go back to the postits/paper cards written during the previous two phases (and displayed after each phase) to identify the issues they've discussed and to consider including them in their final results.

All the design issues discussed by the teams throughout their design processes were collected by the facilitator present during the workshops - independently from one team to another. In addition to the issues collected, the notes made by the participants on their design results were also considered for further analysis.

4.3.1.4 Design Issues Identified

The collected design issues were assigned a DoR_w (the percentage in which they were discussed throughout all the workshops) and, based on these values, they were sorted. Table 4.6 briefly describes the list of design issues discussed. All of the workshops addressed the problem of coordination, the teams providing different solutions for supporting collaborators in coordinating their work. Each solution depends on the context of the collaborative activity explored. Most of the teams considered locking as a solution. Others, looked into the possibility of allowing each group to coordinate itself or in designating one of the collaborators as the coordinator and letting him/her decide on the coordination of the others. Also, a large number of workshops addressed the issue of communication and suggested integrating a chat within the application they were designing. Similarly, participants considered adapting the applications they were


CINEMA360
http://www.Cinema360.com/UserSession
Il tuo profilo
NickName
Città in cui vivo: ....
Interessi & Hobby: ....
....
Film Preferiti: ....
Edit profile
Cerca amici !
Amici online
Crea gruppo di Ricerca Collaborativa
Unisciti a gruppo già esistente
Ricerca standard
Menu a tendina degli amici online
Si può creare il proprio gruppo ed invitare amici a partecipare alla creazione della query
Una volta selezionato uscirà l'elenco dei gruppi già aperti (se ce ne sono) creati dai tuoi amici
Profilo editabile visibile dagli altri utenti
Permette la ricerca classica per 1 solo utente
CINEMA360
http://www.Cinema360.com/UserSession/GroupQuery
"Nome Gruppo" creato da "NickName" (Admin)
- Genere Film: Comico
- Lingua Film: Inglese
- Nei pressi di: Caronno Pertusella
- Colonna Sonora di: Originale
- Attori: Leslie Nielsen
User 1
User 2
User 3
User 4
User 5
User 6
Esegui la Ricerca !
Nuovo criterio di ricerca
ChatRoom del gruppo
A me piacerebbe vedere...
Invia
Solo l'Amministratore può aggiungerne
Raggiunta la metà + 1: si può far partire la query!
Si potrebbe fare anche una chat vocale

User 1: Andiamo a vedere un film comico?

User 3: Va bene...basta che però sia in lingua inglese!

User 4: Te e le tue manie del cavolo! Perchè in lingua inglese???

Admin: Per me va bene, basta che ci sia Leslie Nielsen come attore e sia vicino a Caronno Pertusella

User 2: Io ne voglio uno con una bella colonna sonora originale...

SPIEGAZIONE: Una volta entrati in un gruppo, gli utenti propongono le loro idee nella ChatRoom dedicata ma è solo l'Amministratore che può aggiungere nuovi criteri di ricerca e far partire la query (una volta che almeno la metà + 1 dei partecipanti al gruppo hanno dato il loro ok segnando in verde la propria casella).

Figure 4.2: Partial results of the illumination phase - Mockup for a collaborative search application

designing to several devices, some of them providing different interfaces for different types of devices considered (i.e. phones, laptops). Social features were considered by most of the teams, as well as supporting the awareness of the collaborating users with respect to each other's actions on the shared resource.

ID	Design Issue	DoR_w
14	How to support the coordination of a collaborative process in order to ensure that all collaborators participate in the process and that the resource remains consistent at all times?	100
14.1	— use timers so that each collaborator gets access to the resource for a given amount of time	7.69
14.2	— use separate, independent blocks for each collaborator so that there is no dependency among contributions coming from different members of the same team	15.38
14.3	— link the application to a community and let this community decide how to coordinate itself	23.07
14.5	— support the creation and execution of workflows	7.69
14.6	— any time a collaborator starts editing the shared resource or part of it, the resource or that specific part are locked until they are saved	23.07
14.8	— one of the collaborators (for example, the creator of the resource) is the coordinator of the entire process, being the decision maker	23.07
6	Integrate communication tools within the application - instant messaging, chat	76.92
2	Interaction through Web based recorded walls; Web based collaboration	69.23

Table 4.6 – continued from previous page

22	Adapt the application to the several devices	69.23
22.3	— target mobile devices	15.38
22.1	— use one shared device such as a tabletop	7.69
22.2	— use different inter-connected devices	7.69
22.5	— have a script for identifying automatically the device and adapt the application to that device	7.69
1	Provide separate layers for collaborative and non collaborative activities	53.84
4	Support the users in choosing their collaborators	46.15
7	Allow each collaborator to visualize what the others are doing in real-time	46.15
9	Support collaborators in tagging, ranking, commenting the shared resource they are working on	46.15
15	Usability - users decide how they will use the application	38.46
15.1	— allow the users to appropriate the application to their needs, provide flexibility	15.38
15.2	— multi-language application	15.38
18	Support the competitions among different teams	38.46
17	Use different media for transmitting/capturing information (various input, output channels)	30.76
28	Support the creation of groups of collaboration; users can join such groups, leave them or create new ones	30.76

Table 4.6 – continued from previous page

5	Transform the collaborative process into a game; use it for entertainment	23.07
33	Adapt the application to the participants; allow them to decide on the parameters of their collaborative work so that they can customize their collaboration	23.07
11	Design the application as a teaching application; embed educational goals	23.07
45	Envision the collaborative process as a city event - citizens' collaboration	23.07
52	Provide ways for the application to support its users through feedback on their actions and corrections	23.07
30	Support collaborators in visualizing who else is connected to the application at a given time	23.07
16	Collaboration is the composition of individual contributions created in a non-collaborative manner; divide the overall work in blocks	23.07
20	Registered users have additional privileges	15.38
43	Create mind maps of the collaborative process	15.38
44	Design the collaboration as a street art event	15.38

Table 4.6 – continued from previous page

10	Provide tools for searching/filtering data related to the collaborative process	15.38
24	Track and support the visualization of the history of the collaboration through timelines or log files	15.38
54	Include translation features	15.38
55	Include help features (such as tooltips)	15.38
58	Each user may manage his/her own profile, which would include information relevant to the collaborative process targeted by the application	15.38
59	Link the application to other applications such as calendars or maps, depending on the type of collaborative process the application supports	15.38
24	Track one individual's contribution	15.38
24.1	— use colors for visualizing individual contributions; each user is associated with a color	7.69
24.2	— click on the name of the user and visualize only his/her contribution to the shared resource, all other contributions being faded	7.69
3	Provide means for rewarding all collaborators; provide direct benefits for all the users working together	7.69
42	Visualize the collaboration process on walls displayed remotely on different sites	7.69

Table 4.6 – continued from previous page

46	See people you admire draw in real time	7.69
47	Build the application as a recruiting place	7.69
48	Bring people with different expertise together and allow then to use different tools and work together	7.69
49	Connect people	7.69
12	Gather social data from the collaborative process to support further analysis of the collaborators' interactions	7.69
50	Use layers	7.69
13	What is someone is making trouble? How to apply censorship?	7.69
51	Evolve the application as a perpetual beta; support prototyping, and different versions	7.69
53	Address elderly people	7.69
19	The application is used with medical goals, in various contexts	7.69
56	Create physical places for using the application	7.69
21	Integrate the application with social network services	7.69

Table 4.6 – continued from previous page

57	Support the collaborators in asking for suggestions from the application	7.69

Table 4.6: Design issues collected from the workshops

The issues discussed by the teams did not always address synchronous collaboration in particular. On one hand, there is a fuzzy line between the synchronous and asynchronous mode of a collaborative process, features supporting mostly asynchronous collaboration providing benefits to collaborative processes held synchronously. For example, tracking the history of a collaborative process would prove beneficial in asynchronous contexts, when the collaborators would need to go back to the results of their collaboration long after the collaborative process ended. However, being provided with ways to visualize in real-time the evolution of a synchronous collaborative process might as well support those collaborators involved in the process. On the other hand, tools which support synchronous collaboration might be suited for asynchronous contexts as well, depending on the time dimension in which they are being used. Therefore, the requirements of these types of applications would have several similarities.

4.3.2 Mining through Synchronous Applications Analysis

The applications analyzed are those described in Chapter 3 of this thesis (for a brief reminder, see Table 4.7). Each application was associated with a scenario and each scenario was further on walked through, the purpose of the scenarios being to support the facilitator in identifying those design issues considered in the implementation of each application.

For each design issue, its DoR_s was computed as the percentage in which the issue was considered across the designs of the applications analysed. Table 4.8 lists down these issues together with their DoR_ss. A large number of the applications considered addressed the issue of coordination, several solutions to this being used. Locking was

Application	Domain	Application	Domain
Synergo 3.1.1	drawing	Coagmento 3.2.2	searching
NetDraw 3.1.2	drawing	SearchTogether 3.2.3	searching
DeTransDraw 3.1.5	drawing	Cerchiamo 3.2.4	searching
CO2DE 3.1.3	drawing	VisSearch 3.2.5	searching
LucidChart 3.1.4	drawing	AntWorld 3.2.6	searching
TellTable 3.3.1	text editing	WeSearch 3.2.7	searching
GoogleDocs 3.3.4	text editing	M@M 3.4.1	game solving
CodoxWord 3.3.2	text editing	CPG 3.4.2	game solving
EtherPad 3.3.3	text editing	Sides 3.4.3	game solving
CoSearch 3.2.1	searching	STARS 3.4.4	game solving

Table 4.7: Synchronous applications analysed

popular, as well as embedding the possibility for the collaborating group to coordinate itself. Tracking the history of the collaborative process supported by each tool was one of the most frequent concerns considered. In addition to that, issues such as supporting communication through integrated chat mechanism or supporting awareness among collaborating users were considered by half of the applications considered. Social features and device adaptation were also included in some of the applications analysed.

ID	**Design Issue**	DoR_s
14	How to support the coordination of a collaborative process in order to ensure that all collaborators participate in the process and that the resource remains consistent at all times?	75
14.3	— link the application to a community and let this community decide how to coordinate itself	50
14.4	— voting, agreement rate	5

Table 4.8 – continued from previous page

14.7	— any time a collaborator starts editing the shared resource or part of it, the resource or that specific part are locked until they are saved	15
14.8	— one of the collaborators (for example, the creator of the resource) is the coordinator of the entire process, being the decision maker	5
23	Track and support the visualization of the history of the collaboration through timelines or log files	60
23.2	— support versioning	15
23.2	— support reverting changes	20
23.4	— history is interactive; click on search result, redirects to page	10
23.5	— timelines	5
6	Integrate communication tools within the application - instant messaging, chat	50
7	Allow each collaborator to visualize what the others are doing in real-time	50
7.1	— the server echoes instantly drawing and editing actions	25
9	Support collaborators in tagging, ranking, commenting the shared resource they are working on	45
2	Interaction through Web based recorded walls; Web based collaboration	40
22	Adapt the application to the several devices	35
22.1	— use one shared device such as a tabletop	10
22.6	— tabletop	15

Table 4.8 – continued from previous page

22.3	— target mobile devices	10
22.7	— handheld computers that support Java	5
24	Track one individual's contribution	30
24.1	— use colors for visualizing individual contributions; each user is associated with a color	15
4	Support the users in choosing their collaborators though invites (e-mails or shared links)	20
5	Transform the collaborative process into a game; use it for entertainment	20
27	Annotate the objects subject to collaboration	20
33	Adapt the application to the participants; allow them to decide on the parameters of their collaborative work so that they can customize their collaboration	20
33.1	— assign roles and rights to collaborators	10
1	Provide separate layers for collaborative and non collaborative activities	15
17	Use different media for transmitting/capturing information (various input, output channels)	15
17.1	— visual and audio cues on what the others are doing	5
25	Import/export facility to other formats	15
25.1	— publish doc as a web site/wiki	5
25.2	— pdf, jpg	10

Table 4.8 – continued from previous page

8	Include playback tools based on log files	15
8.1	— records snapshots of the drawing	5
38	Includes a recommendation mechanism	15
28	Collaborators can leave and join collaboration at all times	15
28.1	— notifications are sent on their status	5
19	The application is used with medical goals, in various contexts	10
30	Support collaborators in visualizing who else is connected to the application at a given time. Display a list with all available collaborators	10
31	Provide templates in the community library	10
34	Identify each collaborator with a name/color	10
36	Each searched page is associated with metadata	10
39	Provide a shared summary of the collaboration	10
39.1	— summary of the findings of the search	10
26	Communicate with other similar tools	10
26.1	— share docs online	15
26.1	— AutoCAD via file transfer	5

Table 4.8 – continued from previous page

11	Design the application as a teaching application; embed educational goals	5
18	Support the competitions among different teams	5
20	Registered users have additional privileges	5
21	Integrate the application with social network services	5
21.1	— share doc on social networks	5
29	Uses colors to indicate an object status (locked, available for editing)	5
32	Conflict resolution strategies	5
35	Associate a query history to a search topic	5
37	Includes mechanisms for division of labor	5
40	Provide similar docs of interest to other collaborators	5
41	Store one session's state and resume it later	5

Table 4.8: Design issues collected from analysing the collection of application in Chapter 3

At a closer look, most of the issues collected through the two phases are the same. However, several issues relevant to designing synchronous collaborative systems have been collected only from one of the phases. As example, several applications from those analysed considered supporting each collaborating in identifying his/her individual contribution to the collaborative process. The workshops did not point this as a recurring issue. Supporting collaborative undo processes, allowing collaborators to go back to previous steps of their collaboration is yet another example of issue that was brought to light by the applications analysis but not considered throughout the workshops.

4.4 The Patterns Identified

Starting from the identified recurring issues, I moved on to writing the patterns documenting them. In this process, I made use of several pattern writing tutorials, including Meszaros's paper "A pattern language for pattern writing" [75]. I started by framing the problem addressed by each issue and then I associated it with the solution proposed for it. Once the patterns written, I had them discussed at both EuroPLoP2011[1] and AsianPLoP2011[2] conferences, rewriting them based on the feedback received from the pattern community. The patterns went through two reviewing rounds before being workshop-ed (i.e. face-to-face peer reviewed) in the two conferences. In addition to that, the identified patterns are validated by similar documented issues which I will discuss in the "References" section of each of the patterns presented below[3].

4.4.1 Who is the coordinator?

Context. A group of collaborators work together on the same shared resource (i.e. a shared drawing, a text document, a game board). Each of them contributes to the creation and the evolution of the resource, making sure that together they reach a version of the resource agreed by all those involved in the process.

[1]http://www.hillside.net/europlop/europlop2011/

[2]http://patterns-wg.fuka.info.waseda.ac.jp/asianplop

[3]As a general comment, the following pattern descriptions will exclude the identifier, the keywords, the input and the output defining elements, since Chapter 5 provides an ample discussion on them and the way they are used and inferred.

Problem. If more users work on the same resource in the same time, there needs to exist a coordination mechanism which: a). allows all collaborators to take part in the collaborations and b). maintains the resource in a consistent state at all times. The problem is how to determine who coordinates the collaborative process or what is the suited coordination mechanism for each concrete case.

Symptoms. You should consider applying the pattern when:

- Groups of users work together in the same time on one shared resource. As a general remark, most of the synchronous collaborative applications require the application of this pattern.
- Parallel editing operations are a threat to the consistency of the shared resource.

Forces. When applying the pattern you should consider:

- It may be that the application addresses groups which act as communities with unwritten rules. Such a situation would not ask for an enforced coordination mechanism for the system.
- Some cases ask for a (small) group of users to come to an agreement before specific actions are executed on the shared resource. Such an example is collaborative searching where a group of users should be able to create a query using visual virtual tools and the collaborators may change their contribution to the query at any time. The coordination issue at this point is to make sure that the query gets executed only when all collaborators agree on its content.
- The access to the shared resource and the changes to it may be subject to timing. For example, in particular cases such as games competitions could be supported.

Solution. Therefore: Identify the context of the application and address the coordination issue according to the following considerations:

- If the application addresses a well formed community with unwritten rules, then link the application to the community and to the way the community as a whole coordinates itself. As example, a group using the crosswords solving application may decide that a participant gives the control to another participant at his/her first wrong answer.

- If the context requires one user to initiate the collaboration, then that user might be the coordinator of the whole process. In the case of collaborative search, the user who initiated the query may be in charge of deciding when the query gets executed.

- Locking is a solution for coordination in cases in which time is not necessary an issue and where the common resource supports this operation. Text editing may be subject to this solution by allowing the user who starts editing the document to lock it until s/he saves her/his contribution to the document. It is only after the lock is released that some other collaborator can contribute to the editing of the resource.

- Timers support coordination by allowing each participant to the process to gain control of the resource for a limited time. Games are an example of applications where this coordination option would fit.

- Having separate blocks for each collaborator may also be a solution in cases in which the overall activity does not require to conform to some standards of definition. An example of such a case would be drawing as an artistic activity, where each participant to the collaboration may be in charge of one area of the common display.

References. Patterns addressing coordination in computer mediated collaboration have been written also by Lukosch and Schummer in their book ”Patterns for Computer Mediated Interaction” [94].

- *Floor control* is such an example and it specifically addresses synchronous interaction which can ”lead to parallel and conflicting actions that confuse the interacting users”, making interaction difficult. The solution proposed by the pattern suggests allowing only one user at a time act on the shared resource.

The right to interact with the shared resource is passed among all the interacting users.

- *Vote* is a pattern addressing the issue of quickly testing a group's agreement on a specific issue. The solution proposed by the aforementioned authors is providing an easy way of setting up and running a poll within the application.
- Locking is addressed in *Pessimistic locking*. Locks may be requested and received and it is only after receiving them, one is allowed to modify the state of a shared resource.

Examples. CoSearch is an example of application which allows the initiator of a search query to act as a coordinator, being allowed to decide when the query gets executed. Also, as examples of applications which implement locking as a coordination mechanism consider CO2DE and TellTable.

4.4.2 Integrated chat

Context. A group of collaborators share a common resource which is the subject of their collaboration. However, they work from different locations, without being able to meet and discuss about their collaborative process.

Problem. Since communication is one of the main aspects of any collaborative process, collaborators should be able to exchange messages related to their collaboration, share knowledge based on each individual's expertise, and clarify any additional misunderstandings.

Symptoms. You should consider applying the pattern when:

- The collaborative activity requires collaborators to exchange messages.
- There is no other communication mechanism considered and embedded in the application.

Forces. When applying the pattern you should consider:

- Real-time communication allows collaborators with different expertise to share and clarify any misunderstandings that might come up throughout their process.
- Real-time communication can be used as a coordination tool for those applications which do not embed any coordination mechanism, but allow the community as a whole to decide on a way to coordinate itself.

Solution. Therefore: Integrate an instant messaging feature in the design of the application. In doing that, either link the application to an existing real-time communication application or embed a chat feature within the application. In the cases which allow it, the application could consist in a mash-up between a chat feature and a collaborative activity feature. An example of such a case is the collaborative text editing application, which could be a mash-up between a chat feature and a real-time document editor.

Consequences. Applying the pattern leads to:

- Supporting collaborators in communicating without the burden of switching to another tool when in need to exchange messages.
- Allowing the documentation of the collaborative process by means of a message exchange log.
- Integrating a real-time communication feature in a collaborative application might disturb collaborators from their main collaborative tasks. It is for this reason that the location of a chat feature in the user interface of a collaborative interface should be peripheral, possibly hidden.

References. Lukosch and Schummer wrote the pattern *Embedded chat* [94]. The two are addressing the same problem, proposing the same solution - the integration of a chat mechanism in the collaborative application.

Examples. Examples of applications which integrate a chat feature in their functionality are: NetDraw, CO2DE, LucidChart, EtherPad, SearchTogether, Stars, Synergo, GoogleDocs, Coagmento, Mystery at the Museum. A more detailed description of these applications is presented in Chapter 3.

4.4.3 Eyes wide open

Context. Groups of collaborators are working from geographically remote locations. They need to be aware of the others' activity on the shared resource so that they can contribute to it accordingly.

Problem. Synchronous collaboration asks for all the collaborators to be aware at all times of the evolution of the collaboration. For that, each collaborator must be able to visualize what the others are contributing to the process at any time. In addition to that, each contribution should be made visible to all the collaborators in real-time and possibly made explicit.

Symptoms. You should consider applying the pattern when:

- Updating changes on the shared resource is mandatory for all synchronous collaborative processes.
- Notifying the changes performed on the shared resource is specially important in cases when collaborators are not familiar with each other or when real-time communication is not available.

Forces. When applying the pattern you should consider:

- Some collaborative activities such as drawing or text editing would be highly interrupted by notifications of all updates on the shared resource. Hence, a more selective notification process is needed and needs to be decided on.
- On the other hand, in cases like games it is of major importance that all the collaborators are aware of the others' actions on the shared board. Moreover, in

cases where time is an issue, identifying updates on the board should be fast, and straightforward.

Solution. Therefore: Update any changes on the commonly shared resource (drawing canvas, text area, puzzle/crosswords board) in the collaboration and notify (in real-time) all collaborators of these updates. The choice of notifications would depend on the context of the application:

- An update of the shared resource without any notification would suffice in cases in which the collaborative process would get disturbed by an abundance of notifications. An example would be the collaborative drawing where updates to the canvas would not require notifications and would disturb the collaborators.
- Mail notifications are helpful in the cases in which collaborators would need to keep a track of the collaboration and go back to any step of it after the synchronous process ends.
- Pop-up notifications would suit applications where a). the updates cannot be easily spotted or/and b). the overall collaboration highly depends on the awareness of each collaborator. As example, it might not be straightforward noticing the addition of one piece in a collaboratively solved puzzle. On the other hand, for a game application where collaborators' participation is sequential it is highly important that each collaborator is notified of the changes his/her peers have made along the process.

Consequences. Applying the pattern leads to:

- Collaborators are informed of each others actions, being supported in coordinating themselves as a group.
- Using a notification mechanism not suited for the type of collaborative activity may bring more disadvantages than advantages to the entire process.

References. Awareness and notification techniques are the interest of some of the patterns proposed by Lukosch and Schummer [94]:

- *Remote field of vision* is a pattern addressing the problem of multiple users working simultaneously with a shared text editor. The intent of the solution proposed is to "explicitly indicate the location and scope of each user's view [of the shared document] to every other user". This pattern does not specifically address the awareness of the collaborators with respect to the changes each other make on the shared resource, but with their location in the shared document edited.

- Awareness with respect to the selection of objects located in a shared space is discussed in *Remote selection*. The pattern suggests notifying all collaborators that a specific shared objects has been selected by one of them. No discussion on the type of notification is proposed, however. Moreover, the nature of the shared objects is not specified.

- Still on the topic of collaborative text editing, the pattern *Remote cursor* addresses the awareness issue with respect to the user's interaction with the application (such as the mouse or cursor movements). The solution proposed by the pattern is showing the "text cursor of remote users on a local user's view of the shared editor".

- *Activity indicator* pattern points to the context in which users would require some time to perform an action before that action is made visible to the others. For example, sending a message in an instant messaging application requires some time for the message to be typed by the sender. At the other, the receiver should be notifying about the fact that the message is being typed in. The message exchange example is quite straightforward and it basically resumes the idea of the whole pattern, i.e. "provide an indication of other user's activities while not showing the activity's intermediate results".

- *Change indicator* discusses the issue of indicating that a shared artifact has been changed. The information related to such a change should also include details about the type of change and a link to the new version of the artifact.

Examples. EtherPad automatically updates all the edits on the shared document on all the views of the document without any explicit notification. In CPG (Collaborative Puzzle Game), notifications of changes on the shared puzzle are sent through both visual and auditory media. Sounds are played when a user places a correct piece in the puzzle.

4.4.4 Choose your collaborators

Context. Co-workers get connected to the synchronous collaborative application and they are interested in finding and working with their peers.

Problem. In order to start a synchronous collaborative process, users must meet, either in real spaces or in the virtual. Users should be provided with the option of getting together and collaborating with their own peers.

Symptoms. You should consider applying the pattern when:

- It is expected that members of the same team will use the application.
- The application is game-based or used for educational purposes.
- The effectiveness of the collaboration depends on collaborators knowing each other.

Forces. When applying the pattern you should consider:

- In game applications, users might want to challenge each other and start competitions among collaborative teams.
- In some contexts, it is necessary that people who know each other collaborate, so they need a way to find their collaborators and form a team.
- Each user might need to know who the users available for collaboration are. Moreover, the application needs to embed a feature which would support users in inviting their peers to collaborate.

Solution. Therefore: Allow each user to choose his/her collaborators as follows:

- Provide a list with all the users currently available.
- Allow a user to search for his/her peers in the list of available users.
- Allow users to invite each other to collaborate by creating a group. In the case of games, one user may challenge others to join a collaborative game.
- Allow each user to join a group already created after s/he logs in to the application.

Consequences. Applying the pattern leads to:

- Supporting community building by allowing users to form and manage groups.
- Allowing users to be aware of the presence of other peers.

References. Several variations and specializations of this pattern are described in [94].

- The pattern *Group* is perhaps one of the basic ideas to start with, i.e. "allow users to manage groups and interact with a group in the same way in which they would interact with a user".
- Closely related to the above, there is the pattern *Bell*, whose basic intent is to inform a group's members already engaged in a collaborative session that a user wants to join that session.
- *Invitation* pattern addresses the issue of one user wanting to specifically collaborate with another one/others. The solution proposed by the pattern is sending and tracking invitations from one user to another/others.
- Ways to provide information about the users participating in a collaborative session are described by the pattern *User list*. They include showing who is currently participating in a session, showing who is currently accessing an artifact, and ensuring that the information is always valid.

Examples. LucidChart provides each collaborator with a list of all the other available collaborators. Other tools with similar features are EtherPad, GoogleDocs, Coagmento.

4.4.5 Collaboration, always social

Context. Groups of collaborators are working together. They are not a well-formed community and they need support in building a trust level within their group.

Problem. Collaboration is, more than anything else, a social process. Collaborators need to be supported in meeting and sharing feedback on the shared resource with each other.

Symptoms. You should consider applying the pattern when:

- The target users do not know each other.
- Collaborators are working on a large number of shared resources.
- The collaborative process supports competitions (eg. games).

Forces. When applying the pattern you should consider:

- Collaboration triggers the formation of communities with common interests and/or common goals. Trust becomes an important issue in such a context, hence supporting communities through social tools enhances their collaboration.

Solution. Therefore: Integrate mechanisms of tagging, ranking, and commenting in the application, as follows:

- Tagging supports the assignment of a label to a resource. This operation supports searching and identifying resources with common characteristics.

- Ranking allows the creation of value scales based on which the resources can be ordered. In this way, one could easily identify the resources ranked higher by his/her community or the group s/he is part of.
- Comments allow collaborators to give feedback on the collaborative process or on the shared resource.

Consequences. Applying the pattern leads to:

- Tagging supports the tracking and retrieving of shared objects.
- Ranking brings a sense of awareness with respect to how a group comparatively evaluates shared objects.
- Comments support knowledge exchange, and communication among the members of a collaborating group.

References. An ample discussion on patterns for the design of social interfaces is presented in [30]. However, not all the patterns written for the area of social interfaces design would prove useful for collaborative applications design. Lukosch and Schummer address some of them in their book, [94].

- *Letter of recommendation* provides solutions for allowing users rate each other on their expertise. Such solutions include providing a method of rating interactions and making these ratings available for all the users of the application.
- *Birds of a feather* suggests supporting users with similar profiles or interaction histories in finding and collaborating with each other.
- Tagging important artifacts using flags for a better retrieval is the issue addressed by the pattern *Flag*.

Examples. Social features are included in tools such as AntWorld, Mystery at the Museum.

4.4.6 My contribution

Context. One shared resource is being edited by a group of collaborators, changes being performed by all the members of the group.

Problem. It is often the case that one might need to know what a particular collaborator has contributed or what s/he her/himself has added to the collaborative process. Users should have available a straightforward and user friendly way to track their own contribution to the collaborative process.

Symptoms. You should consider applying the pattern when:

- Reports on individual contributions are required at some point during the collaborative process.
- Individual contributions must be tracked for further analysis or ranking.

Forces. When applying the pattern you should consider:

- Identifying each collaborators contribution supports quantitative assessments of the collaborative process as well as competitions among teams.
- Collaborative efforts are often the composition of individual non-conflicting contributions towards a common goal. It is therefore useful to allow the identification of these individual contributions to the overall process.

Solution. Therefore: Support each collaborator in tracking down his/her contribution to the collaboration, as follows:

- For the cases where the shared resource is textual (text editing, crosswords solving) and where the group of collaborators is relatively small, assigning different colours (hence, defining a colour scheme for the application) to each collaborator is a solution. In this way, each user's contribution is highlighted in a different colour.

- The applications for which the shared resource is an image may highlight one's contribution by representing (at one's request) only those shapes (in cases such as drawing) or pieces (in cases like puzzle solving) added by a particular user.
- A more intrusive solution to this problem would provide each user with the possibility of dragging the mouse over parts of the shared resource and, as answer to that, visualize (locally) tooltips containing information on the author of that particular part.

Consequences. Applying the pattern leads to:

- A possible overloaded interface (too many colours, too many signs).
- The need for the users to learn and recognize the identifiers of each other's contribution.
- Supporting awareness by allowing each user to know what others and him/herself have added to the process.

References. None of the patterns available in the literature address specifically this problem.

Examples. In text editing applications such as EtherPad and CodoxWord, each user is associated with a colour and the text edited by a specific user is highlighted in his/her colour.

4.4.7 Track history of collaboration

Context. Groups of collaborators follow a collaborative process with a particular final goal in mind. After the synchronous collaboration ends, they might want to replay the process or gather specific data with respect to it.

Problem. Synchronous collaboration processes are being held in real-time, so it could be the case that a lot of the information on the dynamics of the collaborative group and on the knowledge exchanged is lost. Hence, providing a way of tracking the history of the collaboration supports: a). replaying the process, b). gathering social data relevant to the collaboration, and c). learning processes.

Symptoms. You should consider applying the pattern when:

- The collaborative process needs to be replayed after it has ended.
- Additional data captured by the process needs to be collected and analysed later on.
- The evolution of the shared resource needs to be tracked step by step.

Forces. When applying the pattern you should consider:

- One benefit of collaborative processes is that they allow collaborators to learn from each other. Such learning processes can be supported by going through the actions performed and messaged exchanged during the collaboration and analyzing them.
- It is often important to go back to previous steps of a process, hence to have access to previous versions of the shared resource. Also, changes made on the shared resource might need to be undone.

Solution. Therefore: Track the history of the collaboration and make it available either through repositories, log files or timelines.

- Repositories are a useful solution for tracking the history of collaborative processes and for having a versioning system of the resources shared.
- Log files offer the possibility of rewinding and replaying the process. They are written in a standard format decided with respect to the context of the application, and they contain information on the actions performed and the messages exchanged by the collaborators.

- Timelines provide a helpful visual tool for tracking the collaboration process.

Consequences. Applying the pattern leads to:

- Ensuring collaborators no data is lost along the way.
- Supporting learning processes on the basis of replaying ended ones.

References. Saving the history of a collaborative process is the focus of the following patterns described in [94]:

- *Activity log* suggests storing all users' activities on the shared artifacts in a log. Access to the log should be provided to all users.
- The idea of a timeline is pointed out in the *Timeline* pattern in the context of long-term asynchronous and synchronous interaction. This makes it easy understanding who has been active at a specific point in time.

Examples. LucidChart includes a revision history feature which supports reverting changes, or starting a new document from a previous version of an existing one. EtherPad provides the collaborators with the possibility of saving the revisions made on a document, so that the history of the collaborative process is tracked. Lastly, SearchTogether not only stores the entire history of a collaborative search session, but it also makes this history interactive, allowing each user to click on any of the query terms in order to view the results it produced.

4.4.8 With or without collaboration

Context. Collaborating users share a public area of the application.

Problem. Users might need, at times, to sketch their ideas before adding them to the area visible to all collaborators. They need tools to support them in externalizing and evaluating their ideas before sharing them with their collaborators. Also, it might be the case that users need to try out solutions without interfering with the others' actions or without blocking the collaborative process.

Symptoms. You should consider applying the pattern when:

- Making a contribution to the shared resource requires more trials and time.
- Collaborators prefer to keep certain contributions private.

Forces. When applying the pattern you should consider:

- One coordination mechanism used in collaborative systems would lock the shared resource as long as one collaborator edits it. However, synchronous processes ask for multiple collaborators to work together in real-time. The resource being locked by one user leads to the situation in which the other collaborators are denied any contributions.
- On the other hand, some might feel more comfortable using an application in a non-collaborative way unless they find real benefits in the collaborative process.

Solution. Therefore: Provide users with an additional private area, not available to the other collaborators. Inside this area, each collaborator has total control and s/he is provided with tools specific to the context of the application. For example, in the cases of applications where sketching plays a major role, the private area of the application should provide the user with sketching tools.

Consequences. Applying the pattern leads to:

- Supporting externalization processes.
- Also targeting users not necessarily interested in using the application as collaborative.

References. Close to the idea of the current pattern is the *Masquerade* pattern which supports controlling how much private information one reveals to other users "when interacting in a collaborative environment"[94]. However, *Masquerade* only relates to the user's profile and to the control of the information contained within it. It does not refer to the users' activity and to the way it should be controlled prior or during being shared.

Examples. Coagmento supports both individual and collaborative search session. In GoogleDocs, one can sketch individual contributions in a private document prior ro making them public to his/her collaborators through a shared document.

4.4.9 Annotate

Context. Groups of collaborators share a common resource and work from different locations. Their collaboration heavily relies on the resource they share and work on, synchronously.

Problem. Synchronous remote collaboration could bring a series of misunderstandings among the collaborators. The issue raised is how to support the collaborators' common understanding and their reasoning on the shared resource.

Symptoms. You should consider applying the pattern when:

- A group of users is working on a shared resource in real-time from remote locations.
- Users have different expertise and a chat feature might not be enough for them to reach a common understanding of the issue, subject to their collaboration.

Forces. When applying the pattern you should consider:

- Common understanding is highly important in collaborative processes. It improves the overall productivity of the collaborative group. However, collaborators may have different backgrounds and expertise, hence misunderstandings and communication gaps can occur at any time.

- Having as support the object of the misunderstanding and being able to point to it and associate it with the description of the issue of concern supports communication among collaborators. Moreover, different domains use different channels of communication (audio, textual, video). It is for that reason that users should be provided with various channels for expressing themselves.

Solution. Therefore: Allow the collaborators to annotate the shared resource they are collaboratively creating. Annotations may be textual, audio or video material. Assigning an annotation to the resource or to parts of the resource signals a possible open issue or misunderstanding coming from one of the collaborators. For any annotation, it should be possible that the other collaborators answer, using textual, audio or video channels. Design explicit controls which allow users to:

- Select the area they want to annotate using either a mouse, keyboard, or gestures (for touch screens). Also, provide users with the option of selecting the entire content of the shared resource.

- Create an annotation associated to the selected area by either typing text, or uploading audio or video material in an designated area of the GUI.

- Associate an annotated area with a visual link to the actual annotation. By clicking the link, the user is shown the content of the annotation.

- Once visualizing an annotation, any user may add to it, by either commenting as answer to the issue raised or uploading additional material to the annotation.

Consequences. Applying the pattern leads to:

- A more efficient communication among the collaborating users.

- A rough way of documenting the collaborating process.

- A fast growth of the number and content size of the annotations on a shared resource might signal major insufficiencies in the resource.

References. Annotations have been discussed in various pattern collections, such as [30] or [94]. As examples, consider the following:

- The issue of sharing comments on specific content is addressed in the pattern *Shared annotation*. Users should be able to enter such comments on specific parts of the shared resource and their comments should be displayed together with the actual content of the resource.

- The structure of an annotation is further discussed in *Threaded discussions*. Although this pattern's authors defined it for the context of a textual communication channel such as a forum, a threaded discussion may constitute the structure of an annotation as well. This would allow collaborators to answer each others comments on specific parts of the shared document.

Examples. In NetDraw, any collaborator can link the object collaboratively drawn with descriptive text that others could read and add to.

4.4.10 Collaborative undo

Context. Collaborators working on the same shared resource in real-time agree to give up some of the changes performed on the resource. They identify redundancies in the document or discover error in the editing process.

Problem. Negotiation is common to collaborative processes. As result of negotiation processes, the collaborators may decide to renounce (partially or totally) to specific contributions. They need a mechanism that allows them to identify the changes they want to give up and to obtain a stable version of the shared resource for continuing their collaboration.

Symptoms. You should consider applying the pattern when:

- Users are prone to undo modifications made collaboratively.
- The collaborative process involves a larger number of users.

Forces. When applying the pattern you should consider:

- Undoing actions performed in a collaborative context may affect the other collaborators' contributions, which could be related to the one being undone.
- Misunderstandings are common to collaborative processes; hence it might happen that parts of the document collaboratively edited become redundant or unnecessary.

Solution. Therefore: Track changes performed by each collaborator and allow collaborators to undo modifications on the shared resource. This might ask them to go back to previous versions of the document. The undoing mechanism should allow:

- Undoing the last modification made on the resource,
- Undoing the last n modifications made on the resource, in the reverse order they were made (the last modification made is the first to be undone),
- Going back to a particular version of the document, identified by metadata such as the date of the creation, the date of the last modification,
- Selecting the changes to give up from a chronological list of all the changes performed on the shared resource by the collaborators.

Consequences. Applying the pattern leads to:

- The possibility of undoing changes to the shared resource without affecting other's contribution.
- Allowing losing partial contributions to the document collaboratively edited.

References. Specific situations of collaborative undo are discussed by Lukosch and Schummer in [94]:

- The case in which a collaborator has performed conflicting changes is addressed in *Optimistic concurrency control*. As solution, the authors propose rolling back or transforming the change.
- The authors also propose a way to detect such conflicting changes, the issue being addressed in *Conflict detection*.

Examples. LucidChart is one example of applications which supports reverting changes or starting a new document from a previous existing version of one. Codox-Word supports the undo of any editing actions without affecting the work of the other collaborators.

4.4.11 Support versioning

Context. A group of collaborators contribute to the development of a shared common resource. This development processes could have different phases, each phase providing an intermediary version of the resource.

Problem. Users may be interested in the evolution of the shared resource, the object of the collaboration. How to allow them to visualize or/and edit the previous states of the shared resource? Moreover, how to support users in exploring different alternatives in the development of the resource, and possibly develop (in parallel) two paths that have started as one?

Symptoms. You should consider applying the pattern when:

- The shared resource is subject to various phases of development, the results of each phase being needed for further developments.
- Users need to go through, and possibly edit older versions of the resource they are sharing at a given moment.

Forces. When applying the pattern you should consider:

- Often times, collaborators get insight from following the evolution of the resource they commonly share and work on. For that, they need a way to visualize the various versions of the resource and the changes performed from one version to another.
- Keeping a history of the versions of a document asks for a structure able to store the tree-like evolution of resource, and which includes a naming convention/protocol.
- Domains such as games may not need a mechanism for tracking the versions of the shared game board since going back to previous states of the game does not necessarily occur.

Solution. Therefore: Support the creation of a versioning system for each document edited. This would consist in:

- Creating an initial version of the document at its first creation. This version would be stored by the versioning mechanism.
- At every save of the document a new version is added to the versioning system of the document.
- Going back to a previous version, editing, and saving it leads to the creation of a branch in the versioning system of the document. The branch would start from the version chosen for editing prior to any modification being made on it.
- Each version stored is identified by a set of metadata comprising an identifier of the version, the changes it includes, and the author(s) of these changes.

Consequences. Applying the pattern leads to:

- A better organization of the collaborative process.
- An efficient structure for tracking the evolution of the shared resource.

- Means of documenting aspects of the collaborative process.

References. Lukosch and Schummer address the issue of versioning in the pattern *Immutable versions*, where they suggest storing all shared objects in a version tree making sure that all the versions stored are immutable. As a consequence, any modification of the object will then be stored as a new version.

Examples. CO2DE supports versioning by creating a mask structure of the document edited which shows how work has evolved. Such a mask stores the contributions of each collaborator, the changes made from the previous mask, alternative proposals, and negotiations among collaborators. Released in 2007, CodoxWord is a real time sharing and group editing tool. CodoxWord allows the recovery from errors and the backtracking to previous versions of the shared document through a versioning mechanism.

4.4.12 Shared summary

Context. The collaborative process ended and, after a while, the collaborators want to go through the highlights of their collaboration. They would either want to know the final results of their process or visualize statistical data related to their collaboration.

Problem. How to allow and support the collaborators to go through the highlights of their collaborative process without being forced to reread all the details of the process at the end of a collaborative session or longer after the session ended?

Symptoms. You should consider applying the pattern when:

- The highlight of the results of the collaboration is relevant to the user even long after the collaborative process ends.
- The domain of the collaboration and the type of the shared resource allow the summarization of any relevant data.

Forces. When applying the pattern you should consider:

- Tracking the history of a collaborative process may lead to large documents, hard to read, understand and search in. Results obtained through collaborative processes may be useful long after the process ends. Hence, collaborators are forced to go back and search for those results mostly relevant to the collaboration.
- For some domains, collaborative processes follow cycles of development and the results relevant to the overall process may be the end results of each cycle. An example for that is collaborative searching where one cycle would represent the process of reiteration of a query until all collaborators are satisfied with the results.

Solution. Therefore: Automatically generate a summary of the results obtained through the collaborative process and make it available to all collaborators. This summary may include:

- Results obtained through the collaborative search, in case of applications targeting collaborative searching. Collaborative searching usually goes through more iterations, each iteration contributing a set of partial results to the final result. The collection of these partial results would help both document the searching process and provide a summary of the results obtained at each step of it.
- Statistics on the evolution of the collaboration, in case of collaborative game applications. Games support competition among various teams. Generating a shared summary with the partial results of the game at various moments in the game allows tracking the highlights of the game's evolution.
- Simplified lists of the changes performed on the shared resource, in cases such as collaborative text editing. Going through all the changes performed on a text during a synchronous collaborative editing session may be tedious and inefficient. Therefore, filtering these changes based on some criteria (such as time of editing, author, keyword) and providing a summarized list of them would support collaborators in finding relevant information related to the process even after this has ended.

Consequences. Applying the pattern leads to:

- Supporting collaborators in filtering the essential results of their collaboration.
- Documenting the collaborative process, making available its summarized results even after it ends.

References. None of the patterns available in the literature address specifically this problem.

Examples. An earlier version of WeSearch is SearchTogether, an application which was designed to enable both synchronous and asynchronous remote collaboration in web search. The application automatically creates for each collaborative search session a shared artifact that summarizes the findings of the search session. The particularity of the application is that this summary is also interactive in that one user may access the results of the search directly from the shared summary.

4.4.13 Adapt application to device

Context. Groups of users who use different devices (e.g. mobile phones, iPads, laptops) wish to collaborate in real-time using the same application.

Problem. The problem is to support each user in using the device of his/her choice while still being able to collaborate with other users which use different devices.

Symptoms. You should consider applying the pattern when:

- The system is designed for a team whose members are obliged (due to their role in the collaboration) to use different devices for interacting with the system.
- The system is designed for larger communities.

Forces. When applying the pattern you should consider:

- Different devices have different technical constraints. Screen size, memory size, input/output means impose a set of technical constrains that affect the design processes of applications. Hence, it is required that throughout the design of an application which targets several devices such constraints are considered.
- On the other hand, providing different designs for one application (a design for each device) highly affects the overall quality of the final software product in terms of maintenance, understandability, and reusability. Keeping consistent a large number of instances of the same application is error prone and risky.

Solution. Therefore: Provide different interaction techniques for different devices and support the materialization of one application on different devices. Materializing a specific application on a set of different devices asks for a level of abstraction which decouples the technical details characterizing the device and the specification of how these details affect the materialization of the system on that device. Describe the following characteristics independently of the functionality of the application (common on all devices):

- Display characteristics refer to the screen size, and resolution. The materialization of a dialog box might require one screen on a laptop and several screens on an iPhone, even if the functionality behind it is the same. Hence, decouple the representation of the appearance of the application from the implementation of its functionality. Design the GUI for each device independently from the single implementation of the functionality.
- Input/output means differ from one device to another. For example, the same application may be materialized on a laptop device which uses a mouse and a standard keyboard and on a tabletop device which uses gestures for input.
- Interaction affordances are dictated by the nature of each device and must be considered accordingly. As example, a tabletop allows all collaborators to gather around it, hence being able to communicate more efficiently. A mobile device allows each collaborator to move freely while interacting with it, having in the same time a lower attention span. Therefore, consider such implications when

designing for a specific device.

Consequences. Applying the pattern leads to:

- Providing different materializations of the same system, suited for different devices.
- Being able to decouple the functionality of the application from its appearance and interaction affordances on different devices. This facilitates the maintenance of the application and its understandability.

References. The purpose of this pattern is not to go into hardware and architectural details on platform specific adaptation, but to point out the issue and hint on how to tackle it. A thorough discussion on such details is presented by Nobel and Weir in their book "Small Memory Software: Patterns for Systems with Limited Memory" [76] which contains patterns on issues such as:

- managing memory use;
- using secondary storage;
- compressed representation of data;
- memory allocation techniques;

Examples. NetDraw is a Java application which provides 2D collaborative drawing features in a client-server architecture. It has a thin client, suitable for running on any device that supports Java. LucidChart is a web tool released in 2008 which supports the collaborative drawing of diagrams such as UML diagrams, flow diagrams. LucidChart is available on any device that supports browsing.

4.4.14 Customize collaboration

Context. A well formed community, following a set of rules, collaborates through the application. The members of the community have diverse backgrounds and expertise and there is a protocol of collaboration they follow. They may not find useful a set of default options embedded in the application.

Problem. How to support collaborators in customizing their collaborative process based on the rules of their community and on their preferences?

Symptoms. You should consider applying the pattern when:

- The system is addressing well-formed communities.
- The collaborative process is clearly defined, assuming that users have particular roles in the collaboration.
- The visualization and the editing of the shared resource support different parameters of customization.

Forces. When applying the pattern you should consider:

- One of the issues of concern in collaborative processes is the division of labor, i.e. the rules based on which collaborators decide what will each of them be responsible for. For that, each group should decide on the ways to adapt this division to its dynamics.
- More than often collaboration brings together people with different backgrounds and expertise. In order to support them in complementing each other, the application should allow them to choose the collaborative tasks they want to be in charge of.

Solution. Therefore: Support collaborators in customizing their collaborative processes by allowing them to:

- Assign each other roles in the collaborative process. Each collaborator might choose the role for him/her self to have in the process or one of the collaborators – the coordinator – assigns the roles for the others.
- Assign each other rights on the collaborative resource that would allow them to perform different tasks on the same document. Each collaborator might choose the rights for him/her self to have in the process or one of the collaborators – the coordinator – assigns the rights for the others. For example, one might choose to only be able to visualize the shared resource and have no possibility to edit it.
- Set visualization or editing options such as: the size of each area of the user interface, and/or document specific editing features. Once a user changes such options, all the modifications are made visible to all the rest of the collaborators.

Consequences. Applying the pattern leads to:

- Providing the collaborating users with the freedom of deciding the parameters of their collaborative process.
- This would be useless or even inefficient in the cases when the collaborators are not familiar with one another or do not follow a well-defined process in their work.

References. In their paper, "The Role of Roles in Computer-mediated Interaction"[68], Lukosch and Schummer address the issue of customizing a collaborative process by proposing three patterns on this:

- The pattern *Role* aims at modelling "the expected interaction in the collaborative application". The pattern answers the problem of users finding it difficult to structure their interaction in the group using the application collaboratively, proposing as solution the definition of roles which describe "what the owner of the role is supposed to do".
- In order to make all the members of the group aware of each other's role, the pattern *Role indicator* suggests visualizing the role of each user whenever the user is shown in the interface.

- Users who are not interested in participating in the collaborative process, but in observing it are allowed to have the role of *Spectator*.

Examples. CodoxWord allows the assignment of different roles and rights to the collaborators editing one document. In EtherPad, any collaborator may set visualization options of the shared workspace. These settings automatically affect the view of all the other collaborators and examples of such options are: the display of the number of each line within the document, or of the authorship colors (the color identifying a specific collaborator's contribution).

4.4.15 Resume collaboration

Context. Breaks and interruptions may occur in real-time collaboration. At some point during the collaboration of a group of users, their process is interrupted by one of the collaborators who leaves temporarily the application.

Problem. Due to interruptions, users might want to pause the collaborative process for a while and resume it later, starting over from the state they left the application in.

Symptoms. You should consider applying the pattern when:

- The presence of all the collaborators is vital to the collaborative process.
- The absence of one user generates a deadlock in the collaborative process.

Forces. When applying the pattern you should consider:

- Synchronous collaboration requires all collaborators to be present and use the application in the same time. One collaborator's absence may trigger overhead to the overall collaborative process; hence, all collaborators are forced to pause their work for a while.

- Reconstructing the partial results of a collaborative process from scratch may require a lot of extra effort and time and may not lead to the same results obtained in the first place.

Solution. Therefore: Allow collaborators to pause the collaborative process by providing intuitive controls for that action, Furthermore, store the state of the session they shared in a format that would allow its resume. Pausing may occur in one of the following situations:

- After all collaborators agree to it, case in which the entire collaborative process is paused. Restoring the session would be initiated by one collaborator.
- One user decides to take a break, case in which the pause only occurs on his/her side. This disables him/her from viewing the interactions of the other users and the evolution of the shared resource. Also, his/her absence is signalled to all the other collaborators. Once s/he resumes his/her participation to the collaborative process, s/he gets an update of the changes performed meanwhile by his/her collaborators. Also, s/he may continue his/her interaction with the shared resource in its current state.

Consequences. Applying the pattern leads to:

- Allowing each collaborator to signal his/her absence and allow the others to pause the process.
- A mechanism for dealing with interruption without losing any information or putting the burden on part of the collaborators.

References. The pattern *Persistent session* touches on the issue of resuming the collaborative process described above by aiming to make the results obtained in a collaborative session "available for reviewing or resuming collaborative activities". For that, it suggests storing the results of a collaborative session on a central server and allowing users to access this master copy for either reviewing purposes or for resuming the session.

Examples. WeSearch is a tabletop application which supports collaborative Web search among groups of up to 4 users. Each collaborator is allowed to pause the collaborative process. As result, session files store the current state of a session and allow its resume. These files store each session in a format viewable on other platforms as well, hence the results of a resumed collaborative session can be accessed on other devices.

—

In this chapter, I describe a design pattern mining method to be used in interaction design together with its application and the results derived through that - a collection of 15 design patterns for the design of applications which support synchronous collaboration. The definition of this method is based mostly on techniques broadly used in interaction design, but not consider in the context of pattern identification. The main *strength* of the method relies in the fact that it supports the exploration of different data sources, allowing the correlation of the different results obtained. Both design workshops and existing applications analysis are used for identifying recurring design problems and solutions to tackle them. In addition to that, the method may easily be replicated in other mining areas, allowing the identification of patterns for various areas of interaction design.

One of the *limitations* of the method right now relies in the fact that it requires several other evaluation cycles for better understanding how far it can go. According to such cycles, the method might go through slight modifications in order to adapt to the requirements of other areas of design. The purpose of this work is to look at one such application cycle and try to derive some lessons learned based on it. Crucial to the success of this method is the involvement of professional designers during the workshops as well as the consideration of a large collection of software applications during the second phase. The larger the pool of design issues collected, the better. Compared to the other methods described for pattern mining (see 2.2.5), the method presented below provides a structured approach for pattern identification mixing both inductive and deductive techniques, and it uses several sources of data to derive the final results. Once described, the identified patterns may be validated by existing literature or related collections.

This chapter describes 15 patterns derived using the method. Once the recurring design issues have been identified, I proceeded to actually writing the patterns. This process was supported by literature documenting pattern writing processes [75] and it usually initiated by identifying the core problem addressed by the design issue and the solution most commonly adopted for tackling it during the workshops and through the implementations considered. At this point, this chapter does not propose any grouping for the patterns, this issue being the core topic of the next chapter.

Chapter 5

Relating Patterns: From the Collection to the Language

Most of the available collections of design patterns are organized as pattern languages. On one hand, patterns are never isolated, relationships between them existing. On the other hand, pattern language structures are easier understood and browsed. Even if authors of patterns usually group and relate the patterns they describe, none of these processes are documented. The goal of this chapter is to describe a method for generating a pattern language out of a collection of design patterns. Such efforts address mainly pattern authors willing to structure their collections in pattern languages and heavily support pattern users in understanding and making use of patterns organized in such fashion.

5.1 Definition of a Pattern Language

In Christopher Alexander's words, "*a pattern language has the structure of a network. [...] when we use the network of a language, we always use it as a sequence, going through the patterns, moving always from the larger patterns to the smaller, always from the ones which create structures, to the ones which then embellish those structures, and then to those which embellish the embellishments...*

Since the language is in truth a network, there is no one sequence which perfectly captures it. But the sequence which follows, captures the broad sweep of the full network; in doing so, it follows a line, dips down, dips up again, and follows an irregular course, a little like a needle following a tapestry.

The sequence of patterns is both a summary of the language, and at the same time, an index to the patterns. If you read through the sentences, you will get an overview of the whole language. And once you get this overview, you will then be able to find the patterns which are relevant to your own project" [9].

Borchers [22] brings some formalization to the notion of pattern language, defining a formal syntactic notation:

1. A pattern language is a directed graph PL = (P, R) with nodes P = p_1, ..., p_n and edges R = r_1, ..., r_m.
2. Each node p ∈ P represents a pattern.
3. For two nodes p, q ∈ P, we say that P references Q if and only if there is a directed edge r ∈ R leading from p to q.
4. The set of edges pointing away from the node p ∈ P is called its references and the set of edges pointing to it is called its context.

Borchers addresses the rationale behind using design patterns, and more particular pattern languages pointing out that "to use the pattern language framework in the process of designing an interactive software system, it is not mandatory to follow a single fixed design method" [22]. In other words, the use of pattern languages flexibly fits into several development processes and their steps, patterns appearing at most of the stages of development as follows:

1. Know the user [22]. The issue at this point is identifying whether the application domain considered is suitable for expressing its concepts in pattern format. As hinted by the author, any application domain comprising "some sort of creative,

designing, or problem-solving activity" is subject to being expressed in terms of patterns "because the rules and guidelines that lead people in that application domain in their activity can be formulated as design patterns". Using the pattern format may be regarded at this stage as a convention for putting on paper requirements-related issues which are tracked anyway, but with an explicit focus on capturing forces, alternatives, and connections in the working patterns provided by the users.

2. Competitive analysis [22]. During this phase, an investigation of similar products existing in the market is performed for identifying "different solutions to the problems of the product area". Through a process similar to the one described in Chapter 3, successful recurring solutions observed in competing products can be captured and expressed through inter-related design patterns, i.e. pattern languages.

3. Setting usability goals [22]. Usability goals include learnability, efficiency of use, memorability, low error rate. They get weighted against each other and prioritized. In terms of design patterns, they translate into forces in patterns, explaining the trade-offs of each of these goals.

4. Parallel design [22]. Parallel design allows the exploration of a larger design space through the design of several initial prototypes of a user interface. These prototypes can be designed in parallel by independent teams. Similar to guidelines, design patterns can at this point provide a source for consistency across parallel designed prototypes, creating a common ground and preserving the usability goals. In addition to that, patterns provide a shared vocabulary within a team supporting efficient communication and ensuring that "the same design concepts are known and respected throughout the interface".

5. Empirical testing [22]. Design patterns for the application domain may constitute a "resource to construct realistic scenarios for testing", closely modeling the

concept pertaining to the domain itself. Moreover, problems discovered during testing may further be documented as design patterns, since a pattern language is a dynamic structure open to changes throughout its application. Therefore, "all pattern languages used will and should evolve even during the project, to capture the progress in understanding the problem space and improving the design solution".

6. Collect feedback from field use [22]. Deploying an application asks for collecting further feedback from its actual use in the field. This type of knowledge is expressed by the users, following to be used by the designers for improvements. Therefore, the vocabulary used for such purposes should bridge different communities (assuming that the end-users work in domain remote from software design). At this point, the application domain pattern language "plays an important role as a common vocabulary between UI experts and users". Moreover, feedback is also a possible source of improvement suggestions for the patterns in themselves, since successful solutions documented through patterns find their validation, while suboptimal results get to be reconsidered.

5.2 Representing and Visualizing Pattern Languages

In 2003, the participants at the CHI workshop "Perspectives on HCI patterns: concepts and tools" defined an XML DTD for a language for design pattern description, the Pattern Language Markup Language (PLML) [1]. The main goal for the definition of such a language was to reach a consistent template across those pattern authors have used by providing a general structure for pattern documents. Some of the consequences of this are:

- Patterns from a collection could refer to patterns in other collections.
- Common elements across collections of patterns can be identified.
- Patterns from disparate authors could be combined into specific, thematic collections.

The elements included in the DTD are: pattern_id, name, alias, illustration, problem, context, forces, solution, synopsis, diagram, example, rationale, confidence, literature, implementation, related_pattern. In addition to that, PLML contains a series of elements that indicate authorship and change management; they are: author, credits, creation_date, last_modified, and revision_number.

Being defined as graphs, pattern languages may use the visualization benefits provided by specialized graph visualization tools. A brief survey of such tools includes the following:

- *Medusa.* Medusa is a graph visualization tool developed in Java. It is designed to be "a simple and an intuitive tool for customization of interaction graphs of any kind" [55]. Medusa handles large graphs easier by layout algorithms and the option to hide certain edge types. Users can create their own data files which are simply tab-delimited text fields describing edges relationships. These files are the input for the graph rendering process. Medusa displays up to 10 multiple edges concurrently between nodes and several node properties can be described, such as color, position, shape, and annotation.

- *Graphviz.* Graphviz is an open source graph visualization software. The Graphviz layout programs take descriptions of graphs in a simple text language, and make diagrams in useful formats, such as images and SVG for web pages, PDF or Postscript for inclusion in other documents or display in an interactive graph browser. Graphs may be generated from external data sources, but they can also be created and edited manually, either as raw text files or within a graphical editor provided by Graphviz.

- *Gephi.* Gephi is an open source software for graph and network analysis [19]. The goal of the tool is to "help data analysts make hypothesis, intuitively discover patterns, isolate structure singularities or faults during data sourcing". The visualization module uses a special 3D render engine to render graphs in real-time. Highly configurable layout algorithms can be run in real-time on the graph window, this feature providing especially useful for displaying large graphs. The architecture of Gephi supports graphs whose structure or content varies over time, and proposes a timeline component where a slice of the network can be retrieved at different times.

- *CCVisu.* In the specialized category of software-structure extraction, there is CCVisu, "a lightweight tool that takes as input a software graph model and computes a visual representation of the system's structure, i.e., it structures the system into separated groups of artifacts that are strongly related" [21].

5.3 Pattern Language Generation Method

Generating a pattern language implies:

1. *identifying a collection of design patterns* and
2. *identifying the relationships existing between the patterns in the collection*

An overall description of the methods to be used in identifying patterns is covered by this thesis in Section 2.2.5. Moreover, in Chapter 4, I described, as one of the results of my work, a pattern mining method for interaction design. In this section, I will cover the second step in the generation of a pattern language, presenting an approach for identifying relationships between design patterns in an existing collection.

The approach is based on the definition of an ontology for representing the domain addressed by the pattern collection. As examples of domains, one could consider interactive exhibits [22], e-government [70], web accessibility [45] or, as in the case described in this thesis, synchronous collaboration. A domain is initially described by a set of design issues, these design issues being collected during design workshops and through the analysis of software applications in the particular domain (more on this in 4.2). They represent problems, solutions, examples, consequences, and any other information relevant to the design of a system in the area targeted, therefore bits of knowledge from a specific domain. Design patterns are documentations of those design issues mostly discussed throughout the workshops and mostly considered in the implementation of concrete applications.

The ontology is build in two steps: 1) by identifying the set of concepts it models, and 2) by determining the relationships between these concepts. The set of concepts includes two types of concepts: design issues, and keywords. The keywords are part of a glossary able to comprise elements of the domain's vocabulary. The set of relationships are identified between design issues and keywords, and among keywords. The approach makes use of 5 types of relationships: Equality, Equivalence, Specialization, Composition, Association - to be detailed further on in the chapter. Therefore, each design issue is associated with a set of keywords and different relationships are identified between pairs of keywords. Once the ontology defined, it drives the generation of a structure connecting design issues, and implicitly those design issues further documented as patterns. This structure is the basis of the pattern language.

More formally, three phases are defining the approach: 1) the *concepts identification* phase, 2) the *relationships identification* phase, and 3) the *pattern language generation* phase.

5.3.1 Concept Identification

During a series of design workshops and through the analysis of a set of software applications, a collection of design issues, D = $\{d_1, \dots , d_n\}$, is collected (Section 4.2). These issues represent problems, solutions, examples, consequences, and any other information relevant to the design of a system in the area targeted. Each design issue has a unique identifier, this identifier being used throughout this process for pointing to a specific design issue. Once a design issue gets documented as a design pattern, the identifier remains unchanged and it is used in the identification of the pattern.

An initial step towards the generation of a pattern language consists in associating each design issue (d_i) with a set of keywords (K_i). In order to avoid undesired noise in the finally generated pattern language, a word can be a keyword only if used in less than 10% of the associations. Hence,

$$\forall d_i \in D, \exists K_i = \{k_{i1}, k_{i2}, ..., k_{in_i}\} s.t. d_i \rightarrow k_{ip}, p = 1, ..., n_i \tag{5.1}$$

, where $\rightarrow$ is the association function[1].

These keywords are words belonging to the statement of the issue and/or words strongly related to the statement of the issue. As example, the design issue "*Support collaborators in providing tags, comments, annotations, rankings*" is associated with the set of keywords - *tagging*, *ranking*, *comments*, *social*, *community*.

Associating each of the design issues considered with a set of keywords leads to:

$$\begin{gathered} d_1 \rightarrow K_1 = \{k_{11},\ ...,\ k_{1n_1}\} \\ d_2 \rightarrow K_2 = \{k_{21},\ ...,\ k_{2n_2}\} \\ ... \\ d_n \rightarrow K_n = \{k_{n1},\ ...,\ k_{nn_n}\} \end{gathered}$$

The set

$$DIM = \{(d_i, k_{i\beta}) / d_i \in D, k_{i\beta} \in K_i, d_i \rightarrow K_i\} \tag{5.2}$$

is defined as the *Design Issues Map* and represents all the pairs associating a design issue with a keyword. The set of all the keywords is:

$$K = K_1 \cup K_2 \cup ... \cup K_n \tag{5.3}$$

and it represents a glossary comprising elements of the domain's vocabulary.

The input of this first phase consists in the collection D of design issues, used for describing a specific design domain. The output, on the other hand, is the set of keywords K 5.3 and the set DIM 5.2, pairing design issues with associated keywords (Figure 5.1).

[1]The association function directly maps a design issue to a set of keywords, hence implicitly to each keyword in the set.

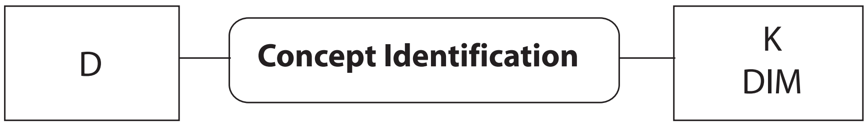


Figure 5.1: Concept Identification Flow

5.3.2 Relationship Identification

A second step towards the pattern language generation is the identification of relationships between the keywords, i.e. the elements of the set K 5.3. The approach makes use of 5 types of relationships, R:

1. *Equality* ("="). Two keywords – k_1, k_2 – are in a "=" relationship if k_1 is equal to k_2.
   As example, "collaborator" = "collaborator".

2. *Equivalence* ("≡"). Two keywords – k_1, k_2 – are in a "≡" relationship if k_1 is a synonym of k_2.
   As example, "online" ≡ "web_based".

3. *Specialization* ("ISA"). Two keywords – k_1, k_2 – are in a "ISA" relationship if k_1 is a sub-type of k_2.
   As example, "tabletPC" ISA "device".

4. *Composition* ("HASA"). Two keywords – k_1, k_2 – are in a "HASA" relationship if k_2 is a part of k_1.
   As example, "library" HASA "templates".

5. *Association* ("RELATEDTO"). Two keywords – k_1, k_2 – are in a "RELATEDTO" relationship if k_1 is associated to k_2.
   As example, "Instant_Messaging" RELATEDTO "communication".

   Moreover, the RELATEDTO relationship is enhanced with a description explaining the association between the 2 keywords. Going back to the example, "Instant_Messaging" $RELATEDTO_{used-for}$ "communication".

Identifying all the relationships between the keywords, leads to the definition of the *Keywords Map*, representing all the pairs of related keywords:

$$KM = \{(k_{\alpha x}, k_{\beta y}, R)/k_{\alpha x} \in K_\alpha, k_{\beta y} \in K_\beta, k_{\alpha x} R k_{\beta y}\} \tag{5.4}$$

, where R $\in$ {*Equality*, *Equivalence*, *Specialization*, *Composition*, *Association*}.

K
Relationship Identification
KM

Figure 5.2: Relationship Identification Flow

The input of the second phase is the set K 5.3 of keywords, decided on during the first phase. The output of the *Relationship Identification* phase consists in KM 5.4, the Keywords Map which pairs any two related keywords (Figure 5.2). The Design Issue Map (DIM) 5.2 and the Keywords Map (KM) 5.4 make up the ontology supporting the generation of the pattern language as specified below.

5.3.3 Pattern Language Generation

The generation of the pattern language requires two sub-phases: 1) the generation of the *Design Issue Language*, and 2) the generation of the *Pattern Language*.

5.3.3.1 Design Issue Language Generation

In identifying the relationships between the design issues collected, the following generation rule is applied:

Consider $d_\alpha \in$ D and $d_\beta \in$ D

$$d_\alpha \rightarrow K_\alpha = \{k_{\alpha 1}, ..., k_{\alpha n_\alpha}\}$$
$$d_\beta \rightarrow K_\beta = \{k_{\beta 1}, ..., k_{\beta n_\beta}\}$$

Then, d_α R d_β if $\exists\ k_{\alpha x} \in K_\alpha$ and $k_{\beta y} \in K_\beta$ such that $k_{\alpha x}$ R $k_{\beta y}$.

This is to say that a design issue, d_α is in a relationship R with another design issue (i.e. is related to), d_β if d_α is associated with a keyword, $k_{\alpha x}$ which is in a relationship R with another keyword $k_{\beta y}$ to which d_β is associated (Figure 5.3).

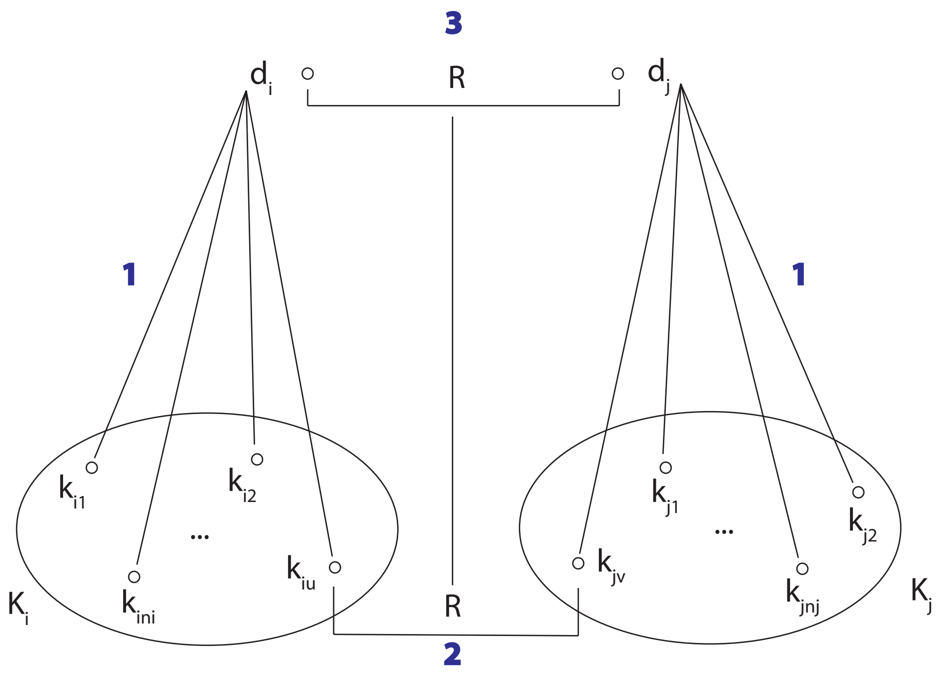


Figure 5.3: Illustrating the rule used for identifying relationships between design issues

The rule is applied to all the design issues in the set D, generating the *Design Issue Language* (DIL), comprising all the design issues and the relationships between them.

$$DIL = (D, \{(d_\alpha, d_\beta, R)/d_\alpha \in D, d_\beta \in D, d_\alpha R d_\beta\}) \tag{5.5}$$

For outputting the DIL 5.5, this sub-phase takes as input the collection of design issues, D, together with the set KM 5.4, the output of the previous phase, and the set DIM 5.2 (providing information on the keywords associated to each design issue) (Figure 5.4).

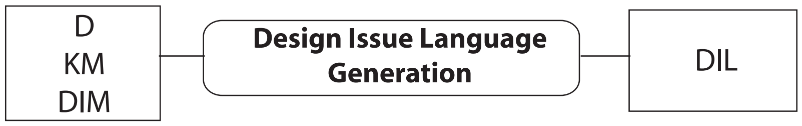


Figure 5.4: Design Issue Language Generation Flow

5.3.3.2 Pattern Language Generation

Not all the design issues are design patterns. All the design issues discussed throughout the workshops and all those considered in the implementation of concrete applications are collected. For each design issue, its degree of recurrence is computed as the percentage in which the issue has been addressed throughout the workshops (DoR_w) and the implementations considered (DoR_s), as defined in Section 2.2.5. Further on, the design issues are sorted based on their degrees of recurrence, and those design issues with the highest degrees are documented through design patterns. Hence, a design issue, d_i, is considered to be documented as a pattern, p_i, if $DoR_w(d_i)$ and $DoR_s(d_i)$ belong to the top DoRs computed with respect to both the workshops and the applications analysis.

Consider P = $\{p_1, p_2, ..., p_n\} \subseteq$ D the set of design issues with the highest degrees of recurrence and which are further documented as design patterns. Therefore, the *Pattern Language* is defined, as:

$$PL = (P, \{(p_\alpha, p_\beta, R)/p_\alpha \in P, p_\beta \in P, p_\alpha R p_\beta\}) \tag{5.6}$$

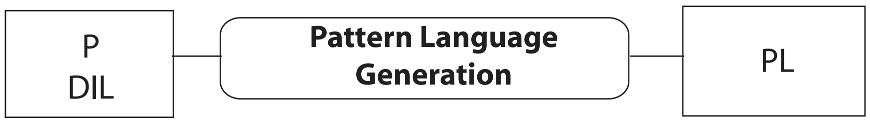


Figure 5.5: Pattern Language Generation Flow

For outputting the *Pattern Language*, the input for this sub-phase is the collection of patterns, P, and the language DIL 5.5 (Figure 5.5).

The generation of the DIL 5.5 offers the possibility of placing the pattern language in a context, modelling - together with KM 5.4 - the domain addressed by the patterns, and supporting a better understanding of each pattern and the implications of its use. Moreover, design issues that are not documented as patterns might contain information relevant to the considered design patterns. This allows designers to better investigate specific design problems/solutions that have not been included in the patterns yet.

5.3.4 Overall Process

To summarize, this section describes the entire workflow in the application of the method. As described in Figure 5.6, there are 9 steps to it:

1. The set of collected design issues, D, triggers the identification of the set of design patterns, P (see Section 4.2 for details).
2. The set D is the input of the first phase of the process, the *Concept Identification* phase. The goal of this phase is to associate each design issue with a set of keywords which both characterize and relate to the statement of the issue.
3. The *Concept Identification* phase outputs two sets:
    - The set DIM 5.2 containing pairs of the form (d_i, k_{ij}), associating a design issue (d_i) with a keyword (k_{ij}).
    - The set K 5.3 containing the set of all the keywords used in the above mentioned associations.
4. The set K 5.3 is the input of the *Relationship Identification* phase. The goal of this phase is to identify possible relationships between the keywords in K 5.3. The five types of relationships used are *Equality*, *Equivalence*, *Specialization*, *Composition*, *Association*, a complete description of them being available in Section 5.3.2.
5. The *Relationship Identification* phase outputs the set KM 5.4, which contains tuples of the form $(k_{iu}, k_{jv}, \text{R})$, representing pairs of related keywords.

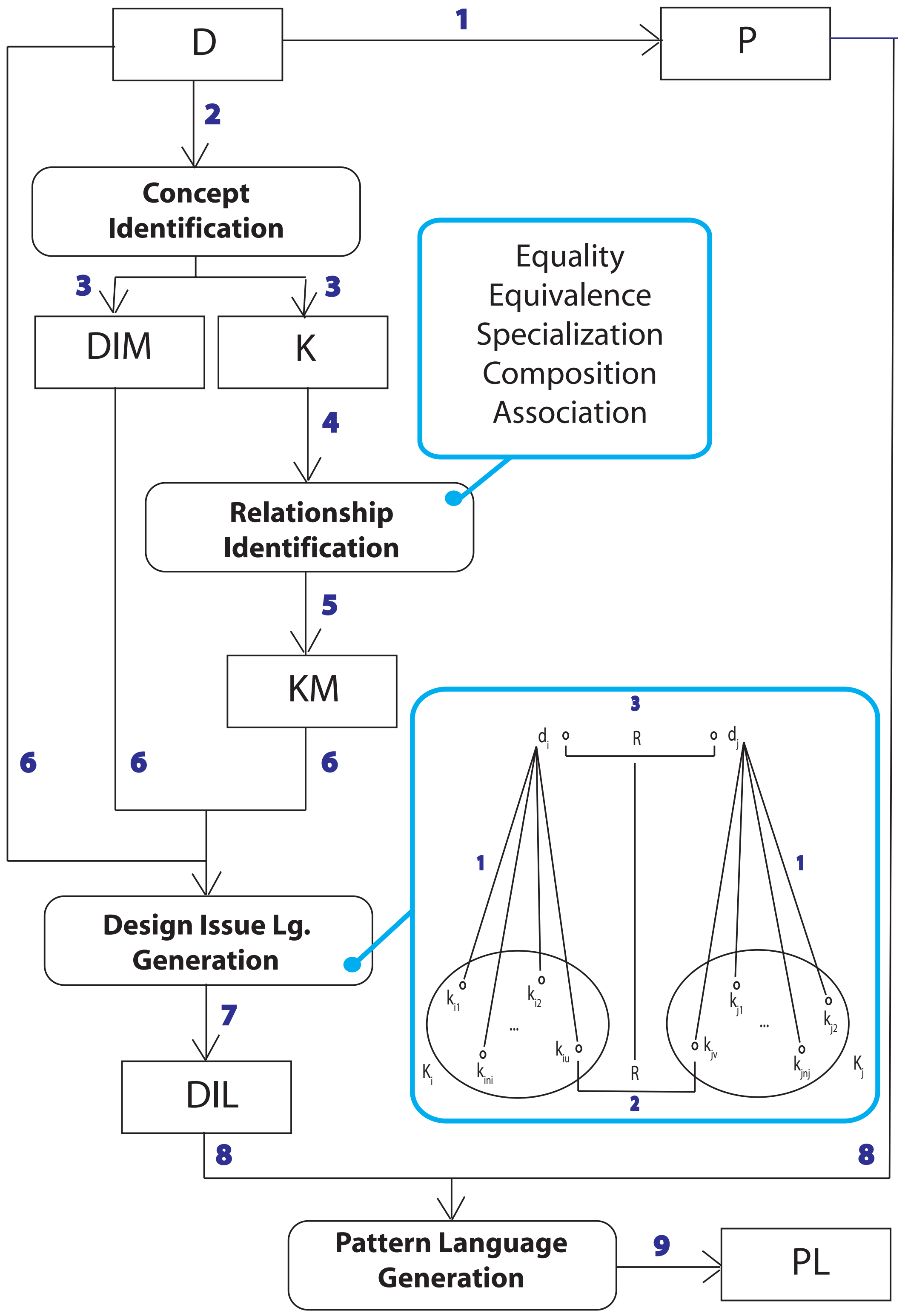


Figure 5.6: Pattern Language Generation Overall Flow

6. The *Design Issue Language Generation* phase takes as input the sets D, DIM 5.2, and KM 5.4 and has as goal deriving relationships between the elements of the set D according to the rule 5.3.3.1.

7. The output of the *Design Issue Language Generation* phase is the Design Issue Language 5.5, a directed graph structure comprising the set of design issues (the nodes) and all the relationships identified between them (the edges).

8. DIL 5.5 is the input of the *Pattern Language Generation* phase. In order to generate the pattern language, all the design issues considered for being documenting as patterns are extracted from DIL 5.5. The relationships identified between the issues are considered as relationships between the corresponding patterns.

9. The *Pattern Language Generation* phase outputs the pattern language, PL 5.6 representing the directed graph structure containing all the patterns (the nodes) together with the relationships between them (the edges).

5.3.4.1 Associated-to vs. Related-to

Throughout this chapter, I have used both "*...is associated to...*" and "*...is related to...*" in various contexts. These two mean different things in different contexts and help framing the process. At this point, some clarification notes are in order:

- A design issue/pattern *is directly associated to* a set of keywords, hence to each of the keywords in the set.

- Two keywords *are directly related to* each other, a relationship R being identified between them.

- Two design issues/patterns *are indirectly related to* each other, the relationship between them being derived according to the generation rule 5.3.3.1.

Design issue	Keywords
6 (Integrated chat)	*communication*, *Instant_Messaging*, *chat*
7 (Eyes wide open)	*visualization*, *awareness*
9 (Collaboration, always social)	*tagging*, *ranking*, *comments*, *social*, *community*
1 (With or without collaboration)	*separate layers*, *non-collaborative*, *collaborative*, *brainstorming*
33 (Customize collaboration)	*customize*, *collaboration*, *community*, *rights*, *roles*

Table 5.1: Design Issues Map (fragment)

- A design issue/pattern *is indirectly related to* a keyword (k) if the set (K_i) of keywords associated with the design issue/pattern contains a keyword ($k_{i\alpha}$) such that k R $k_{i\alpha}$.

5.4 The Method Applied

Starting from the collection of patterns described in the previous chapter, I went through the steps depicted in Figure 5.6 in order to derive possible relationships between the patterns in the collection.

5.4.1 Identifying Concepts

The set of design issues, D, is described in details in Section 4.3. Out of this set, a collection of 15 patterns were derived 4.4. As a first step in the process, I associated each design issue in D with a set of keywords. Table 5.1 presents some examples of design issues and the keywords associated to them, the complete DIM 5.2 set (i.e. the output of this first phase) being presented in Appendix 7.3.

Each design issue was represented by its unique identifier, the mapping between identifiers and design issues following the description in Tables 4.6 and 4.7. For associating the issues with keywords, I split the statement of the issue in words and I considered as keywords the representative words with a less than 10% degree of usage in the associations. In addition to that, for some of the design issues, I associated as keywords words closely related to the statement of the design issue (a major consequence, a possible symptom).

5.4.2 Identifying Relationships

As a second phase of the process, I proceeded to identifying relationships between the keywords. The complete KM 5.4 set (i.e. the output of this phase) is described in Appendix 7.3, some examples of such relationships being presented below:

- *awareness* = *awareness*,
- *groups* ≡ *teams*, *availability* ≡ *user_status*, *initiator* ≡ *first_editor*,
- *pdf* ISA *format*, *education* ISA *goals*, *first_editor* ISA *coordinator*,
- *session* HASA *session_state*, *community* HASA *library*, *collaborators* HASA *roles*, *groups* HASA *collaborators*,
- *visualization* $RELATEDTO_{supports}$ *awareness*, *track* $RELATEDTO_{applies-to}$ *changes*, *invites* $RELATEDTO_{triggers}$ *join*, *highlight* $RELATEDTO_{used-for}$ *identify_data*, *join* $RELATEDTO_{applies-to}$ *groups*.

5.4.3 Generating the Pattern Language

Based on DIM 5.2 and KM 5.4, I derived relationships between the design issues, following the generation rule 5.3.3.1. A few examples of such relationship inferences are reported below:

- *tabletop* ISA *device*, hence 22.6 ISA 22,

- $groups \equiv community$, hence $4 \equiv 9$,
- $visualization$ HAS $shared\_resource$, hence 7 HASA 27,
- $rights\ RELATEDTO_{applies-to}\ shared\_resource$, hence 33 $RELATEDTO_{applies-to}$ 27,
- $community\ RELATEDTO_{requires}\ coordination$, hence 9 $RELATEDTO_{used-for}$ 14.

The set of design issues together with all the relationships derived between them form the *Design Issue Language*. The pattern language, PL 5.6, comprising the collection of patterns described in Section 4.4, is obtained after eliminating the design issues which were not considered for being documented through design patterns and all the relationships involving such design issues. At this point, potential areas for automation became clear and the next section aims at clarifying them. Following this discussion, the automatic pattern language generation gets described later on in this chapter.

5.5 Human Intervention and Automation in the Pattern Language Generation

The application of the pattern language generation method raised a couple of questions, opening the issue I raise in this section:

- What can be automatized when applying the method or, in other words, where can the computer help?
- What is the human intervention in the process or, put in other words, what are the decisions that need to be made by the person applying the method?

In answering these questions, I decided to follow each phase of the process described in Figure 5.6 and identify answers to the questions above:

1. *Concept Identification.*

   During this phase, there is room for automation in a few areas. First, natural language processing tools can be used for splitting the statement of each design issue in tokens (words) and, therefore, associate the issue with these tokens as keywords. However, a total automation at this level would generate a lot of noise in the final results since words such as adverbs, prepositions are heavily used, hence being part of each design issue's statement. Ways to reduce such noise include:

   - Allowing human intervention, so that the relevant keywords are decided by the human.
   - Introducing a threshold which would determine whether a word can be considered a keyword (i.e. if a word is used as a keyword for more that a given percentage of design issues, that it is no longer considered a keyword). This makes room for a second possibility of automation, allowing the computation of such percentages for each of the words and deciding on the least used to represent the pool of keywords.
   - A mixed approach using both of the above.

   The representation of the output of this phase - the sets DIM 5.2 and K 5.3 - may take various formats from simple text files using a human-defined protocol to more complex data base structures.

2. *Relationship Identification.*

   During this second phase, relationships between keywords are identified. Even if the set of possible relationships is determined and limited, there is still little room for automation. Except for the case in which all the possible keywords and all the possible relationships existing between them are stored in a dictionary, the tuples representing related keywords 5.4 need to be decided by the person applying the method.

The representation of the output of this second phase - the set KM 5.4 - can be organized through various formats, the basic one being a text file using a human-defined protocol.

3. *Design Issue Language Generation.*

   Deriving the relationships between the design issues in the set D is totally subject to automation. Taking as input the sets D, DIM 5.2 and KM 5.4, it is possible that the machine automatically triggers the relationships between the design issues, applying the rule 5.3.3.1. The representation of the input sets is discussed above and the design issues are all represented by their unique identifiers, a map of the identifiers to the design issues being provided additionally for support.

   The representation of the output of this phase is a graph, which is defined by the set of design issues as the nodes and the relationships between them as the edges. Such representations can be expressed through basic tools such as text files using a human-defined protocol or more complex tools for representing graphs.

4. *Pattern Language Generation.*

   Generating the pattern language translates into identifying a sub-graph of the graph representing DIL 5.5. Knowing which of the design issues are documented by patterns, the generation of the pattern language is subject to full automation as follows:

   - The set of patterns, P, is decided on a priori.
   - The set of relationships between the patterns is obtained after eliminating from DIL 5.5 all the pairs of related design issues which are not further considered to be documented as patterns.

The representation of the pattern language is a graph and, similar to the DIL 5.5, can use complex representation tools or simply text files. The issue at this point is supporting the visualization of the pattern language in a useful and understandable way, text files not falling necessarily in this category. For this reason, specialized tools for graph visualization can be used. Such tools use a textual representation (of a specific format) of a graph in order to represent it graphically. The output of the method - the pattern language - may, therefore, be described in such a format so that it allows its input and graphical representation in a graph visualization tool.

5.6 Tool Support

Thoughts on automatizing the process of generating a pattern language have led me to consider the development of a software tool able to support both the application of the method and the usage of its results. The tool would be used to both automatically generate the pattern language and query the knowledge base represented by the language.

5.6.1 Identified Requirements

As described in Section 5.5, several areas of automation can be identified in the application of the method hereby described. These areas trigger the definition of a set of requirements for the development of a tool able to support the following:

1. *Data Representation.* Each of the data sets considered throughout the process (either as input or output parameters) gets stored in files in defined protocols of representation.

2. *Pattern Language Generation.* The tool automatically generates the pattern language according to the process described in Figure 5.6, being provided with the

sets D, DIM 5.2, and KM 5.4.

3. *Pattern Language Visualization.* The tool supports the export of the pattern language in formats which allow its graphical representation through specialized applications.

4. *Pattern Language Querying.* Designers using a pattern language should be able to query the language. The tool should be able to answer the following types of queries:

   (a) Given the unique identifier of a design pattern, it returns the set of design patterns related to it, specifying for each pattern the type of relationship.

   (b) Given the unique identifiers of 2 design patterns, it returns the type of relationship existing between them.

   (c) Given a type of relationship (R), it outputs the pairs of design patterns between which R exists.

   (d) Given a keyword, it returns all the design patterns associated with that keyword.

   (e) Given 2 design patterns, it returns the set of keywords associated to both patterns.

   (f) Given a set of keywords, it returns the collection of design patterns related to the keywords.

   A detailed description of ”associated-to” and ”related-to” concepts is provided in 5.3.4.1.

5.6.2 Scenarios of Use

Considering the above described requirements, the questions are: ”Who does such a tool address to?” and ”What scenarios of use can be identified?”. A set of such scenarios are described below:

- From a collection to the language.

  First, as one of the goals of the tool is to support the automatic generation of a pattern language, the tool addresses communities of pattern writers interested in translating their collections of patterns in graph structures representing pattern languages. The advantages of a pattern language over a collection of patterns are discussed in details in Section 2.2.7, the main one consisting in the possibility of relating design problems and being provided with solutions for such inter-dependent problems.

- From keywords to patterns.

  As a second goal, the tool aims at supporting designers interested in using already generated pattern languages. Facing a design problem, one would be interested in knowing what are the design patterns describing partially and/or entirely the specific problem. Since more patterns may describe different facets of the same problem, these patterns are inter-related. Hence, the solutions they propose are influenced by each other's forces. The designer faced with this situation should be able to create a query consisting in a set of keywords related to the problem and get as result a collection of design patterns related to the query (i.e. those design patterns related to the set of keywords contained in the query).

- From patterns to other patterns.

  After identifying a useful design pattern and applying the solution proposed by the pattern, one would be interested in knowing how does this impact the overall design process and what consequences are triggered. Moreover, the application of one pattern might ask for the application of a related pattern, as well. The tool would support the identification of such related patterns as well as the consequences triggered by each pattern applied.

- From patterns to relationships.

  A designer might use two different patterns, possibly in different parts of the same project. S/he would want to know if these patterns are in some way related and, if so, how they are related. Specific relationships between patterns might trigger modifications in other parts of the project, as well. For example, a pattern p_i as a specialization of another pattern p_j might ask for the consideration of both its forces and the forces of pattern p_j.

- From the language to its graph.

  Not being familiar with the domain addressed by a pattern language, one would like to explore it for getting insight on the issues to be faced during design processes within such a domain. This would translate into browsing the patterns included in the language and exploring the relationships between them. For languages containing a smaller number of patterns this is not necessarily an issue. However, even in these cases and mostly in the cases of larger languages, the graphical representation of the pattern language would prove useful.

5.6.3 Design Considerations

After clarifying the high-level requirements and the audience of the tool, I moved on to considering several design decisions and goals. These considerations constitute a first iteration of the tool, able to bring to light additional aspects and features.

5.6.3.1 Data Representation

For each of the data sets used throughout the process, I considered a specific representation protocol to support storing the data sets in files.

- D:

d_i *<newline>*, where d_i is the unique identifier of the i^{th} design issue.

- P:
  p_i *<newline>*, where p_i is the unique identifier of the i^{th} design pattern.

- DIM 5.5:
  d_i *<space>* k_{ij} *<newline>*, where d_i is the unique identifier of the i^{th} design issue and k_{ij} is the j^{th} keyword associated to it.

- K 5.3:
  k_i *<newline>*, where k_i is the i^{th} keyword in the set.

- KM 5.4:
  k_{iu} *<space>* k_{jv} *<space>* R *<newline>*, where k_{iu} is the u^{th} keyword associated to the i^{th} design issue, k_{jv} is the v^{th} keyword associated to the j^{th} design issue, and R is the relationships between them.

- DIL 5.5:
  **nodes*
  d_i *<newline>*
  ...
  $*edges$
  d_i *<space>* d_j *<space>* R *<newline>*
  ...
  , where *nodes marks the beginning of the nodes representation (i.e. the design issues), and *edges marks the beginning of the edges representation (i.e. tuples comprising the identifiers of pairs of design issues together with the relationship identified between them).

- PL 5.6:
  **nodes*
  p_i *<newline>*
  ...
  $\ast$*edges*
  p_i *<space>* p_j *<space>* R *<newline>*
  ...
  , where *nodes marks the beginning of the nodes representation (i.e. the patterns), and *edges marks the beginning of the edges representation (i.e. tuples comprising the identifiers of pairs of patterns together with the relationship identified between them).

5.6.3.2 Pattern Language Generation

The algorithm for the design issue language generation is described below:

```
foreach $d_i \in D$
  foreach $d_j \in D$
  {
    foreach $k_i u \in K_i$
      foreach $k_j v \in K_j$
        if (i != j)
        switch (R)
        {
          "=": addEdge($d_i$, $d_j$, =); break;
          "≡": addEdge($d_i$, $d_j$, ≡); break;
          "ISA": addEdge($d_i$, $d_j$, ISA); break;
          "HASA": addEdge($d_i$, $d_j$, HASA); break;
          "RELATEDTO": addEdge($d_i$, $d_j$, RELATEDTO); break;
        }
    }
```

The algorithm parses the set D, pairing any two design issues and identifying whether there is a pair of keywords (k_{iu}, k_{jv}) such that:

- One of the design issues is associated to k_{iu}.
- The other design issues is associated to k_{jv}.
- The two keywords are related.

All the tuples of the form (d_i, d_j, R) are stored, following to be parsed so that for each pair (d_i, d_j) the strongest identified relationship is considered for the graphical representation. The generation of the pattern language will exclude those design issues not considered for being documented as patterns and the pairs (d_i, d_j), involving such design issues. The detour via the design issues is motivated by two aspects. First, design issues (even those not documented through patterns) contain valuable design information and some of them are related to those which get documented through patterns. Hence, ignoring them would lead to loosing potential valuable information. Secondly, some design issues might, in time, be considered for being documented as patterns since design, in general, is quite flexible and constantly pushed further by innovation and creative ideas.

5.6.3.3 Pattern Language Querying

The six types of queries supported by the tool are briefly described in Section 5.6.1. The input and output parameters for each of the queries together with the short description of the algorithm which answers the query are described below:

1. *Given the unique identifier of a design pattern, it returns the set of design patterns related to it, specifying for each pattern the type of relationship.*

    - Input.
      (p_α), where p_α is the unique identifier of a design pattern.

- Output.
  (p_α, p_β, R, R.description)*, where p_α is given as input, and p_β is the identifier of a pattern related to the input pattern by the relationship R. In case R is of type RELATEDTO, the R.description provides information on the relationship type.

- Algorithm.
  The algorithm parses the structure representing PL 5.6 and returns those pairs of edges in which one of the patterns is identified by the input identifier p_α. In case no such pair is identified, the result returned is void.

2. *Given the unique identifiers of 2 design patterns, it returns the type(s) of relationship(s) existing between them.*

- Input.
  (p_α, p_β), where p_α and p_β are the identifiers of the two design patterns.

- Output.
  (p_α, p_β, R, R.description)*, where p_α and p_β are given as input, and R is the relationship existing between them. In case R is of type RELATEDTO, the R.description provides information on the relationship type.

  The output consists in all the tuples of the form described above, representing all derived relationships between the two patterns provided as input. Even if the strongest relationship gets graphically represented within the pattern language, additional relationships could provide relevant information on the applicability of the two patterns.

- Algorithm.
  The algorithms parses the structure representing PL 5.6 and returns the edges defined by the two input identifiers together with the type of relation-

ship existing between them. In case the two patterns are not related (i.e. no edge defined by their identifiers exists in PL 5.6), the returned result is void.

3. *Given a type of relationship (R), it outputs the pairs of design patterns between which R exists.*

   - Input.
     (R), where R is a type of relationship.

   - Output.
     $(p_\alpha, p_\beta)^*$, where p_α and p_β are related by relationship R, provided as input.

   - Algorithm.
     The algorithm parses the structure representing PL 5.6, returning all the pairs of patterns related by a relationship of type R.

4. *Given a keyword, it returns all the design patterns associated with that keyword.*

   - Input.
     (k), where k is a keyword.

   - Output.
     $(p_\alpha)^*$, where p_α is the identifier of the pattern to which the keyword provided as input is associated.

   - Algorithm.
     The algorithm parses the structure DIM 5.2 and checks for each design issue: a) if it is a pattern and b) if the input keyword is associated to it. The output result consists in the identifiers of those patterns to which the input keyword is associated. In case the keyword is not associated to any

patterns, the returned result is void.

5. *Given 2 design patterns, it returns the set of keywords associated to both patterns.*

   - Input.
     (p_α, p_β), where p_α and p_β are the identifiers of the two design patterns.

   - Output.
     $(k_i)^*$, where k_i is associated to the patterns given as input.

   - Algorithm.
     The algorithms parses the structure DIM 5.2, identifying the set of keywords associated to the input patterns. The two sets are further parsed in order to identify their common elements. In case the two patterns have no words in common, the returned result is void.

6. *Given a set of keywords, it returns the collection of design patterns related to the keywords.*

   - Input.
     $(k_i)^*$, where $(k_i)^*$ is a set of keywords.

   - Output.
     (p_α, k_i, R, R.description)*, where p_α is the identifier of the pattern related to one of the keywords given as input, and k_i is associated to the pattern p_α and related by relationship R to one of the keywords given as input. In case R is of type RELATEDTO, the R.description provides information on the relationship type.

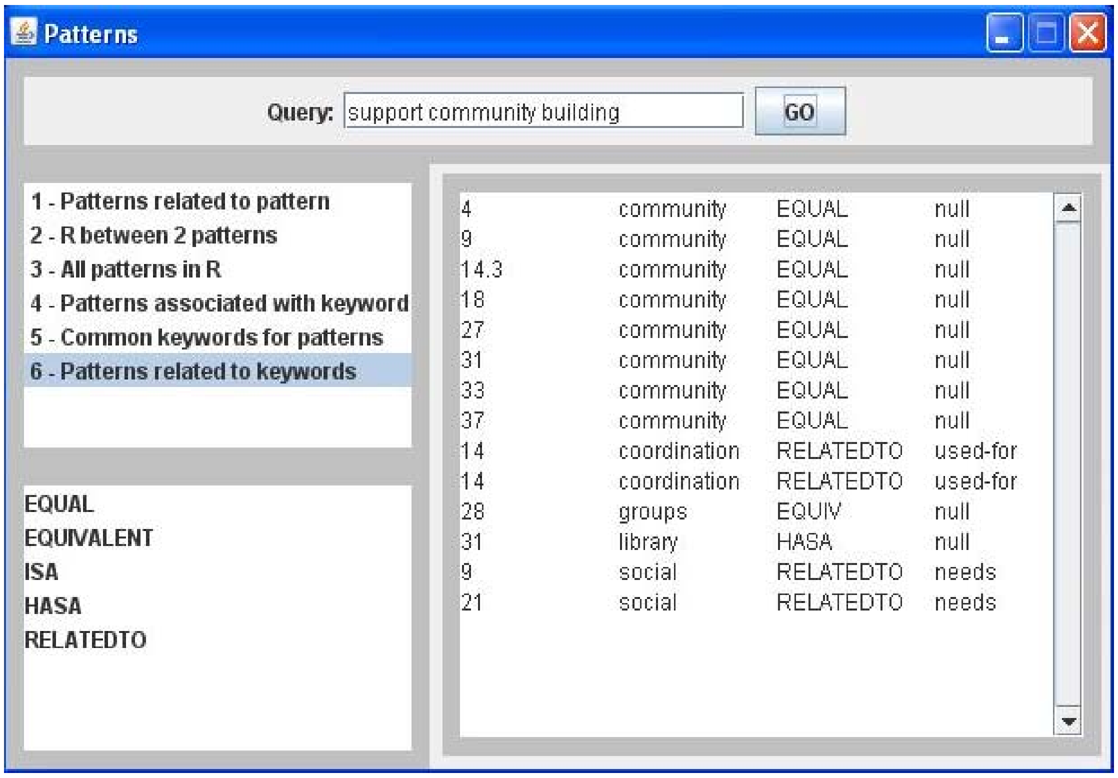
- Algorithm.
  The algorithm parses the structure DIM 5.2 and checks for each design issue: a) if it is a pattern and b) if any of the input keywords is associated to it. All the direct associations are added to the result set (due to the average size of pattern collections, efficiency concerns are not an issue at this point). Further on, for each of the input keywords associated to any of the patterns, the structure representing KM 5.4 is parsed in order to identify keywords related to them. The keywords related to the input ones are looked for in associations with other patterns. In case such associations are identified, the patterns are indirectly related to the input keywords. All the indirect relationships are also added to the result set. In case none of the keywords are related to any of the patterns, the returned result is void. A detailed description of the ”associate-to” and ”related-to” concepts is provided in Section 5.3.4.1.

5.6.3.4 Graphical User Interface

The tool's interface (Figure 5.7) provides a simple *query editor* allowing the user to input his query (i.e. a set of keywords or the identifiers of one or more design patterns). Keywords or identifiers are separated by spaces similarly to the input mode of search engines.

An area containing the possible types of queries the user might be interested in is also provided - the *query option* area. The user selects the type of query and inputs the keywords/identifiers for the query to be run. In case the text input from the user (provided in the *query editor*) does not match the type of query selected, an error message is displayed on the screen.

For the third type of query, the user is required to input a type of relationship existing between patterns. For that, the tool provides an area containing the possible relationships existing between patterns. For such a query to be executed, the user needs to simply select the type of query (in the *query option* area) and a type of relationship.



Query: support community building

1 - Patterns related to pattern
2 - R between 2 patterns
3 - All patterns in R
4 - Patterns associated with keyword
5 - Common keywords for patterns
6 - Patterns related to keywords

EQUAL
EQUIVALENT
ISA
HASA
RELATEDTO

4	community	EQUAL	null
9	community	EQUAL	null
14.3	community	EQUAL	null
18	community	EQUAL	null
27	community	EQUAL	null
31	community	EQUAL	null
33	community	EQUAL	null
37	community	EQUAL	null
14	coordination	RELATEDTO	used-for
14	coordination	RELATEDTO	used-for
28	groups	EQUIV	null
31	library	HASA	null
9	social	RELATEDTO	needs
21	social	RELATEDTO	needs

Figure 5.7: GUI of the tool supporting in querying the pattern language obtained as result of applying the pattern language generation method

Results are displayed in an designated area - the *results of the query* area. For now, the results are displayed according to the output protocols described in Section 5.6.3.3. Improvements in the result presentation are an open issue to be considered in a second iteration in the development of the tool. In the example depicted in Figure 5.7, the user runs a query of type 6, asking for the patterns related to a set of keywords provided as input. The keywords in the example are "support", "community" and "building". The results include the patterns directly associated to each of the keywords provided (the first 8 entries) and the patterns related to each of the keywords provided. For example, "groups" is a synonym of "community", and the patterns identified as 28 is associated with the keyword "groups"; hence the pattern identified as 28 is in an *Equivalence* relationship with the keyword "community".

5.6.4 Testing

In order to test the features of the above described application, I run the pattern language generation feature on two existing collections of design patterns:

1. The collection described in Section 4.4, addressing synchronous collaboration.
2. The collection provided by Jennifer Tidwell on web application design [103].

Surely, more test cases would provide stronger evidence of the tool's strengths, but for the purpose of this work the two collections listed above are considered. The motivation for choosing the two is two-fold. On one hand, the collections address two different domains and include patterns addressing different communities. This aims at showing that the tool (and the method underlying it) is independent of the domain addressed by the collection used as input. On the other hand, the two collections are different in size, the second one being much larger. This aims at proving that the tool (and the method underlying it) is scalable, hence it can be used for larger collections as well. The tests aimed at investigating the following:

- The application of the method described above in this chapter.

- The generation algorithm for the pattern language generation.
- The querying algorithms on the generated pattern languages.

5.6.4.1 Case 1: Synchronous Collaboration

I started from the data presented in Section 5.4 and used it as input of the *Pattern Language Generation* phase. As intermediary result, Figure 5.8 illustrates the graph representation of the design issues (represented by their identifiers), the set of keywords, and both the DIM 5.2 (set of pairs mapping a design issue to a keyword) and KM 5.4 (set of pairs mapping two keywords) sets. Different colored arrows represent different types of relationships. The graphical representation was made through Medusa[1], an interaction graph viewer and editor described in more details in Section 5.2.

The *Pattern Language Generation* phase had as output the PL 5.6, depicted in Figure 5.9. The output PL 5.6 is described according to the protocol used by Medusa for input files, allowing the visual representation of the language as a directed graph. Each of the nodes in the graph are associated to the identifier of a design pattern and each directed edge links two related patterns. Different colours indicate different types of relationships, according to the legend provided. Surely, the set of relationships between the patterns may be extended providing that new relationships - not identified by the tool - are spotted once the language is used. For validating the relationships outputted by the tool, I walked them all through trying to identify the extent in which the patterns are related and compare this with the output of the tool. For example, the pattern Support versioning ISA pattern Track history of collaboration since the history of a collaborative process is kept through a versioning system. Moreover, the pattern Integrated chat is associated to the pattern Who is the coordinator? since an integrated chat mechanism can be used for the coordination of a collaborating group.

[1]Available at: http://sourceforge.net/projects/graph-medusa/

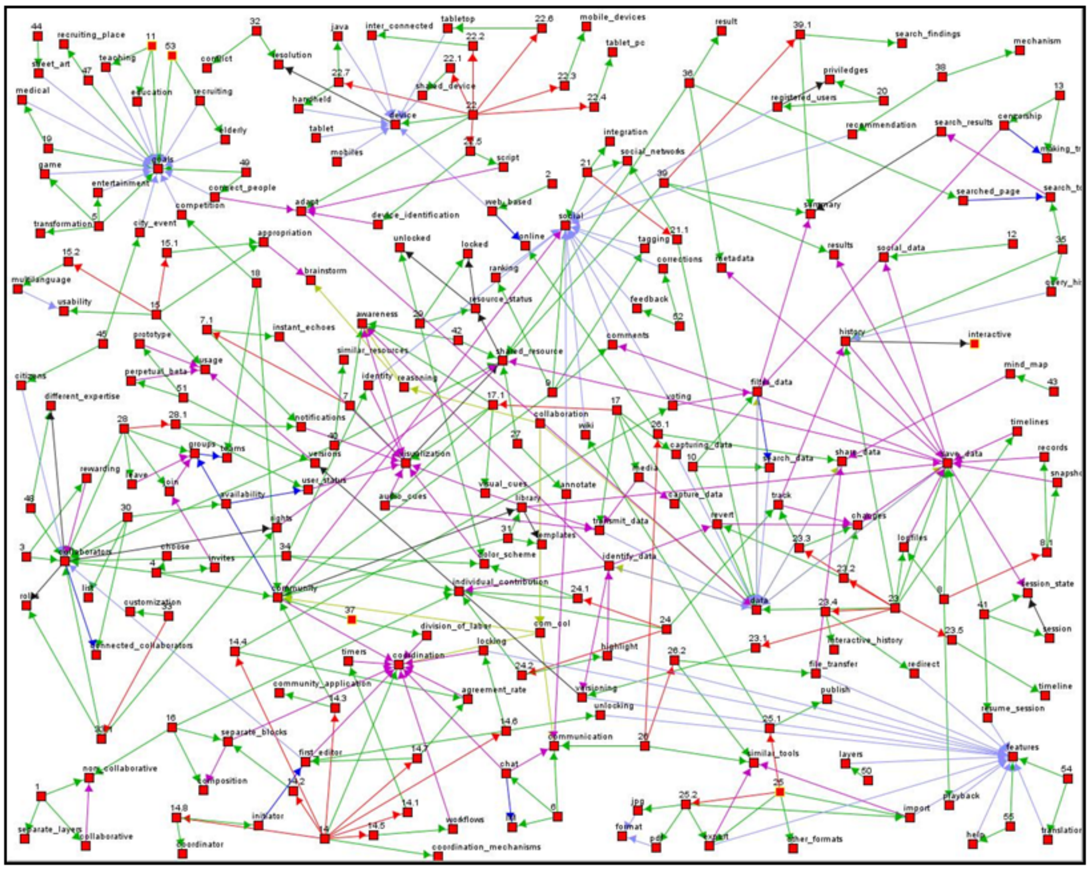


Figure 5.8: A glimpse of the method application - design issues, keywords, the DIM and KM sets

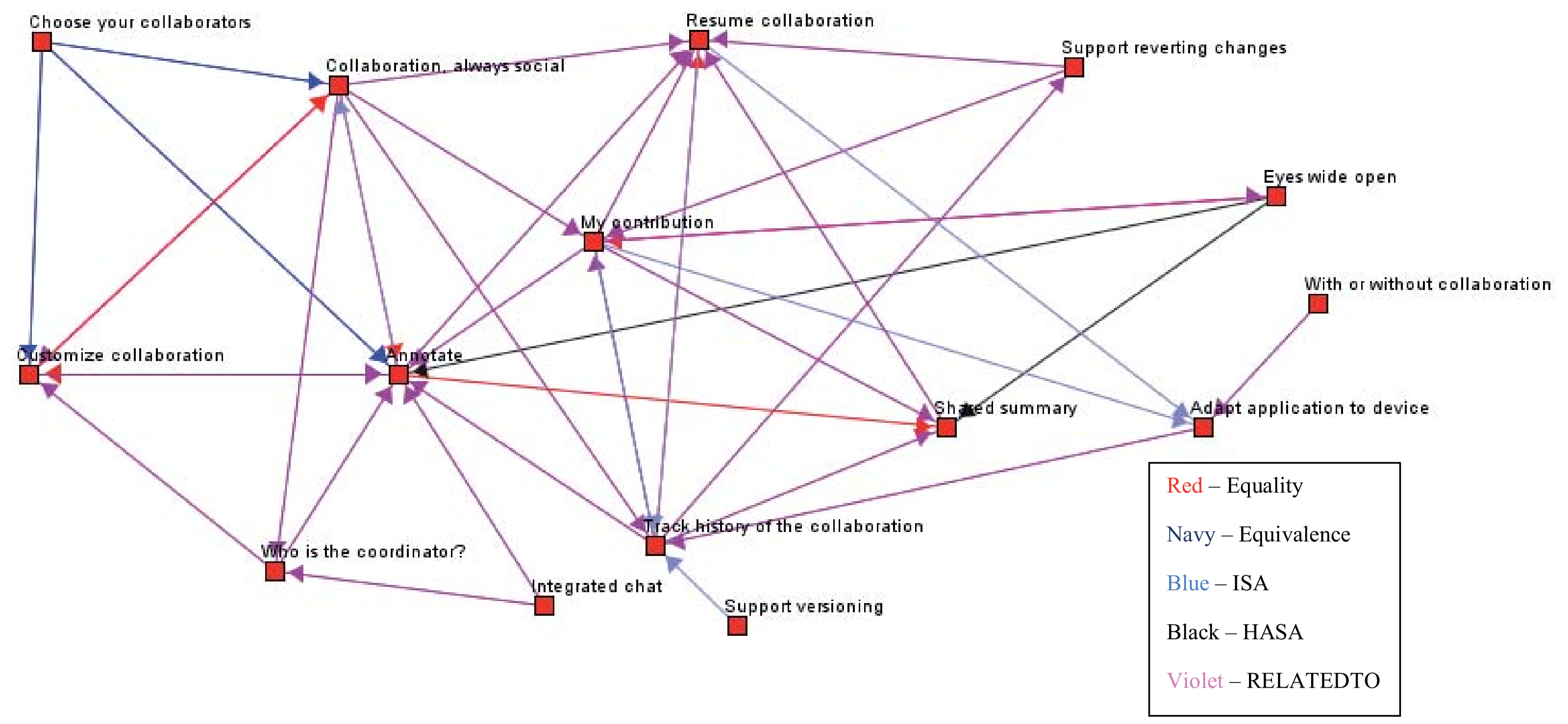


Figure 5.9: A pattern language for the design of synchronous collaborative systems

Design issue	Keywords
1 (Two-Panel Selector)	*two_panel_selector*, *selectable_lists*, *easy_access*
2 (One-Window Drilldown)	*one_window*, *option_menu*, *space_navigation*
3 (Wizard)	*wizard*, *ordered_navigation*, *branched_tasks*, *chunking_the_task*, *physical_structure*
4 (Extras On Demand)	*on_demand*, *hide_features*, *simplification*, *dropdown*
5 (Intriguing Branches)	*additional_branches*, *original_flow*, *clear_return*, *resume_task*

Table 5.2: Design Issues Map Web Design Case (fragment)

5.6.4.2 Case 2: GUI Design

As a second test case, I considered the collection of design patterns proposed by Jenifer Tidwell and made available online at: http://designinginterfaces.com/firstedition/. I started by assuming that the 44 design patterns made available represent the set of design issues, D which describes the domain targeted by the collection - interaction design of web applications. That is to say that I considered the set P of design patterns to contain the same elements as the set D of design issues, hence all the potentially identified design issues were documented through patterns. This simplification does not reduce in any way the workflow of application of the method. It is solely meant to allow the focus to be placed on the pattern language generation.

I associated each design issue/pattern with a set of keywords. For these associations I used the documentation of each pattern. First, I used the name of the pattern as a keyword and I added keywords relevant to the descriptions of the problem and the solution as proposed by Tidwell. The number of keywords associated to each pattern varied between 2 and 5. In addition to that, I considered a keyword as such only if it appeared in less than 10% of the associations. A fragment of the DIM 5.2 is described in Table 5.2, the full structure being presented in Appendix .3.

All the keywords used in the associations belong to the set K 5.3, which is entirely presented in Appendix .3. Further on, I identified relationships between the elements of the set K 5.3. Some of these relationships are described below, the full KM 5.4 structure being presented in Appendix .3.

- *wizard* = *wizard*,
- *hide_features* ≡ *hide_flow*, *distinguishing_sections* ≡ *separate_sections*, *magnified_projection* ≡ *zoomed_area*,
- *ordered_navigation* ISA *space_navigation*, *alternate_shades* ISA *color_codes*, *constrained_resize* ISA *resizing*,
- *resizing* HASA *resize_modes*, *single_page* HASA *physical_structure*, *unfolding_task* HASA *additional_branches*, *ordered_navigation* HASA *transition*,
- *initiate_navigation* $RELATEDTO_{applies-to}$ *space_navigation*, *jump_to_item* $RELATEDTO_{supports}$ *complex_value_selection*, *row_striping* $RELATEDTO_{triggers}$ *chuncking_the_content*, *command_history* $RELATEDTO_{includes}$ *initiate_navigation*, *memorable_names* $RELATEDTO_{used-for}$ *label_actions*.

Taking as input the DIM 5.2 and the KM 5.4 (in this case, D = P), the tool outputs the pattern language depicted in Figure 5.10. Since D = P, DIL 5.5 is the same as PL 5.6. Similar to the previous test case, the set of relationships between the patterns may be extended providing that new relationships - not identified by the tool - are spotted once the language is used. For validating the relationships outputted by the tool, I walked them all through trying to identify the extent in which the patterns are related and compare this with the output of the tool. For example, the pattern Good Defaults ISA Input Hints, since intuitive default input values are subtypes of input hints. Moreover, the pattern Action Panel HASA Progress Indicator, since indicating the progress of an action is part of any executing action interaction display (each action in the panel is associated with a progress indicator once executed).

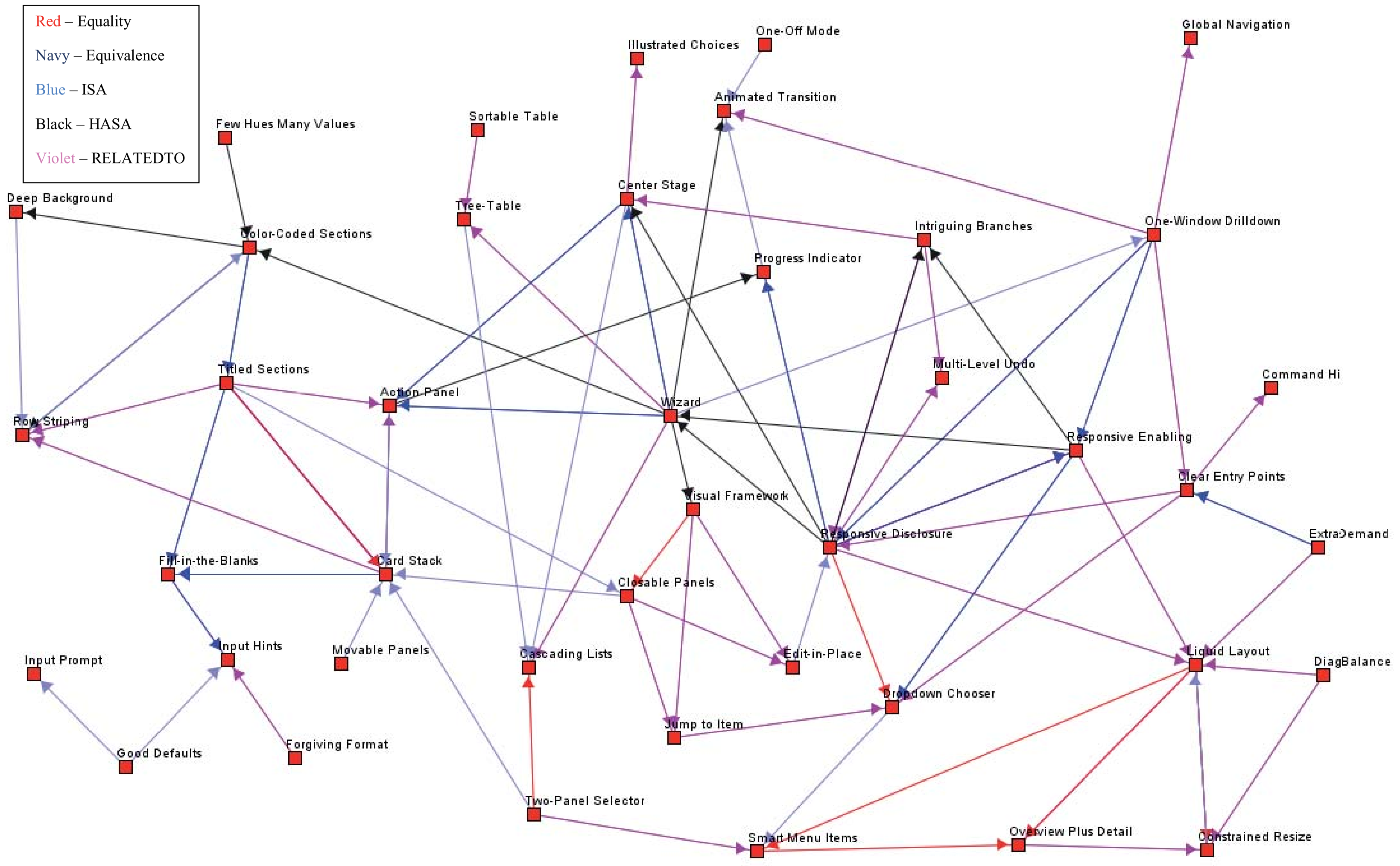


Figure 5.10: A pattern language for the design of GUIs for web systems

5.6.5 Open Points and Ideas for the Next Iteration

After the first iteration of the support tool for generating and querying pattern languages, several open issues and improvement ideas became clear.

1. Additional support in automatizing the initial steps of the process. Associating each design issue with a set of keywords and then identifying relationships between these keywords requires human intervention. However, as discussed in Section 5.5, several techniques can be investigated for improving this and supporting further automation of these steps.

2. More intuitive GUI features in the query results presentation. For now, the tool's interface is quite simple and it is meant as proof-of-concept. Further developments are needed for the GUI to be user-friendly and support a more efficient exploration of the query results.

3. Support in browsing the graphical representation of the PL 5.6. The pattern language is represented as a graph which may be further browsed. For now, the tool exports the PL in the format supported by Medusa and the graph can be browsed solely through Medusa. Investigating ways to support such processes is much needed.

4. Generating pattern languages from other existing collections of patterns. Two collections have been test subjects for the tool. However, more tests are needed, hence other collections of patterns available today could be the input of the tool. This would support both testing the limitations of the tool and helping find room for improvements or additional needed features not yet included.

—

In this chapter, I introduce a method to be used for generating pattern language structures out of collections of design patterns. This comes as an answer to the lack of methodological support in the area and aims at supporting both pattern authors interested in generating pattern languages out of the collections they wrote and pattern users interested in retrieving and browsing knowledge captured by such collections.

The major *strength* of the method relies in supporting the semi-automation of such a generation process. Human intervention is still needed, but the process is largely automated. In addition to that, it is supported by a tool able to help guide through the method's application. This tool acts as a pattern language generator and as a search engine localized on the pattern language, supporting the execution of queries on it.

A *limitation* now is that the method and the tool supporting its application need to be strengthened by several other evaluation cycles. The two test cases described in this chapter address two collections documenting different domain (synchronous collaboration and GUI design), but other test cases would strengthen and validate the applicability of the method. Also, the tool's implementation requires a more user-friendly GUI and possibly some accompanying usability tests.

Chapter 6

Evaluating Patterns: Impact and Strategies in the Collaborative Use of Design Patterns

The usefulness of a knowledge repository represented as a collection of design patterns is largely recognized [31], [35], [52], but little work has been done in investigating and measuring the impact such a collection has on collaborative design processes involving designers. An overview of the documented investigations is described in Section 2.2.6 and includes examples of patterns being used in teaching, design, and in particular participatory design. The aim of my work is to understand how design patterns are used in collaborative design processes. An initial step towards such understanding is the case study described in this chapter, involving novice designers.

6.1 Objectives and Rationale

The overall objective of the case study is the evaluation of the concept of design pattern in the context of collaborative design processes. Given a collection of design patterns, *how would a team of designers make use of this collection* and *what would be the impact of this use*? This is, in other words, the question this chapter addresses. A first step consists in involving novice designers - students in Computer Science - and have them use the collection of patterns described in Chapter 4. The participants are grouped in teams and, being provided with the patterns, are asked to use them in performing a simple GUI design task. Even if strong conclusions with respect to generalizations of the use of design patterns in collaborative design processes ask for more empirical work, the present efforts provide a starting point for further understanding and investigation.

In designing the case study, I initially addressed the question of understandability and tried to evaluate in what measure the concept and the content of the design patterns are easily grasped by novice designers. A low degree of understandability would strongly bias the results of an investigation on usefulness, since not understanding the rationale of the concept would lead to either not using the concept at all or using it without taking advantage of its full capacities.

Having ensured the fact that the patterns presented do not impose any understandability constraints, I further considered addressing the issue of usage and investigate what actions do novice designers perform on the collection of patterns. I started with a pool of actions documented in the literature as being subject to design pattern support - browsing the collection, searching for a documented problem, analysing and applying a solution proposed by the patterns - and analysed the degree in which each of these activities is performed, and whether there are other common actions participants perform.

An additional decision with respect to the design of the case study was the form of representing the patterns. Considering that the collection provided is rather small (15 patterns) and that the participants work together around a table, the decision on the representation form to use consisted in paper cards. However, after observing the use of the patterns represented as paper cards, I addressed the issue of the feasibility of this type of representation and of the possible alternatives.

The case study also aims at getting some insight on the overall picture, trying to get feedback from the participants on their experience using the patterns. The interest at this point is in understanding how the patterns made a difference from the point of view of the (novice) designers. These results are further correlated with the analysis of the dialogues and the interactions held during the design task in order to assess the connection between the designers' feedback and the course of actions performed throughout the task.

Lastly, I aimed at looking into abstracting strategies participants develop while working with the patterns. The rationale for this relies in understanding the processes followed by designers while working with patterns in experimental contexts of collaborative design processes and in deriving the requirements for tools to support the use of design patterns in real-world similar contexts.

To summarize, I considered addressing the following research questions:

1. Are the format and the content of an existing collection of design patterns easy to understand for novice software designers?

2. Having available a collection of design patterns targeting a design area, what are the actions mostly performed by novice designers while collaboratively using them?

3. What pattern representation would best fit for working with a collection of design patterns?

4. What is the overall impact of using design patterns in collaborative design processes?

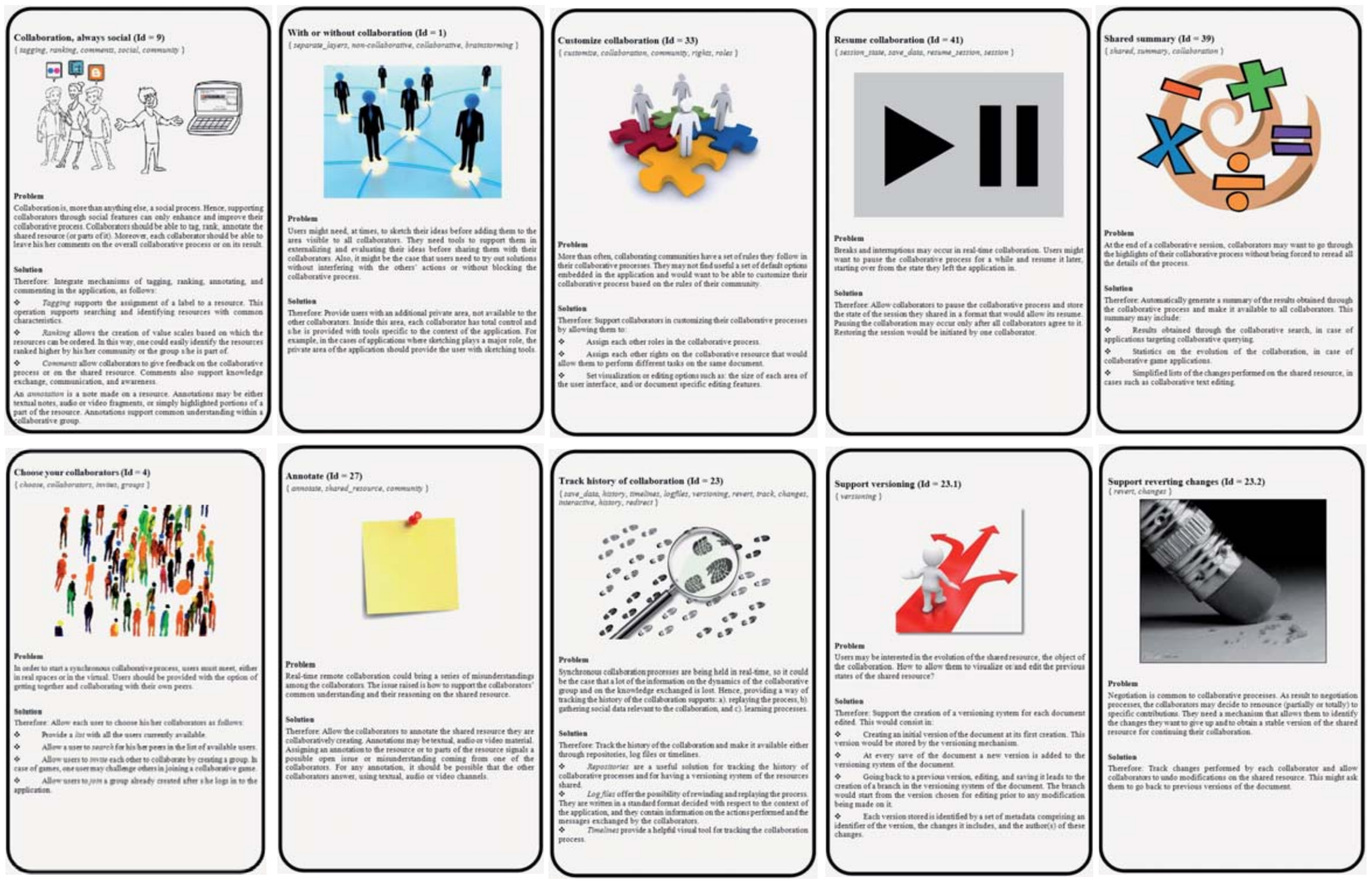
5. What strategies do novice designers develop in working with a collection of design patterns?

6.2 Method

6.2.1 Procedure

The case study was organized in the form of a series of design workshops, each workshop bringing into the lab one team of novice designers. Each such workshop lasted for 2 hours, one facilitator being present each time. The facilitator's role was to: a). introduce the participants to the workshop, b). walk them through each phase of the workshop, c). take notes of their interactions, and d). observe them throughout the workshop and support them if needed.

Each team was presented with a brief overview of the goals of the workshop and with the collection of the 15 patterns described above. Each pattern was represented on a paper card, being described by its name, its unique ID, the set of keywords associated to it, a representative illustration, the problem addressed by the pattern, and the solution proposed to tackle the problem (Figure 6.1). The restrictive description was mainly enforced by the paper card representation of the patterns and by the time length of each workshop.



Collaboration, always social (Id = 9)
{ *tagging, ranking, comments, social, community* }

Problem
Collaboration is, more than anything else, a social process. Hence, supporting collaborators through social features can only enhance and improve their collaborative process. Collaborators should be able to tag, rank, annotate the shared resource (or parts of it). Moreover, each collaborator should be able to leave his/her comments on the overall collaborative process or on its result.

Solution
Therefore: Integrate mechanisms of tagging, ranking, annotating, and commenting in the application, as follows:
❖ *Tagging* supports the assignment of a label to a resource. This operation supports searching and identifying resources with common characteristics.
❖ *Ranking* allows the creation of value scales based on which the resources can be ordered. In this way, one could easily identify the resources ranked higher by his/her community or the group s/he is part of.
❖ *Comments* allow collaborators to give feedback on the collaborative process or on the shared resource. Comments also support knowledge exchange, communication, and awareness.
An *annotation* is a note made on a resource. Annotations may be either textual notes, audio or video fragments, or simply highlighted portions of a part of the resource. Annotations support common understanding within a collaborative group.

With or without collaboration (Id = 1)
{ *separate_layers, non-collaborative, collaborative, brainstorming* }

Problem
Users might need, at times, to sketch their ideas before adding them to the area visible to all collaborators. They need tools to support them in externalizing and evaluating their ideas before sharing them with their collaborators. Also, it might be the case that users need to try out solutions without interfering with the others' actions or without blocking the collaborative process.

Solution
Therefore: Provide users with an additional private area, not available to the other collaborators. Inside this area, each collaborator has total control and s/he is provided with tools specific to the context of the application. For example, in the cases of applications where sketching plays a major role, the private area of the application should provide the user with sketching tools.

Customize collaboration (Id = 33)
{ *customize, collaboration, community, rights, roles* }

Problem
More than often, collaborating communities have a set of rules they follow in their collaborative processes. They may not find useful a set of default options embedded in the application and would want to be able to customize their collaborative process based on the rules of their community.

Solution
Therefore: Support collaborators in customizing their collaborative processes by allowing them to:
❖ Assign each other roles in the collaborative process.
❖ Assign each other rights on the collaborative resource that would allow them to perform different tasks on the same document.
❖ Set visualization or editing options such as: the size of each area of the user interface, and/or document specific editing features.

Resume collaboration (Id = 41)
{ *session_state, save_data, resume_session, session* }

Problem
Breaks and interruptions may occur in real-time collaboration. Users might want to pause the collaborative process for a while and resume it later, starting over from the state they left the application in.

Solution
Therefore: Allow collaborators to pause the collaborative process and store the state of the session they shared in a format that would allow its resume. Pausing the collaboration may occur only after all collaborators agree to it. Restoring the session would be initiated by one collaborator.

Shared summary (Id = 39)
{ *shared, summary, collaboration* }

Problem
At the end of a collaborative session, collaborators may want to go through the highlights of their collaborative process without being forced to reread all the details of the process.

Solution
Therefore: Automatically generate a summary of the results obtained through the collaborative process and make it available to all collaborators. This summary may include:
❖ Results obtained through the collaborative search, in case of applications targeting collaborative querying.
❖ Statistics on the evolution of the collaboration, in case of collaborative game applications.
❖ Simplified lists of the changes performed on the shared resource, in cases such as collaborative text editing.

Choose your collaborators (Id = 4)
{ *choose, collaborators, invites, groups* }

Problem
In order to start a synchronous collaborative process, users must meet, either in real spaces or in the virtual. Users should be provided with the option of getting together and collaborating with their own peers.

Solution
Therefore: Allow each user to choose his/her collaborators as follows:
❖ Provide a *list* with all the users currently available.
❖ Allow a user to *search* for his/her peers in the list of available users.
❖ Allow users to *invite* each other to collaborate by creating a group. In case of games, one user may challenge others in joining a collaborative game.
❖ Allow users to *join* a group already created after s/he logs in to the application.

Annotate (Id = 27)
{ *annotate, shared_resource, community* }

Problem
Real-time remote collaboration could bring a series of misunderstandings among the collaborators. The issue raised is how to support the collaborators' common understanding and their reasoning on the shared resource.

Solution
Therefore: Allow the collaborators to annotate the shared resource they are collaboratively creating. Annotations may be textual, audio or video material. Assigning an annotation to the resource or to parts of the resource signals a possible open issue or misunderstanding coming from one of the collaborators. For any annotation, it should be possible that the other collaborators answer, using textual, audio or video channels.

Track history of collaboration (Id = 23)
{ *save_data, history, timelines, logfiles, versioning, revert, track, changes, interactive, history, redirect* }

Problem
Synchronous collaboration processes are being held in real-time, so it could be the case that a lot of the information on the dynamics of the collaborative group and on the knowledge exchanged is lost. Hence, providing a way of tracking the history of the collaboration supports: a). replaying the process, b). gathering social data relevant to the collaboration, and c). learning processes.

Solution
Therefore: Track the history of the collaboration and make it available either through repositories, log files or timelines.
❖ *Repositories* are a useful solution for tracking the history of collaborative processes and for having a versioning system of the resources shared.
❖ *Log files* offer the possibility of rewinding and replaying the process. They are written in a standard format decided with respect to the context of the application, and they contain information on the actions performed and the messages exchanged by the collaborators.
❖ *Timelines* provide a helpful visual tool for tracking the collaboration process.

Support versioning (Id = 23.1)
{ *versioning* }

Problem
Users may be interested in the evolution of the shared resource, the object of the collaboration. How to allow them to visualize or/and edit the previous states of the shared resource?

Solution
Therefore: Support the creation of a versioning system for each document edited. This would consist in:
❖ Creating an initial version of the document at its first creation. This version would be stored by the versioning mechanism.
❖ At every save of the document a new version is added to the versioning system of the document.
❖ Going back to a previous version, editing, and saving it leads to the creation of a branch in the versioning system of the document. The branch would start from the version chosen for editing prior to any modification being made on it.
❖ Each version stored is identified by a set of metadata comprising an identifier of the version, the changes it includes, and the author(s) of these changes.

Support reverting changes (Id = 23.2)
{ *revert, changes* }

Problem
Negotiation is common to collaborative processes. As result to negotiation processes, the collaborators may decide to renounce (partially or totally) to specific contributions. They need a mechanism that allows them to identify the changes they want to give up and to obtain a stable version of the shared resource for continuing their collaboration.

Solution
Therefore: Track changes performed by each collaborator and allow collaborators to undo modifications on the shared resource. This might ask them to go back to previous versions of the document.

Figure 6.1: Evaluation workshops - Paper card representation of patterns

The initial phase of each workshop asked the participants to go through the patterns and to get familiar with them. No strategy was suggested, all of the teams being free to follow their own approach for looking the patterns over. All the misunderstandings or unclear issues were discussed with the facilitator and collected for further analysis. As a second phase, each team was presented with a list of problems and was encouraged to choose one problem for which to design, using the patterns provided, a software application. The problems addressed different areas of synchronous collaboration, such as collaborative drawing, collaborative text editing, collaborative game solving, or collaborative search.

The designs were meant to consider solely the GUI and the interaction process of the application. To support their design processes, the teams were encouraged to answer the following questions: a) who are the users targeted?, b) what is the motivation of the users' collaboration?, c) when and where can the application be used?, and d) how can the users interact with the application? [28]. Also, they were asked to sketch their ideas, express all the design problems they encounter and, possibly, create a mock up of their overall design. Their conversations were recorded and a facilitator observed all of the teams, taking notes of their interactions. Lastly, each participant answered a questionnaire providing feedback on the overall process followed and on the use of the patterns. The complete questionnaire is provided in Appendix .6.

6.2.2 Problems

The list of problems proposed during the workshops included collaborative drawing, collaborative text editing, collaborative search, and collaborative game solving.

The problem of collaborative drawing asked for the design of a software application which would allow painters, graphic designers and/or visual artists to collaboratively create one diagrammatic representation. The problem of collaborative text editing required participants to design an application which would allow a group of users to create a summary of a written text in a synchronous collaborative fashion. The set of games considered for collaborative solving consisted of puzzles and crosswords. The common requirement for both was that more users solve one game in the same time. The problem of collaborative search required that more users are able to perform one web-search from remote locations.

6.2.3 Participants

The total number of participants was 75, out of which 75% were male, and 25% were female[1]. 66 of them (88%) were first year students in Computer Science, 8 (11%) were following their second year, while 1 of them was a third year student in Design. Out of the 75 participants, only two have had prior experience with working with design patterns. Solely 9 of the participants (12%) had more than 3 years experience with designing software applications, while the rest of 88% were novice software designers, with less than 3 years experience in software design.

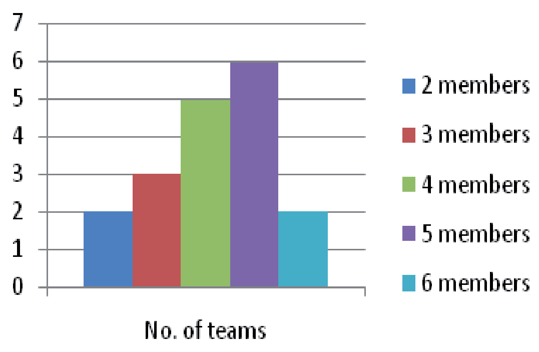


Figure 6.2: Evaluation workshops - Team member distribution

The 75 participants were divided into 18 teams with the member distribution of each team depicted in Figure 6.2. The majority of the teams were formed of 5 participants, while 2 of the teams were formed of 2 participants each. The differences between the number of participants in each team had little impact on the results obtained, a

[1]There could be gender differences at play in the results obtained; however, these differences are beyond the scope of this thesis to explore.

detailed analysis on that being provided further on in Section 6.3.

6.2.4 Measures

This evaluation process addresses the set of questions described in Section 6.1. For each of the questions, measures were defined and used to quantify the final results. These measures were derived for the following data collection sources:

- audio recordings of each team's conversations;
- notes taken by a facilitator present during each of the workshops;
- questionnaire .6 filled in by each participant at the end of the workshop (used for providing feedback on the process followed);
- results of the design task followed (annotated sketches);

1. *Are the format and the content of the existing collection of design patterns easy to understand for novice software designers?*

   Having used the patterns provided, the participants were asked to choose the most useful and the least useful pattern defining element (i.e. name, ID, keywords, picture, problem, solution) in understanding the patterns. Moreover, they were asked to order the pattern defining elements from the most useful element to the least useful element in supporting the understandability of the presented patterns. After having gone through all the patterns and having used them, the participants were asked to rate on a Likert-type scale (with 1 – not at all understandable, 2 – not understandable, 3 – I can't say, 4 – understandable, and 5 – very understandable) the degree of understandability of each of the patterns presented. The patterns were available to the participants throughout the rating. The average rate was computed for each of the patterns, and a global rate was calculated for identifying the overall understandability of the whole collection.

This measure helped in various ways. On the one hand, I identified those patterns which proved to be difficult to grasp and rewrote them based on the participants' feedback. On the other hand, the quantitative data collected allowed measuring the rate of the entire collection. It is safe to assume that participants went through all of the patterns, hence their rates are based on their experience browsing all of the patterns, since the collection provided was rather small. In the case of larger collections, the settings for such an evaluation would have to differ at least with respect to the time length of each workshop.

2. *Having available a collection of design patterns targeting a design area, how do novice designers make use of it during collaborative design processes?*

Through the questionnaire, the participants were asked to rate on a Likert-type scale (with 1 being not at all useful, 2 – not useful, 3 – I can't say, 4 – useful, and 5 - very useful) the degree of usefulness of the patterns for each of the following documented uses of patterns:

(a) understanding the design space of the application;

(b) searching for design problems;

(c) searching for solutions for already identified design problems;

(d) communicating with the other members of the team;

(e) remembering similar design situations encountered;

(f) brainstorming for design ideas for the application;

For each of the above activities, the average rate of the answers was computed. Moreover, the feedback from the participants tried to identify which of the above activities was mostly supported by each of the patterns.

The activities considered initially are the common documented uses of design patterns. It is for that reason that I considered them as starting point. The rationale behind this measure was understanding the degree in which designers perceive their involvement in each of these activities. In addition to that, subject to this evaluation is also the exploration of other activities and sequences of activities designers perform using the patterns (more on that to follow at 5).

3. *What pattern representation would best fit for working with a collection of design patterns?*

   After having used the patterns in their paper card representation, the participants were asked to choose which, in their opinion, would be the most suitable representation for a collection of patterns having as options the following:

   - paper cards;
   - wiki applications;
   - search engines;
   - specialized application with personalized features;

   Bearing in mind that the participants only used the patterns in their paper card representation, I was aiming to obtain feedback on the appropriateness of this form of representation as opposed to other possible forms. All of the participants were familiar with wikis, search engines, and they were explained what a specialized application for working with patterns would be (i.e. it would support the most common activities performed using the patterns; an example of such a tool is the one described in Chapter 5). However, their answers reflect their personal preference and are not yet supported by a comparative study for drawing further conclusions.

4. *What is the overall impact of using design patterns in collaborative design processes?*

   The overall impact of using the patterns was measured by:

(a) the Likert-type scale (with 1 being not at all useful, 2 – not useful, 3 – I can't say, 4 – useful, and 5 - very useful) ratings participants assigned for each of the patterns with respect to the usefulness of the pattern in the overall process

(b) the qualitative feedback provided by the participants as answer to the open ended question: "How have the patterns supported your design process?"

Rating each pattern allowed the computation of an overall rate for the entire collection. Such a value gets complemented by qualitative data coming from the open-ended question addressed to the participants. Overall, the goal at this point is to get a feeling of how the participants perceived the overall design task, and in particular, in what degree they felt the patterns made a difference.

5. *What strategies do novice designers develop in working with a collection of design patterns?*

The conversations of all the teams were recorded and transcribed. Their dialogues were divided into sentences (i.e. small fragments of dialogues – usually lines of the dialogues – related to a particular concept or action), all those sentences containing references to the patterns provided being filtered and considered for further analysis. The coding scheme used for coding the sentences referencing patterns classified these sentences as indicating :

(a) *Browsing* the collection - going through the patterns one by one.

(b) *Reading* a pattern - holding a card and reading the entire content of the patter described by it.

(c) *Using* a solution - explicitly applying the solution proposed by a patterns or applying the solution after studying the pattern, without explicitly indicating this fact.

(d) *Adapting* a pattern - adjusting the solution proposed by a pattern and applying the adapted solution in the given context.

(e) *Modifying* a pattern - changing the form of a pattern and making these changes explicit. For example, adding a new defining element for a pattern would count as modifying the pattern.

(f) *Searching* for a pattern - identifying a problem and searching for a pattern which addresses this problem.

(g) *Discussing* a pattern with other members of the team - explaining the elements of a pattern to each other, possibly making associations to similar situations met in other projects.

(h) *Referencing* a pattern - pointing a pattern or reminding of a pattern (either by its name, identifier or some keyword associated to the pattern).

(i) *Re-referencing* a pattern - coming back to a pattern already discussed for clarifying open points.

(j) *Generating* a design idea inspired by a pattern - coming up with a new design element (and incorporating it in the team's design) based on consulting a pattern.

The strategies used by the teams were abstracted from:

- the sequences of actions (i.e. those defined by the coding scheme) they followed in isolated contexts of their design processes; more on this in Section 6.3.3.2.
- the ratio of the sentences in each category over the total number of sentences considered; more on this in Section 6.3.3.1.
- the facilitator's notes on the participants' interactions; more on this in Section 6.3.1.

6.3 Results

6.3.1 Direct Observation

All of the teams were closely observed by a facilitator who took notes on their design processes. Before detailing the quantitative results obtained, I will briefly describe the process followed by each of the teams, according to the notes taken through direct observation.

6.3.1.1 Team no. 1 - Markers

The first team chose the problem of collaborative puzzle solving and started by reading together each pattern, one by one. Four of the patterns (i.e. Shared summary, Revert changes, Track history, Who's the coordinator) were not clear to the team and the facilitator discussed them in more details. Further on, the team assigned to each pattern a word characterizing its core idea. During their design process they referred to each pattern using the word associated to it. While going through the patterns, the team made no reference to the name of each pattern; they only paid attention to the text of the problem and the solution. Often, they tried to find examples of application for each pattern in order to better understand and clarify its idea and the way they can use it. In the initial stage of the design process, the team made a list with all the problems they were interested in addressing in the design of the application they were working on. Then, they went through the patterns in order to identify those which addressed the problems on the list. For each pattern they used, they marked its identifier on the sketch they were working on. Going back to previous steps of their process, they would remind themselves about the decisions taken at that step through the help of the pattern marked as used at that point.

6.3.1.2 Team no. 2 - Selectiveness

The second team chose the problem of collaborative puzzle solving. They read the patterns all together and, in the end, one member of the team went through all of them once again explaining to the others the main idea of each pattern. Three patterns were not clear, the team asking the facilitator to provide more details on these. The team initially ignored the patterns and began with setting the context for their design process (i.e. thinking about the target users and the main affordances of the application). Moreover, they did not consider necessary the use of all the patterns, motivating that they would only turn to them when faced with a difficult situation or when in search for new ideas. This is precisely what they did. First, they faced the problem of designing a mechanism to be used by the application for pointing out who placed each piece of the puzzle on the board. For the answer, they turned to the patterns and used one of them (i.e. My contribution). Further on, they faced the problem of more users accessing the same piece of the puzzle in the same time. They remembered that this was one of the issues addressed by the patterns and turned back to them. Towards the end of their design process, the team agreed to go through the entire collection again and check for ideas they were not considering until that point. As result of this walk through, they added to their design social features such as rankings.

6.3.1.3 Team no. 3 - Misunderstandings

The third team worked on the collaborative text editing problem. They initially went through the patterns, each one reading a subset of them. They moved on to the design of an application for the problem considered and initially ignored the patterns. When asked about them by the facilitator, the participants went through them all in order to check which ones they already considered and how the others were reflected in their work. They stopped at the pattern With or without collaboration and considered applying it only that the main idea of the pattern was misunderstood, the team being inclined to switch to an application to be used individually. It was only after the facilitator's intervention that the participants got back on track. The team continued their process with little influence from the patterns. It was only by the end of the workshop that they all went through the entire collection in order to evaluate which of the issues they considered and how.

6.3.1.4 Team no. 4 - Confirmations

The fourth team chose the problem of collaborative puzzle solving. They placed the patterns on the table and each of the members of the team randomly read some of the patterns. Being inspired by one of the patterns (i.e. With or without collaboration), they initiated their design with the idea of a private area of the application. Further on, all their process revolved around this idea. Faced with the problem of having the users of the application coordinate while using a shared resource (the puzzle board), the team turned to the patterns and identified the one dealing with this problem (i.e. Who is the coordinator?). It is only now that they asked the facilitator to provide them with more details on the pattern and on the solution proposed by it. After discussing all the options, the team agreed to apply one of the solutions proposed. There were several times when the team was not sure about what decisions to make. This made them search for confirmation from the patterns. One example is the situation where it was not clear how to point out who are those users trying to solve the puzzle at a given time. The team hinted a possible solution, but it was only after reading the pattern Choose your collaborators that they agreed on how to proceed and applied the solution proposed. The team went through the collection over and over again throughout all the process. They learned the patterns while going through them and trying to find ideas and solutions, as opposed to other teams which did not start the design process before all the members of the team got familiar with all the patterns. Browsing the collection was also a way to conclude and evaluate the completeness of their work - they made clear what patterns they used and in which way and why some of them remained unused.

6.3.1.5 Team no. 5 - Minimal use

The fifth team chose the problem of collaborative drawing. Each member of the team went through 3-4 patterns, the whole team covering all the collection of patterns. Each member of the team further explained the patterns s/he read to the rest of the team, making sure that all are aware of all the patterns. Three of the patterns were not clear and the team asked the facilitator to provide more background on them (i.e. Choose your collaborators, Who is the coordinator?, and Revert changes). This team made a minimal use of the patterns. They initially placed them aside and focused on the design without hinting to them. Due to the fact that the decisions taken by the team did not point towards a collaborative application, the facilitator intervened, asking the team to specifically refer to the patterns. This brought little change, the team browsing the collection of patterns once. However, this one time browsing moved their focus from an application designed for individual use to an application designed for collaboration. Moreover, at this point the team enhanced their design with some of the solutions proposed by the patterns (eg. the chat mechanism, social features), but without specifically referring to the patterns themselves.

6.3.1.6 Team no. 6 - Fundamental problems

The sixth team chose the problem of collaborative drawing. They all went through the patterns, making sure that each member of the team is aware of the core idea of each pattern. Some of the patterns (namely, Collaboration, always social and Choose your collaborators) reminded the team of other types of applications such as social networks. This kind of associations helped the team better understand how to choose one possible starting point for their design process. The team spent a considerably large amount of time discussing the patterns at the beginning of their work, even prior to choosing the problem they would design for. They expressed the fact that it was due to this discussions that they explored their options and understood how to structure their overall design process. The specific thing they did for each pattern was to explain to themselves the problem addressed by it, so that they become aware of all the "fundamental problems" (as they called them) they should be aware of. During designing, they would come back to a pattern and analyse its solution once they faced the problem addressed by it. Interesting enough, initially the team planned on using all the patterns. It was only after one of the members explained the following "*we don't have to use them all. If you decide to implement that and solve that problem, then the pattern proposes you a solution*" that they gave up this idea. As already documented in the literature and observed during teaching design patterns, novices are specifically biased towards making use of a larger number of patterns. After deciding on the problem, the team filtered out all the patterns they were considering using. Further on, at a later stage of their process, they walked through all of them once more in order to make sure they did not miss anything fundamental.

6.3.1.7 Team no. 7 - Inspiration

The seventh team chose the problem of collaborative puzzle solving. They all went through the patterns and discussed them between themselves, trying to find for each pattern a situation where it would be applicable. Annotate pattern was not clear for the team, the facilitator providing more background on it. In addition to that, the team asked for a detailed comparison between the patterns Annotate and Integrated chat, with an emphasis on the differences between these two. During the actual design process, the team made little use of the patterns. It is only after the facilitator asked about the patterns used that the team walked through all of them and discussed those used and those not used, but relevant to their work. At this point, they got some ideas on things they were not considering and incorporated them in their sketch. It was the case of the Shared summary - at first not considered at all, but after going through the patterns discussed and adapted to their application as final game statistics displayed as a summary of the collaborative game. Going through the patterns as through a check list, the team did not recognize that they actually used one of them (Eyes wide open). They initially considered that this pattern had nothing to do with their design and asked for more details on it. After being explained the its core idea, the team realized that they initially misunderstood it and actually used it in their design without being aware of this fact.

6.3.1.8 Team no. 8 - Pattern-driven

The eighth team chose the problem of collaborative crosswords solving. The team placed all the patterns on the table, arranged as in a matrix and read them all together. Immediately after choosing the problem, the team decided to turn to the patterns and go through them one by one. Starting from the problem addressed by each pattern, they asked themselves how the particular problem would match their design context. For example, they phrased the following: "*For example, how can you undo a wrong word written by another player? (starting from Collaborative undo)*". In other words, the patterns drove their design process, inspiring them and providing them with hints. The team heavily used the patterns also for getting ideas on how to structure their design process and how to address fundamental collaboration problems.

They went through the solutions proposed for each of the patterns and discussed on how should each solution match their context. It is safe to say that the patterns represented the kernel of this team's work, since they were discussed, applied, adapted, referenced throughout all the workshop. It was not clear for the team whether they should use the entire collection or just a sub-collection. The facilitator allowed them to make a decision on that and did not influence them in any way. Eventually, they did not make use of all the patterns, but they walked through all of them for several times to make sure they did not miss anything. The patterns they did not use fell in two categories: 1) patterns which they considered difficult (such as Support versioning) or 2) patterns which they did not consider fundamental for the problem they chose (for example, Annotate).

6.3.1.9 Team no. 9 - Confidence

The ninth team chose the problem of collaborative crosswords solving. They went through the patterns all together, each one reading a subset of them. After they started designing, they referred to the patterns as "the problems". Each time they would come across an issue addressed by the patterns (such as coordination, or awareness), they would be remembered of the patterns. They would go to the specific pattern addressing the issue at hand and discuss the possibility of applying it. Overall, the collection had a small impact on the team's process, supporting them, however, to make design decisions with a higher degree of confidence.

6.3.1.10 Team no. 10 - End cycle filters

The tenth team chose the problem of collaborative drawing. They initially arranged all the patterns on the table and went through them so that each member of the team reads a few of them. After having done that, they turned their attention to the problem they chose and started coming up with possible ideas for its design. They ignored the patterns for a while, but when facing the problem of making part of the application private they turned to them for support. This triggered a walkthrough of all of the patterns, helping the team enhance their design and address issues not considered until that point. To conclude an initial cycle of their work, they went through the entire collection again and put aside all those patterns used. Moving on and bringing new elements to their design, the team did not reference the patterns any more. However, the issues they addressed were closely related to the problems (if not the problems themselves) documented by the patterns. It is safe to assume that at this point the team was familiar with the collection and used it (even if without referencing it) being aware of it.

6.3.1.11 Team no. 11 - Redundancies

The eleventh team chose the problem of collaborative puzzle solving. The team's approach to working with the patterns was to initially filter some they considered "fundamental". Some of these patterns were Integrated chat, Adapt application to device, and Collaboration, always social. Moreover, they considered some of the patterns redundant. As example, they considered the pattern Annotate to be similar to the pattern Collaboration, always social. It is for this reason that the team decided to exclude from the very beginning some of them. Throughout their work, the team went through the patterns and considered addressing each of the problems documented by them. They analysed the solutions proposed by each pattern and collaboratively decided on which solution fits their design or how the solution proposed should be adapted to their work. For the patterns they have used, they marked their identifier on the sketch (in the place where the pattern was considered). Moreover, they only referred to the patterns through their identifiers. For the patterns they did not use (even from those remaining after filtering them initially), they explained to themselves why the particular pattern is not useful in their design context.

6.3.1.12 Team no. 12 - Division of work

Team number 12 chose the problem of collaborative puzzle solving. Each member of the team selected one pattern from the collection. They read the five patterns selected and discussed them among themselves. These patterns were: With or without collaboration; Collaboration, always social; Integrated chat; My contribution; Eyes wide open. Further on, as a result of their discussions, they chose one pattern and decided to only use that one. It was soon after they started their work that they realized that one pattern would not support much of their process. As a consequence of that, they turned to the collection and went through it. This time, they read each pattern aloud and wrote down those that were addressing the problems they thought relevant for the design of the application they chose. They divided their work so that some of them focused on the actual design and others constantly went through the patterns and checked which of them would fit to be discussed and possibly applied in the current context of their work. Once a pattern was considered suitable to be applied, the team would collaboratively discuss the solutions proposed and the way to apply this solution to their design. The common constant of their work was the division of work as described above and the constant synchronization of the two sub-teams.

6.3.1.13 Team no. 13 - One (pattern) solves all (problems)

The thirteenth team chose the problem of collaborative crosswords solving. The team went through the collection of patterns before starting their work, each member of the team reading a few of the patterns. They chose to use Integrated chat pattern to solve several problems. First, they addressed the problem of communication within the application under design and, for that purpose, the pattern answered clearly. However, they decided on sticking to this one pattern and use it as a coordination and visualization mechanism as well, even if the two issues were addressed in other patterns thoroughly. Because the team did not make much use of the collection, the facilitator decided to go through the patterns together with the team and explain the core idea of each of them once again. The facilitator made no reference to the problem chosen by the team, so she did not in any way bias the team towards using specific patterns in specific situations of their design context. This intervention, however, did not change much of the team's behaviour, the team remaining reluctant to using any other patterns. They motivated their behaviour by the fact that they had never worked with patterns before, so they felt the concept needs more time to be explained and understood.

6.3.1.14 Team no. 14 - Pattern mash-up

The fourteenth team chose the problem of collaborative drawing. Initially, each member of the team read all the patterns, so by the end of this phase all of the members of the team were aware of the entire collection and of the issues it addressed. Their approach was pattern driven in that they basically thought of how to incorporate the patterns they considered more relevant in one application. They took each pattern at a time and designed a feature in the application based on the solution it proposed. By the end of their design process, what they came up with was a mash-up of several solutions proposed by the patterns provided to them. Asked by the facilitator to describe their final result, the team made intensive reference to the collection, pointing out that their intention was to use as many of the solutions proposed as possible.

6.3.1.15 Team no. 15 - Turnarounds

Team number 15 chose the problem of collaborative drawing. The team initially placed all the patterns on the table, arranged as a map and all read the patterns together. Discussing the patterns all together several things became clear for the team. First they all agreed that all the collection describes one domain and that each pattern documents a specific problem in the domain. Such insight was not provided by the facilitator, so the team was in no way biased. Further on, they selected some of the patterns they considered mostly relevant and put them aside for further consideration. Moreover, when faced with a problem such as the situation where more users would draw in the same time, they turned to the whole collection in search of the one addressing the coordination problem. Such turnarounds were common to the work of this team, the participants turning to the patterns often for searching for the problems they were facing or for checking what other ideas they can include in their designs.

6.3.1.16 Team no. 16 - Understanding the domain

The sixteenth team chose the problem of collaborative searching. They initially went through the patterns in a random fashion, each member of the team reading a few of them. They focused intensively on the design of the application considered and in trying to come up with scenarios in which a group of people would search together synchronously. It was unclear for them how such a process would develop and only after the facilitator described an example, things became clearer. However, they did not make use of the patterns throughout this struggle. It was only by the end of the workshop that they turned to the collection in an attempt to check how this would fit in the design they came up with. They considered each pattern and discussed whether the problem addressed by it was considered by them and if not, they explained to themselves why the pattern is not applicable in their situation.

6.3.1.17 Team no. 17 - Refactoring

Team number 17 chose the problem of collsaborative crosswords solving. The team divided the patterns among themselves and each member of the team read 3-4 patterns. After each one read a sub-group of patterns, they swapped them so that everybody gets to read them all. Then, they proceeded to the design without looking too much at the patterns. After they reached a draft of their design, they went through all the collection in order to check which ones they have used. Discussing the patterns helped them clarify several points in their design and made them realize that some of the design decisions taken were leading towards a dead-end. Based on these insights, the team refactored their design, applying solutions proposed by the patterns. Also common to this team was adding elements initially not considered after consulting the patterns. As example of such a situation is adding a new screen to the application under design after discussing the pattern Choose your collaborators. The screen would address specifically the issue of allowing each user of the application to choose his/her collaborators.

6.3.1.18 Team no. 18 - Pattern taxonomy

The eighteenth team chose the problemlem of collaborative drawing. The team read the patterns one by one, all together and divided them in three categories as follows:

- Shared summary; Integrated chat; Choose your collaborators; Resume collaboration;

- Track history of collaboration; Support versioning; With or without collaboration;

- All the other patterns in the collection;

They considered the patterns in the first category easy to understand and apply; those from the second category were assigned a medium level of difficulty; the last category was considered to be containing patterns difficult to understand. The team initially used in their design only the solutions proposed by the patterns in the first category. Later on, they reconsidered their option and agreed to go through all of the patterns, motivating "*See if looking at the patterns, you find something else we should add*". Going through the patterns and discussing them, they eventually added to their design new elements. Also, patterns which at first seemed difficult became more clear once discussed collaboratively. In addition to that, the team associated these patterns with examples of their application the team was familiar with and suggested the facilitator to add to the description of each pattern at least one example of the its application.

—

All in all, the teams made use of the patterns even if not all the teams considered them fundamental to their work. They found the patterns' representation accessible and somehow fun to use which, in addition to the team work, motivated them to get engaged in the whole process. The facilitator's intervention was minimal, so they were not biased in any direction and the choice of using or not using the patterns was all theirs. Even if the concept of design pattern was new to most of them, they were interested in learning about it and using it in this context helped them considerably. The teams turned to the patterns when in doubt, or when searching for ideas. The patterns helped the teams validate their decisions or get more confidence in the path they were following. Some of the teams were somehow selective in terms of the patterns, filtering those they considered fundamental and only using those. Others, on the other hand, were completely driven by the patterns and made extensive use of them. A more detailed description of the strategies of use for the patterns is presented in Section 6.4.2.

6.3.2 Questionnaire Results

Each participant filled in the questionnaire provided in Appendix .6. Based on their answers, this subsection provides their feedback in terms of the level of understandability of the patterns, the degree of usage of the collection, the extent in which they considered modifying the patterns and the overall impact of using patterns in their collaborative work.

6.3.2.1 Understandability

The problem and the solution described by each pattern were the elements considered the most useful in understanding a design pattern, 51% of the participants rating them as such. On the other hand, 78.7% of the participants found the unique ID of each pattern as the least useful element for understanding patterns. As expected, a relatively large number of participants (32.4%) found the name of the patterns helpful. However, even if the illustration assigned to each pattern was expected to help the participants grasp the main idea of each pattern, results showed little evidence of the usefulness of this element.

The average rate of understandability of the collection of patterns provided was 3.91 (S.D. 0.341) on a scale from 1 to 5 (with 5 being very understandable), proving that, overall, the participants faced little trouble in grasping the idea of each pattern and its usefulness. The complete information on the rate of understandability of each pattern is depicted in Figure 6.3.

6.3.2.2 Usage

As answer to the question "To what extent were the patterns useful for the following (on a Likert-type scale, with 1 – not at all useful, 2 – not useful, 3 – I can't say, 4 – useful, and 5 – very useful)?", participants rated:

- Searching for documented problems with an average rate of 4.28 (with 46.7% of the participants rating it as very useful)

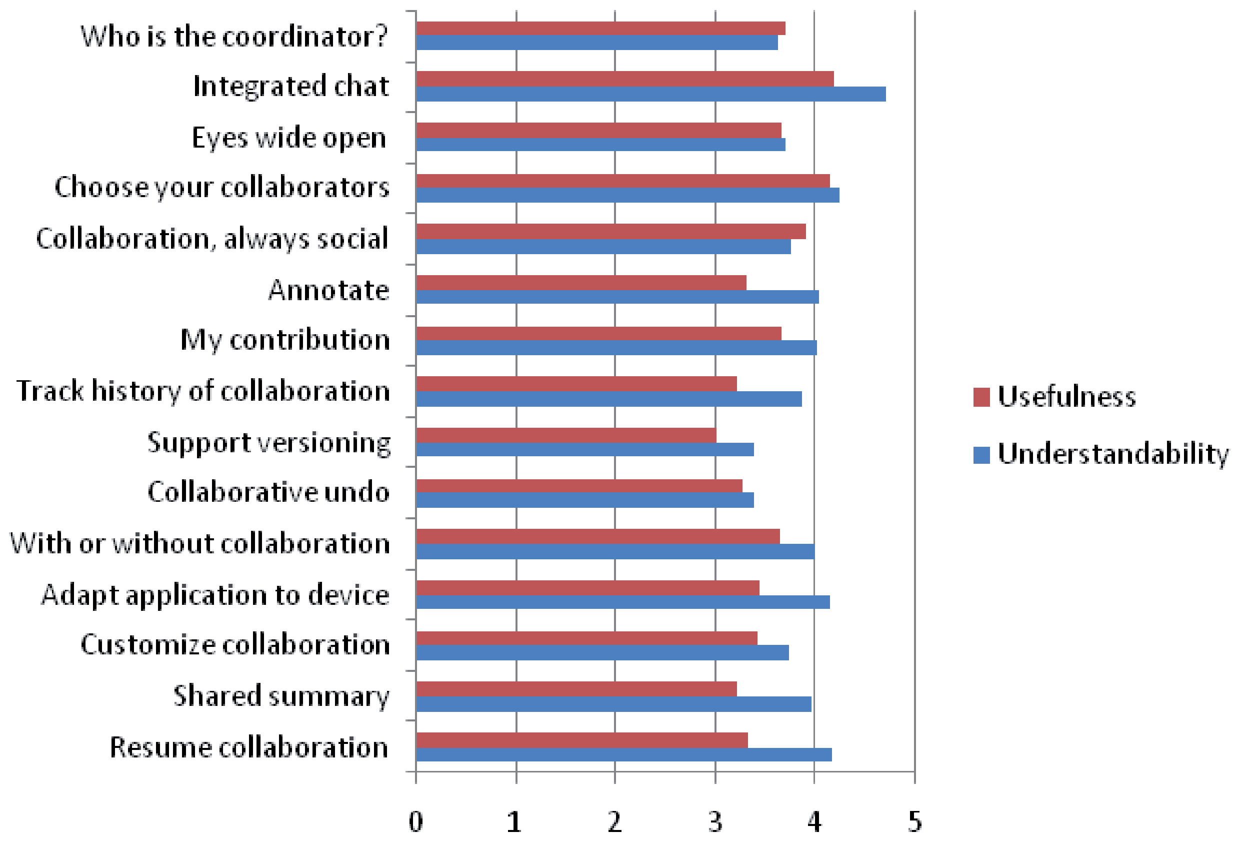


Figure 6.3: Understandability and usefulness rates for each of the patterns

- Searching for solutions for already identified problems with an average rate of 4.13 (with 45.9% of the participants rating it as very useful)
- Communicating with other members in your team, 3.93 (with 44% of the participants rating it as very useful)
- Brainstorming for design ideas for the application under design, 3.79 (with 34.6% of the participants rating it as useful)
- Understanding the design space of the application, 3.48 (with 25.3% of the participants rating it as very useful)
- Remembering similar design situations previously encountered, 3.32 (with 26.7% of the participants rating it as very useful)

Moreover, some patterns proved specifically useful for some of the above mentioned actions, according to the data provided in Figure 6.4. For example, much debate has been around the pattern "Choose your collaborators", the participants trying to come up with solutions for allowing users to start their collaborative process and to choose the users they want to work with. The pattern "My contribution" reminded the participants of several contexts which request the identification of one individual's contribution and of existing applications which support this action. Also, throughout their design processes, the participants mostly searched for solutions for the problems of: a) adapting the application to a specific device, b) allowing the undo operation on a collaboratively edited resource, and c) versioning.

6.3.2.3 Modifiability

Sixty-one (61) of the participants (81.33%) mentioned that the information provided for defining each pattern was enough and that no additional information would be needed. Eight (8) of the participants (10.66%) would have found a set of examples of application of each pattern useful in better understanding the idea of each pattern. Two (2) of the participants (2.66%) suggested adding some more details in the description of each pattern, while keeping the same defining elements. Lastly, one of the

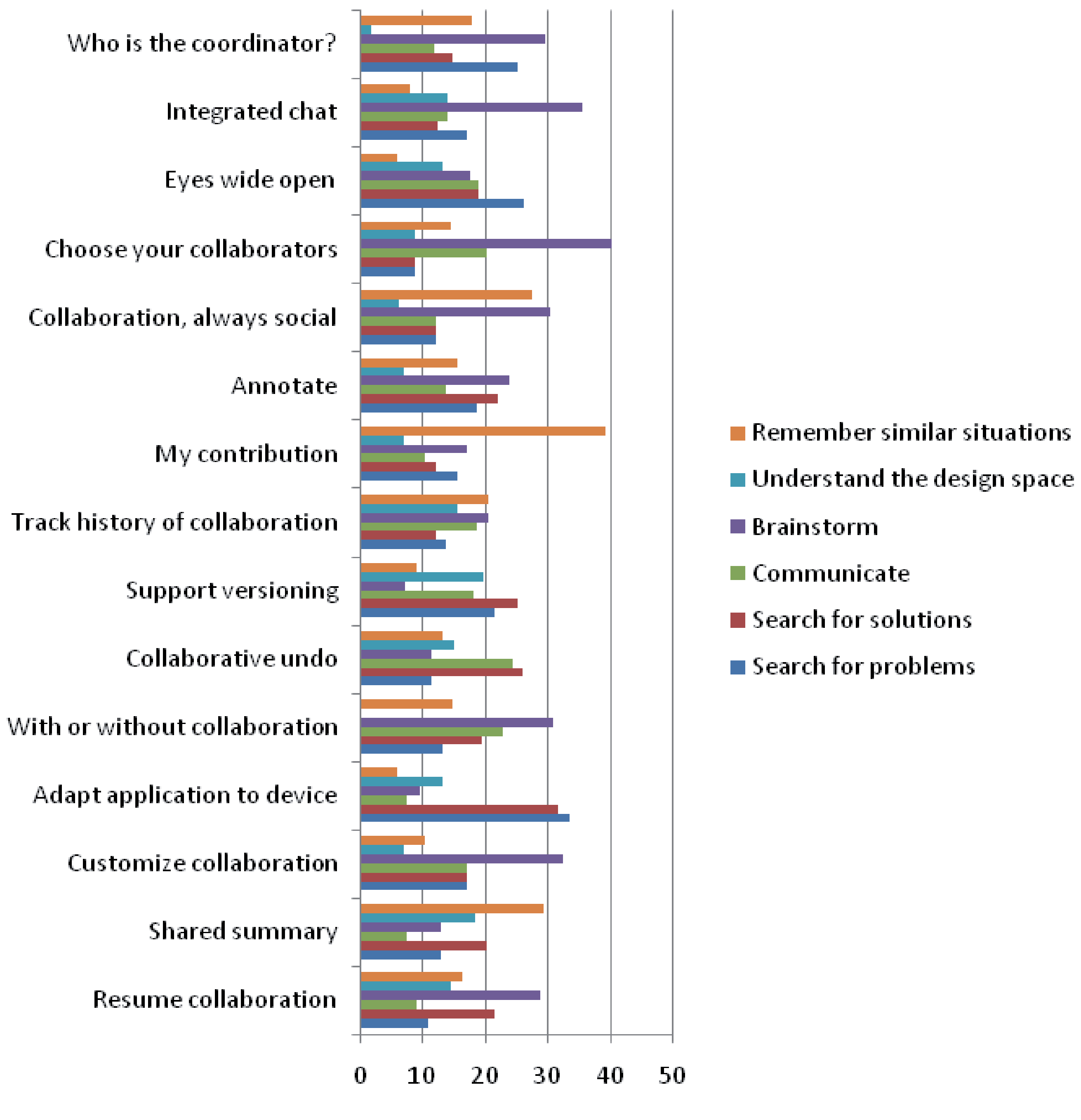


Figure 6.4: Degree of usage (in percentage) of each pattern for specific actions

participants (1.33%) suggested adding a defining element for each pattern able to list down some frequently asked questions related to the pattern.

Having to choose a representation for the collection of patterns which would best suit collaborative design processes, 62.5% of the participants opted for the paper card representation, 28.8% chose a search engine application, 13.9% opted for a wiki-like application, while only 4.2% of the participants considered the option of a specialized application to work with design patterns.

6.3.2.4 Overall Feedback

Asked to rate the overall usefulness of the patterns, participants provided an average rate of the collection (computed as the mean of the usefulness rates of all the patterns – Figure 3) of 3.53 out of 5 (S.D. 0.346). As support for the above mentioned quantitative data, some of the answers to the open question "How have the patterns supported your design process?" included:

- "*They [the patterns] helped us in searching possible problems. We analyzed all of them to check how each pattern applies to our design process.*"
- "*They [the patterns] were very useful in the beginning of the workshop for understanding what should we consider and in which way. Also, during the work, they helped us maintain a coherent and detailed line of work.*"
- "*They were fundamental in guiding us through the realization of the project. They helped us consider things that we wouldn't have considered without them.*"
- "*The patterns allowed the discussion among the group members and the exchange of ideas.*"
- "*The patterns provided indications and a precise schema on which to reason for solving the problems encountered during the workshop.*"

6.3.3 Transcripts

The conversations of the teams were recorded and transcribed for further analysis. The transcripts were divided into sentences (i.e. small fragments of dialogues – usually lines of the dialogues – related to a particular concept or action) and those sentences referring to patterns were coded according to the coding scheme described in 5. The tool used for coding and synthesising results is QDA Miner 3.2[1].

6.3.3.1 Atomic Actions

Browsing the collection was marked in cases when participants:

- were checking not to have missed any of the issues addressed by the patterns:
  - *"Is there anything else we've missed?"*
  - *"Let's look at the patterns again not to forget something fundamental."*
  - *"Let's think about something that we didn't use and we should consider (looking at the patterns)."*
  - *"Ok, let's see what else (looking at the patterns)."*
- were evaluating which of the patterns they have already used until that point in the process:
  - *"So how many patterns did we use?"*
  - *"Let's see which of the patterns we used (went through the patterns and put aside those used)."*
- were trying to come up with new ideas for their design and searching for inspiration:
  - *"Yes, let's look at them [the patterns] one by one."*
  - *"Let's just finish this and then we go through all the patterns together."*

[1]http://www.provalisresearch.com/QDAMiner/QDAMinerDesc.html

- were uncertain about the role of the patterns in the whole process and were trying to understand how the patterns could help - *"Let's see what to do with the patterns. Are we looking at each of them separately, one by one?"*.

—

Reading a pattern during the task was usually performed by one of the members of a team who would afterwards inform the others of the main idea of the pattern:

- *"Ok, there is a thing I read here (the patterns): for understanding who has placed a certain piece, how could this be done?"*
- *"Ok, I have just read this - statistics of the application use in cases of games."*

In other cases, a team would read a pattern all together either in search of inspiration - *"We read the solutions, maybe we get some ideas"* - or for a better understanding of the problems to be faced in their design process - *"Everyone, read it again and we should find what we think that could be problems (i.e. what applies to our design) and write them down"*. Reading a pattern was not always specifically expressed by the participants, but sometimes only noted by the facilitator who observed them - *"Went on to reading [Collaboration, always social]"*.

—

Using (applying) the solution proposed by a pattern would sometimes be marked by a simple acknowledgement of the fact, such as *"Chat, we have the chat"* or *"Great, considered!"*. Other times, the main idea of the solution proposed would be explicitly expressed after having been included in the design:

- *"Ok, ok, when you roll over with the mouse, the square gets illuminated and the tooltip would tell you who the author is."*
- *"Clicking on the name of the person here, it highlights all the pieces placed by that person."*

- *"Each player is assigned a colour."*
- *"We created a control for anyone who would like to pause the game and being an online game, the other players continue to play."*

Using a solution was sometimes marked by a justification of its use; the team would provide arguments on why that specific solution is used in that particular context:

- *"Do we add the chat on the first page as well? yes, because we said users can create groups based on the chat."*
- *"When you solve a puzzle you should have a private area where you try out the pieces and when a piece works well where it is placed, you just add it to the whole puzzle."*
- *"Ok, instead of colouring the pieces in the colour of the author, you could have them extracted a little bit from the context of the board so you can identify them."*
- *"I am user 1. I choose piece X. I click on it. He is user 2 and he can see that the piece is used, it is locked. The others see that it is locked."*
- *"So, we could say that who gets the piece first puts a lock on it, so the piece remains with that person until it is placed."*
- *"Then, you have the pieces on the board, you roll the mouse over them and you see their border coloured in the colour of the person who inserted the piece."*

The use of a proposed solution was also made explicit while the participants were checking to see which of the patterns they have already considered until a certain point in their design process:

- *"We used this one with the memory, and the updates."*
- *"We have said that a piece is locked when one clicks on it."*
- *"Coordinator [Who is the coordinator?] we are using surely, because the area gets blocked."*

Other times, after discussing a problem, the team would conclude to use the solution proposed for it by the patterns - *"So we use this one [Collaborative Undo]"*, or *"So in the end, one can see how many pieces each one placed. So we can use this one with the colours [My Contribution]."* Using a solution also came as a continuation of a thinking process, as if the team would describe the flow of their reasoning and include the use of a proposed solution in their discourse - *"Yeah, and then we add a notification like this one (point to pattern) saying something has changed"*. When using a solution, some of the participants would mark the identifier of the pattern describing it on the sketch of their design - *"We could include here the thing of the break. If all the members agree with the pause, they can take a pause; otherwise there is no pause - mark the id of the pattern resume [Resume collaboration]"*.

—

For *adapting* a pattern, the participants would start from a pattern in the collection and try to use the solution proposed by the pattern only to discover that the solution is either not completely matching their design or is missing some specific elements. Here are some examples of adaptations the teams decided on:

- For the pattern Choose your collaborators, one of the teams decided to adapt the solution proposed (i.e. inviting other peers to join a collaborative session) so that one can invite others by tagging them - *"Decided on tagging a person in order to invite a user to play with you"*. Also, choosing one's collaborators would require the creation of a group to which they can be invited to join - *"After I choose the users I want to work with, I can create a group of my own and invite them"*

- For the pattern My contribution, several ways to adapt the solution were proposed:

  – *"So should we add in the corner of each piece the initial of the player who placed it?"*

  – *"Each player gets an automatic ID - a, b, c, d. The ID is generated at the login So the representation of one's contribution is made by placing a b c in the corner of the piece placed."*

- For the Integrated chat pattern, several teams came up with several ways to adapt the solution to their own design:
  - *"Add also a dialogue box for discussing with the coordinator."*
  - *"In the chat we can also add colours for each user so that the lines of each player are marked in a different colour to support readability."*

  Moreover, they considered adapting the position of the chat feature in the GUI they were designing considering the context in which such a feature can be used - *"The chat must remain on the first page also because one might not know anything about the game so he can ask for opinions in the chat"*. For the case of an application designed for a small device such as a phone, they even considered a simplified version of a chat feature - *"We can have a minichat with phrases already defined"*.

- The pattern Who is the coordinator? suggested locking as a solution. One of the teams working on the puzzle solving problem adapted this solution to answer the context of the game:
  - *"Ok, so when one locks a piece, the piece gets obscure yes, and if I don't find the right place to put it, then it goes back to the pool of available pieces."*
  - *"Well, you click on the piece, it gets locked, you drag it on its position on the board, and that position is blocked as well so no one can place another piece on the same position."*

  A different solution to approach the problem of coordination was timers (included in the pattern Who is the coordinator?). One of the teams considered enhancing it with audio feedback such that it better matched the context of a game played by competing teams - *"Ok, so you would have a timer with the audio for the time passed so you get nervous"*.

Adapting a pattern also referred to proposing alternative solutions for a problem - *"We can choose between: a. click on a piece and get the control for 10 seconds or b. continuous click release it when the piece is placed"*, or *"What if we keep the list of* *players ordered in real time?"*.

—

Modifying a pattern was rare, the participants being reluctant to changing the patterns. None of the teams attempted to change the form in which the patterns were presented. Some of them did, however, express some criticism through the questionnaire.

—

Searching for a pattern was, in most cases performed when a team was dealing with a problem. The sentences coded with the code 'searching' would mostly be phrased as questions:

- *"Actually, when you start the game, how do you determine who participates in the game?"*
- *"The problem is: when you selected a piece, the others select pieces in the same time."*
- *"But in that way how do you know what pieces were placed by each user?"*
- *"And what if I click on the same definition as you?"*
- *"There has to be a way I can notify you about the fact that I am modifying it."*
- *"But how can I find someone in particular? Like if I want to play with you, how do I find you?"*
- *"The first problem is how are we indicating, how are you differentiating the users."*

Searching for a pattern was also common in situations when the team was insecure about the "right" way to go about a certain problem. In some cases, the team already had an idea of a solution in mind, but was not sure it would lead anywhere - *"Also,* *we should consider that showing what each person did may be very annoying, so the question is how to show this in the right manner"*. In other cases, they had no precise idea on how to tackle the problem and would search for guidance - *"How do we do it? Like there is someone who is coordinating the others, or there is no coordinator?"*. Some of the teams would also set a high-level goal of their design process and then they would search for the patterns documenting the problems which met that goal - *"Ok, this is indeed a track. It remembers who, how, what each one did, and when. Hence, this is a track. ok, this is the goal that we are setting. Now, what are the problems that one might face to accomplish this goal?"*. Lastly, the teams would go back to patterns already discussed during their work, searching for them - *"Where was that? (patterns)"*.

—

Discussing a pattern with other members of the team was common. The participants would discuss possible scenarios of application of a certain pattern in order to share with each other their understanding of the patterns:

- *"If I am new and I don't know anyone, I see the list of all available users and I can choose to work with some of them."*
- *"What if I pause and then I start looking at the pieces? No, you can't because once you pause, the screen goes blank, you can't see the pieces anymore."*
- *"We should design what the pattern says. If I want to see my contribution to the puzzle or drawing or whatever, I need a practical way to achieve this."*

At times, it was not clear for the teams how much a pattern covers, or in other words, what are the boundaries of the pattern's application. Some of their conversations were addressing precisely those concerns:

- *"No, what about the compatibility between computers. For example, mac and windows. Is that included there? Ok, let's just decide on java and it can be materialized on all."*

- *"It works on any device: phone, computer yes, we can do that and then make a version for each device yeah, I would like to see you with a photoshop mobile, though."*

A lot of the conversations the teams had were structured as negotiations in which participants of the same team would provide arguments in favour of the point they were trying to make. In most of the cases, these negotiations were around understanding if and how to use a specific solution proposed by the patterns:

- *"It's like having a private area in which you could try out pieces/ no, this makes no sense. You already try it out on the board/ no, it is a puzzle, you couldn't have a puzzle next to it where you play by yourself what if a piece is misplaced? well, we have said that in that case it remains there on the board. Someone else can correct it."*
- *"Should we add some comments? no. no because then people will use strange comments ok, but if I want to comment something, I should be able to do that. I say we add it."*
- *"If you want it to be a competition, you make it as turns. Each one has 20 seconds to place a piece and basta! ok, but the competition already consists in who placed more pieces on the board but it is more like collaboration, not competition."*, or *"So should we let them draw each one in his own area?"*
- *"No, we can just put here the button pause and let them decide when to take the break."*

Misunderstandings were also the focus of the teams' discussions. Such misunderstandings would relate both to the content of the patterns - *"No, we are deciding the rules of the game; not the rules of the collaboration [Customize collaboration]"* - or to the applicability of a pattern in a certain context - *"This is a collaboration, not a competition, so a global chat would be enough it would be a bit too much to have a private chat on a game like this. what happens if for example, we 4 want to play with each other but it is not about playing, it is more about I go there to see if I manage to give a contribution"*. In addition to solving misunderstandings, the teams' conversations were also meant to bring clarifications with respect to each member's understanding of the patterns. The participants would explain to each other the content and the main idea of the patterns:

- *"Exactly, this was one of the issues in the patterns. If one clicks on the piece and drags it, in that moment that piece is locked."*
- *"What is fundamental? The chat or the real time? I think that the real time. The chat is connected to the real-time. If there is no real time, then you don't need a chat."*

In some cases, the teams would make associations between the issues addressed by the patterns and similar issues they were familiar with and implemented in existing software applications:

- *"In poker yes, because it is a time competition, but for the puzzle, I can stay there one hour and not place any piece on the board."*
- *"The chat is real time is the chat integrated in the application? No, the chat opens automatically when you start the application as an insider to the application. It is basically like a pallete that you can open and close as you wish I think this is not very usable. They should all be in the same level ok, we can make it like facebook, you can reduce the chat window when you don't use it, and leave it open when you want to chat."*

Many of the conversations the participants had were meant to help them get a better understanding of the patterns:

- *"And for the pieces we would have a scroll or something to allow you to visualize them all/ ok, and the pieces you select are they in the public area or in the private area?/ they are private well, it depends on how we think about it. If we say that everything you do gets shared with all, then they are public or we could have a private puzzle and a preview of the public one with everything that others have done."*

- *"No, it is like she said you block the area of the word so you can't overwrite something written by someone else so, when I select the definition for 1, the area for that definition is locked and noone else can click it/ yes, exactly/ no, but one can modify it, but not in the same time but I would say it's better to make turns."*

While discussing the patterns, the participants would also evaluate some of the patterns as being unclear or difficult to understand - *"So if you want to delete something, the author gets a notification and may agree or disagree about the deletion. Let's put this pattern aside, it is not clear"*. Moreover, for some of the patterns, the participants brought to light possible drawbacks of applying a specific pattern in a certain context - *"Yes, but if the puzzle is already coloured? If you mark the piece with a color, you can't see the image anymore"*.

—

Discussing the case of *referencing* a pattern brings up two issues:

1. In what situations would a team reference a pattern?
2. In which way would a team reference a pattern?

As answer for the first question, one of the situations when a team would reference a pattern was when one member of a team would summarize the content of a pattern to the rest of the team. In most cases, such summaries would only contain a few words about the pattern, its main idea:

- *"This one. Sketch the ideas before adding them to the public area."*
- *"This one talks about keeping track of what has been done."*

Another situation of a participant referencing a pattern would include the case in which s/he would signal a pattern to the rest of the team, bringing the specific pattern to the attention of the team as a whole:

- *"Look at this, this is interesting. (points to pattern with or without collaboration)."*

- *"Look at this (points to pattern Choose your Collaborators) - provide a list with all the available users!!"*
- *"I mean this one, look - [Who is the coordinator?]'*

Also asking for clarifications - *"Revert changes: now, this one is the difficult one"* - or further explanations - *"This one, what is it? [Choose your collaborators]"* would lead to referencing a pattern.

As for the second issue (i.e. how the participants actually referenced the patterns), in most cases they used a keyword associated to the pattern (either by the paper cards or by them themselves) to bring the patterns to the attention of the team. Some examples of such cases are:

- *"We should use the one with the puzzle [Customize Collaboration]."*
- *"So in the end, one can see how many pieces each one placed. So we can use this one with the colours [My Contribution]."*
- *"The one with the chat we don't need because there is no chat."*
- *"The pause one we skip it?"*
- *"This one with the eye [Eyes wide open] does not have anything to do with our work."*

In other cases, the teams used the pattern's identifier (marked on the paper card) to reference it - *"And we should also write what happens if they do not agree, which is this one - id 14"*. Lastly, working all together round the same table, they also pointed to a pattern or simply picked it up from the pile of cards and showed it to the others.

—

Re-referencing a pattern allowed the participants to go back to patterns previously discussed:

- *"The pause [Resume Collaboration] we discussed it."*

- *"Well, I proposed it and you all said we shouldn't have it."*

Going back to a pattern for reminding themselves what the pattern was about is yet another situation of re-referencing a pattern; take as example the following sentence *"The pause [Resume Collaboration], what did we say about the pause?"*. Other times, a pattern would be re-referenced as being used in the design already - *"Back to the chat/ The chat [Integrated Chat] we placed it"*. The patterns as a whole were sometimes re-referenced, as well; such an example is the case when part of the patterns were previously filtered for being considered - *"They are all [the patterns used] here, I put them aside"*.

—

Generating a design idea inspired by a pattern occurred in one of the following situations:

- Participants combined different ideas described by different patterns to get to a new one:
  - *"Make a mash-up: chat and drawing."*
  - *"We make 2 tabs. One says: "show to" and then you can choose a person in the contacts you want to share it with by sending him an invite by mail or searching him by name in the contacts list. Also add a control "Send" for sharing the drawing."*
  - *"Or have a list will all the users and clicking on a user would tell you the group he is in or that he is available (in the case he does not belong to a group)."*
- Participants were searching for graphical representations of design elements described by the patterns:
  - *"Represent the timer in the UI as a clock."*
  - *"Let's represent the chat as a mobile phone situated in the backpack."*

  - *"Add a track history to the UI in the right side - a video icon which allows you to replay all the process."*

- Participants would make associations between the ideas described by the patterns and other features, common to social/collaborative applications but not included in the patterns:

  - *"We can add a box for comments like a guestbook."*
  - *"So each group has a level. You may join a group based on the level you choose."*

- Participants would first try to adapt an existing pattern to their own context and then add some new element to it. Consider as example: *"You can see in one colour the people in your team and in another colour those other online users"*. The team started from the pattern My contribution, adapting it to the context of a game design and adding a new element to it, i.e. the colour of a team. Similarly, consider *"Each one is associated with a percentage indicating is contribution to the puzzle so to the left, you have the id of each of the 4 players with the percentage of puzzle covered by each, the number of remaining piece"*.

- Participants would come up with new solutions for a problem documented by the patterns:

  - *"We can record all the process as a video so you can re-watch it later."*
  - *"No, let's do this: add three photos in the UI with the photo of the drawing after I finish, the photo after the 2nd one finishes and the photo after the last one finishes."*

—

An overall synthesis of the coding process (Figure 6.5) points out that *browsing* the collection of patterns was quite common, 7.3% of the sentences being coded as such. Similarly, 7.1% of the sentences were pointing to situations where participants were *searching* for a pattern, more than often searching for a pattern leading to browsing the entire collection for finding it. However, searching for a pattern did not always lead to *reading* the pattern (and by this I mean reading the entire content presented by the card on the pattern), since the percentage of sentences coded as "readPatt" was significantly lower (4.3%).

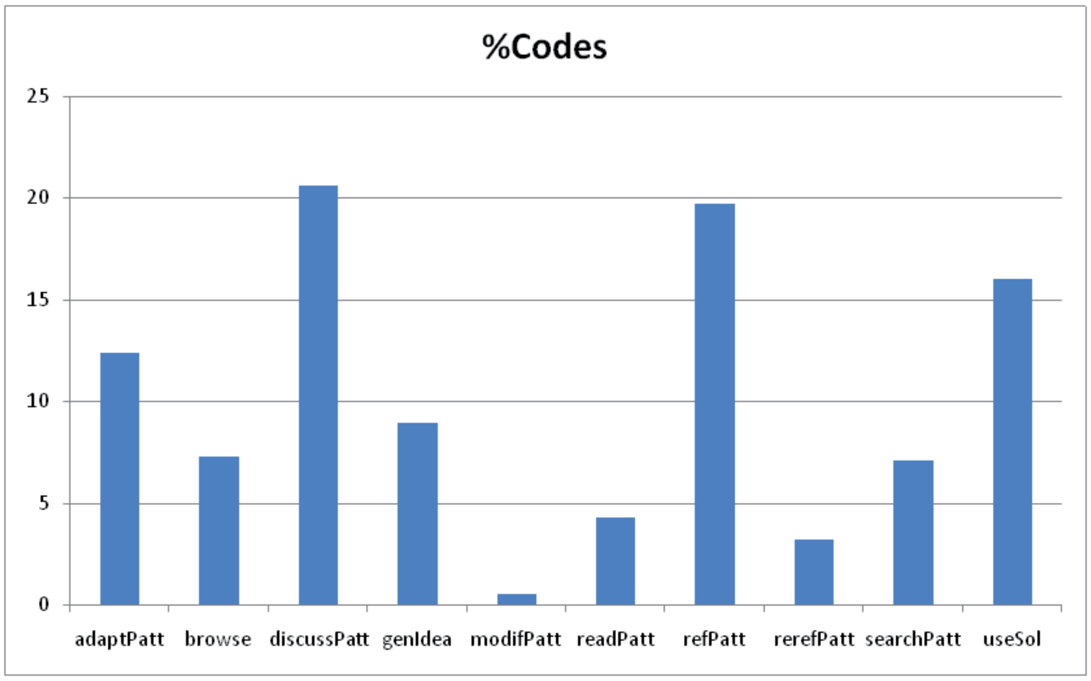


Figure 6.5: Transcripts results of the evaluation workshops - Distribution of the codes (% of codes)

Somehow unexpected, this low percentage associated to reading a pattern is an indication that participants would discuss and even use patterns without completely reading them. Sometimes, the name or even the solution was enough for them to grasp the idea of the pattern and not proceed to reading the pattern even further. Moreover, they deepened their understanding of the patterns by *discussing* them together and explaining them to each other, 20.6% (the highest percent) of the sentences coded indicating this.

As expected, the percentage of sentences coded as *referencing* patterns (19.7%) was just as high as the one of sentences coded as discussing patterns. Referencing a pattern would indicate bringing it to the attention of the whole team for being further discussed and explained. The discussions related to the pattern would compensate the lack of a complete reading of the content of the pattern. A complete reading of a pattern would only be performed when the team would still be unclear about the idea of the pattern or in the case of patterns they considered from the very beginning "difficult" (an example of that is Support versioning).

Sentences pointing to cases where solutions proposed by the patterns have been *used* in the process sum up 16% of all the sentences coded. 12.4% of the sentences were indicating *adapting* a pattern. These two - using and adapting - are closely related; solutions may have been discussed and then used as they were proposed or adapted to the current design context. A common situation was discussing a pattern and then trying to use the solution as proposed, leading to further debate. As a result of this, the team agreed that the solution can not be applied as it is proposed; hence it needs to be adapted to the context. Trying to use the solution of a pattern first, failing and adapting the pattern after further debate and only in the cases where the solution could not be used as it was proposed explain the higher distribution of the "usePatt" code.

A similar reasoning holds true for the distribution of the "genIdea" code which marked those sentences pointing to the cases where participants would come up with new ideas for their design after going through the patterns and discussing them. The percentage of these sentences was 8.9%, relatively close to the one associated to the sentences coded as browsing the collection. The common situation observed was that browsing the collection, the team would check for design elements not yet considered by them and possibly comprised in between the lines of the patterns. As result of this walk through and based on a continuous communication process, the teams came up with new design ideas and incorporated them in their work.

An interesting thing brought to light by this distribution is that the percentage of sentences marking the use of a solution proposed by a pattern is twice the percentage of sentences indicating the generation of new design ideas. Surely, more empirical work is needed to deepen such an observation, but this is a possible indication of the fact that much of the knowledge is reused in design processes rather that generated. Also, another explanation of these results could be that having the repository of patterns available, the team did not feel the pressure of coming up with new ideas and heavily relied on understanding and applying the proposed patterns.

Few sentences were *re-referencing* patterns (3.2%) and even fewer were pointing to cases where participants would want to *modify* a pattern (0.5%). The latter could be explained by the fact that the concept of design pattern was rather new to the participants; hence their reluctance to modify what has been proposed.

6.3.3.2 Action Sequences

Coding alone gives a hint on the actual behaviour of the teams in using the patterns. However, it is only by looking at their overall processes that one can better understand this behaviour. As a further step in this direction, I looked into the sequences of codes assigned to the sentences referring to the use of patterns. For defining a sequence of codes, consider the consecutive sentences p_1, p_2 (i.e. p_1 and p_2 are not separated by any other sentence) in the transcript document of one of the teams. If p_1 is associated with the code x and p_2 is associated with the code y, then xy is a sequence of codes.

As shown in Figure 6.6, for each code x from column A, the percentage of sequences of codes of the form xy (where y is any other code from the list of codes considered) is computed out of all the sequences of codes starting with x. In other words, the maximum on each line of the table basically says: "Out of all the sequences of codes starting with x, most of them are of the form xy, where y is the code representing the column of this maximum value" or "A sentence coded with x was in the maximum percent of the cases followed by a sentence coded with y". Together with a further discussion on the significance of these values, I will provide examples of consecutive sentences coded by such sequences. Consider code x one of the following:

Adapting a pattern (adaptPatt).

B / A	adaptPatt	browse	discussPatt	genIdea	modifPatt	readPatt	refPatt	rerefPatt	searchPatt	useSol
adaptPatt	15.7%	8.6%	15.7%	14.3%	1.4%	8.6%	18.6%	2.9%	5.7%	21.4%
browse	9.8%	7.3%	26.8%	7.3%		4.9%	34.1%	4.9%	17.1%	17.1%
discussPatt	10.3%	8.6%	23.3%	12.1%	1.7%	6.9%	33.6%	3.4%	8.6%	17.2%
genIdea	18.0%		14.0%	24.0%	2.0%	4.0%	12.0%	6.0%	12.0%	10.0%
modifPatt		66.7%	33.3%				33.3%			
readPatt	16.7%		29.2%	8.3%			16.7%	4.2%	8.3%	37.5%
refPatt	15.3%	7.2%	39.6%	3.6%		4.5%	22.5%	2.7%	6.3%	24.3%
rerefPatt	11.1%	16.7%	16.7%	11.1%			27.8%	16.7%		11.1%
searchPatt	10.0%	2.5%	42.5%	7.5%		5.0%	17.5%	5.0%	15.0%	12.5%
useSol	21.1%	8.9%	17.8%	7.8%		2.2%	27.8%	1.1%	5.6%	27.8%

0% - 10%	10.1% - 20%	20.1% - 30%	30.1% - 40%	> 40%

Figure 6.6: Frequency matrix in the transcripts results of the evaluation workshops - Percentage of action A followed by action B

In most of the cases where the participants would try to adapt a pattern to their current design context, the next thing they did was to use the solution of another pattern (21.4 % of the sentences coded with adaptPatt were followed by sentences coded with usePatt). The common scenario occurring was the following: a) the team would be close to the completion of the task, b) they would adapt the solution of a pattern to the current context of their work, c) this would remind them of a previously discussed pattern, in most cases related to the adapted solution just mentioned, d) they would consider using this later solution, incorporating it in the design together with the adaptation first discussed. Some examples of such consecutive sentences are described below:

p_1: *"In the chat we can also add colours for each user so that the lines of each player are marked in a different colour to support readability."*
p_2: *"And then we add the list of available user."*

p_1: *"Ok, so when one locks a piece, the piece gets obscure/ yes, and if I don't find the right place to put it, then it goes back to the pool of available pieces."*
p_2: *"And also, to know who blocked it, the piece gets a border in the colour of the user who blocked it."*

Browsing the collection (browse).

Somehow expected, the majority of the sentences coded as indicating browsing the collection of patterns were followed by sentences referencing patterns (34.1%). Going through the patterns, participants often stopped and referenced the patterns for further discussing them within the team. Some examples of such consecutive sentences are described below:

p_1: *"Ok, let's see what else (looking at the patterns)."*
p_2: *"Tagging, votes (while looking at the patterns). Are we considering tagging or votes?"*

p_1: *What else is here?*
p_2: *Annotations.*

p_1: *"Ok, let's go through all the patterns and see how could we use them."*
p_2: *"Shared summary."*

p_1: *"Let's think about something that we didn't use and we should consider (looking at the patterns)."*
p_2: *"My contribution can be something like in the end we would provide statistics on how many pieces each one placed."*

Discussing a pattern (discussPatt).

Discussing a pattern led in most cases to referencing other patterns, the percentage of sentenced coded with discussPatt followed by sentences referencing patterns being 33.6%. Some examples of such consecutive sentences are described below:

p_1: *"If we create a puzzle played in turns, we solve the problem of the pieces placed in the wrong location. Or you could say I want to place this piece here and then everybody votes, but that because really long."*
p_2: *"I would also add a chat, a simple chat. So I could say: I don't think your piece works well there."*

p_1: *"I would say there is no coordinator. All users are at the same level As for the real-time, I am thinking about a timer, so each person waits for his turn to place a piece on the board yes, this is better. Otherwise you create a mess."*
p_2: *"Yes, and then there is also the pattern with the pause."*

Generating a design idea (genIdea).

24% of the sentences pointing to the generation of new design ideas were followed by similar sentences. Some examples of such consecutive sentences are described below:

p_1: *"So we can decide that more users form a group and they can draw together in the same time. The drawing is shared only with the group and the chat contains only*

the collaborators at a given time."
p_2: *"We make 2 tabs. One says: "show to" and then you can choose a person in the contacts you want to share it with by sending him an invite by mail or searching him by name in the contacts list. Also add a control "Send" for sharing the drawing."*

p_1: *"We can do the same here. You divide the general list on categories: people I already know, people I don't know so, you could have the list of the people you have drawn with before."*
p_2: *"You can see in one colour the people in your team and in another colour those other online users."*

p_1: *"Also, you can see the list of already created teams and join one."*
p_2: *"If you can not find a team you want to join, you can create a new team."*

p_1: *"Also represent a chat in a fb style."*
p_2: *"Let's represent the chat as a mobile phone situated in the backpack."*

Modifying a pattern (modifPatt).

A relatively small number of sentences were coded as indicating the modification of a pattern, therefore the results with respect to this action are rather insignificant. However, most of the sentences coded with modifPatt were followed by sentences indicating the browsing of the collection of patterns. Some examples of such consecutive sentences are described below:

p_1: *"Ok, but for me this is useless knowing at what time what each person did and if you see a piece you can easily see who has placed it and after half an hour you don't really care about what each person did in the game. It's a game."*
p_2: *"Ok, so we have gone through everything."*

Reading a pattern (readPatt).

Reading a pattern led in most of the cases to either using the solution proposed by the pattern (37.5% of the cases) or discussing the pattern even further (29.2% of the cases). Somehow expected, this indicates that reading a pattern either made clear to the participants how to proceed with applying its solution or asked for more explanation on its content. Some examples of such consecutive sentences are described below:

p_1: *"Read this (the coordinator pattern)."*
p_2: *"If we create a puzzle played in turns, we solve the problem of the pieces placed in the wrong location. Or you could say I want to place this piece here and then everybody votes, but that because really long."*

p_1: *"Ok, I have just read this - statistics of the application use in cases of games."*
p_2: *"Great, considered!"*

Referencing a pattern (refPatt).

Referencing a pattern led to further discussing it in 39.6% of the cases. Referencing a pattern basically brings it to the attention of the entire team, the pattern becoming the subject for further clarifications and explanations. Some examples of such consecutive sentences are described below:

p_1: *"This one... sketch the ideas before adding them to the public area."*
p_2: *"It's like having a private area in which you could try out pieces no, this makes no sense. You already try it out on the board no, it is a puzzle, you couldn't have a puzzle next to it where you play by yourself what if a piece is misplaced? Well, we have said that in that case it remains there on the board. Someone else can correct it."*

p_1: *"How do they choose the collaborators? (pointing to the pattern)."*
p_2: *"It can be done randomly or in groups, because it is possible to create groups./ Oh, you mean groups./ Yes, like the playing rooms in poker, you enter, you see the rooms, and you can choose the room so you could either play random, or choose the room."*

p_1: *"The pause one we skip it?"*
p_2: *"Yes, you can just say that in the chat. / Yes, but what if I get out for one hour?/*

You can save and then you leave save what? save all what is done so far and when you enter again you are asked: do you want to update the session? And the drawing is updated with what else has been done."

p_1: *"Just thinking about undo."*
p_2: *"How should the undo work then? Could I delete other people's work we could say like this: if I like the drawing I could just save it locally or I can make it private."*

Re-referencing a pattern (rerefPatt).

Re-referencing a pattern was not often in the first place. However, 27.8% of the sentences re-referencing a pattern were followed by sentences referencing other patterns. As an explanation of this is the fact that going back to a pattern previously discussed, pushed the participants to check other patterns proposed in search of connections and associations able to help them better understand the collection of patterns as a whole. Some examples of such consecutive sentences are described below:

p_1: *"We could include here the thing of the break. If all the members agree with the pause, they can take a pause; otherwise there is no pause - mark the id of the pattern resume. back to the identification..."*
p_2: *"Yes, exactly. It is actually this - My contribution."*

Searching for a pattern (searchPatt).

Somehow expected, searching for a pattern mostly led to further discussing it (42.5% of the cases). On the other hand, the unexpected thing was that in a relatively small number of cases (5%) searching for a pattern led to reading the complete content of the pattern. Some examples of consecutive sentences coded with searchPatt-discussPatt are described below:

p_1: *"Where was that?... (patterns)."*
p_2: *"But you can add the chat in the profile area in the profile you have image, stats, chat. You see someone online, you can just send him a message./ So a chat/email?/*

Yes. So before you start playing you can just ask someone "do you want to play?"/ Ok, so we add an option for writing."

Using a solution (useSol).

Using a solution proposed by the patterns led to using other solutions or referencing other patterns in 27.8% of the cases. This is explained by the fact that the patterns in the collection are not independent. The problems they document are not independent, hence the documentation of these problems (i.e. the patterns) are related to each other. As a consequence, using a pattern leads to using other related patterns or simply referencing other related pattern. Some examples of such consecutive sentences are described below:

p_1: *"Chat, we have the chat."*
p_2: *"Then, you have the pieces on the board, you roll the mouse over them and you see their border colored in the color of the person you inserted the piece."*

p_1: *"When you solve a puzzle you should have a private area where you try out the pieces and when a piece works well where it is placed, you just add it to the whole puzzle."*
p_2: *"Yeah, and then we add a notification like this one (point to pattern) saying something has changed."*

Appendix .6 illustrates the computed Z-values and the probability values for each of the sequences of codes.

6.4 Discussion

6.4.1 Perceived Behaviour vs. Actual Behaviour

In this section, I will address the parallel between the two following aspects:

1. *The way the participants perceived their use of the patterns during the workshops.*

This was captured by the answers they provided through the questionnaires at the end of each workshop. Interesting in this respect was the question asking the participants to mark which of the following actions they consider they mostly performed using as support the collection of patterns:

- *understand* the design space of the application;
- *search* for design *problems*;
- *search* for *solutions* for already identified design problems;
- *communicate* with the other members of the team;
- *remember* similar design situations encountered;
- *brainstorm* for design ideas for the application;

2. *Their actual behaviour in using the patterns for completing the given design task.* This was captured by the recorded conversations and interactions between the participants throughout the workshops. The coding scheme used to analyse the recordings associated the sentences of the transcripts with the following codes:

    - *browse* the *collection* (browse);
    - *read* a *pattern* (readPatt);
    - *use* a proposed *solution* (useSol);
    - *adapt* a proposed *solution* (adaptPatt);
    - *modify* a *pattern* (modifPatt);
    - *search* for a *pattern* (searchPatt);
    - *discuss* a *pattern* (discussPatt);
    - *reference* a *pattern* (refPatt);
    - *re-reference* a *pattern* (rerefPatt);
    - *generate* a design *idea* based on consulting a pattern (genIdea);

A detailed description of how these codes were used and what observations they allowed to be inferred is presented in Section 6.3.3.

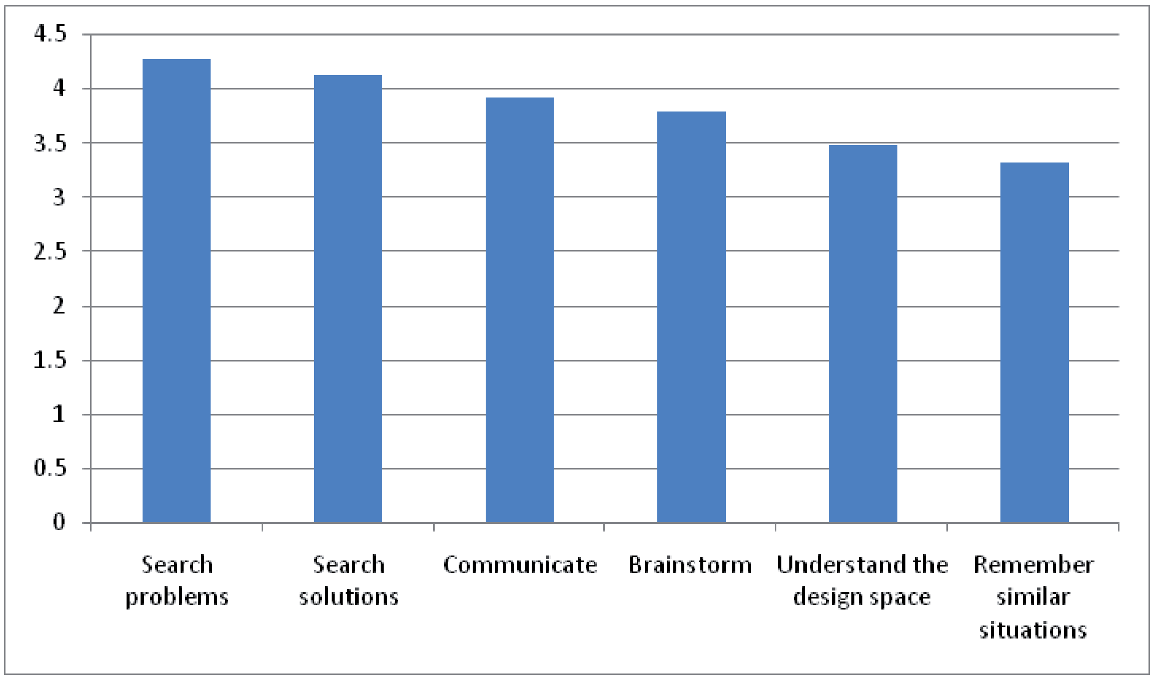


Figure 6.7: Questionnaire results - Use rate for each action

The questionnaires' results (Figure 6.7) indicate that the participants have perceived searching as the most common action they performed. Whether it was searching for problems or for proposed solutions, they rated searching as the most common action they performed during their work. Rated as the third most common action was communicating with other members of the team on the basis of the patterns, closely followed by brainstorming for new design ideas. Interesting enough, the participants did not perceive the patterns as being of much help in supporting them in understanding the design space of the application or in remembering similar design situations.

On the other hand, looking at the transcripts, it can be inferred that the actions the participants mostly performed are discussing a pattern, referencing a pattern and using a solution proposed by a pattern (Figure 6.8). At the opposite pole, participants

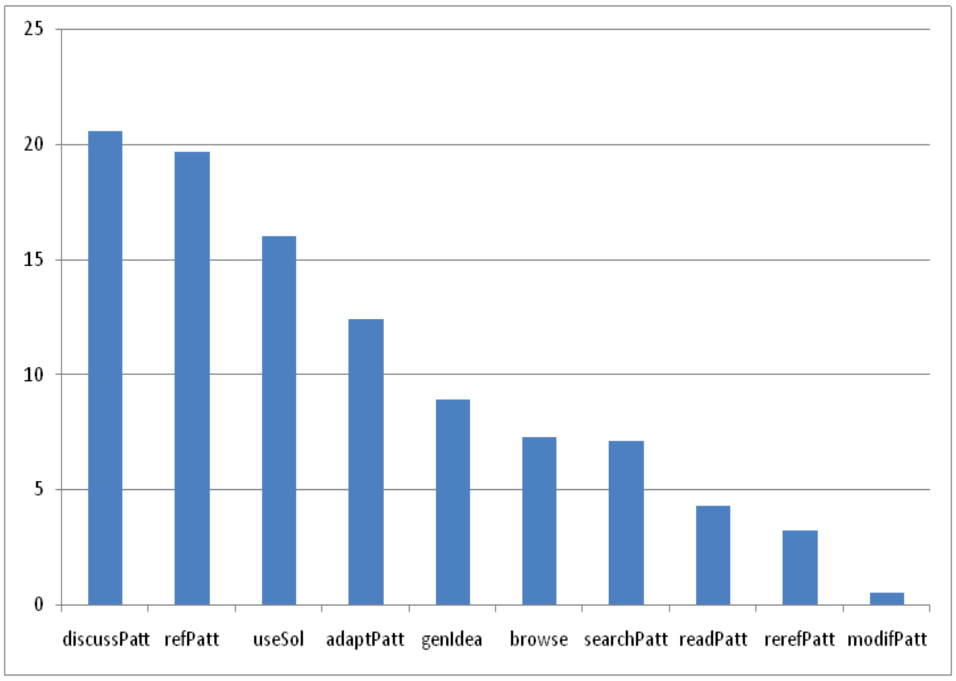


Figure 6.8: Transcripts results - Code frequency for each action

rarely modified patterns, re-reference them or even completely read their content. As a first remark, even if searching seemed to the participants as the most recurring action they performed, communication paid a much more important role in their work.

Looking in more detail, I tried to define a mapping (Figure 6.9) between the two sets of actions: the one provided in the questionnaires and the one described by the coding scheme. The reason they are not the same is that, since the participants were not expected to reflect on their behaviour in using the patterns but rather in performing the design task, the list of actions they were provided with is at a higher level of generality. For them, the patterns were a support which might have even been ignored. It was not expected from them to reason on the way they use the patterns, but to simply use them in case they see fit. On the other hand, the transcripts allowed a more in-depth analysis; therefore, the coding scheme was defined to comprise a larger number of actions.

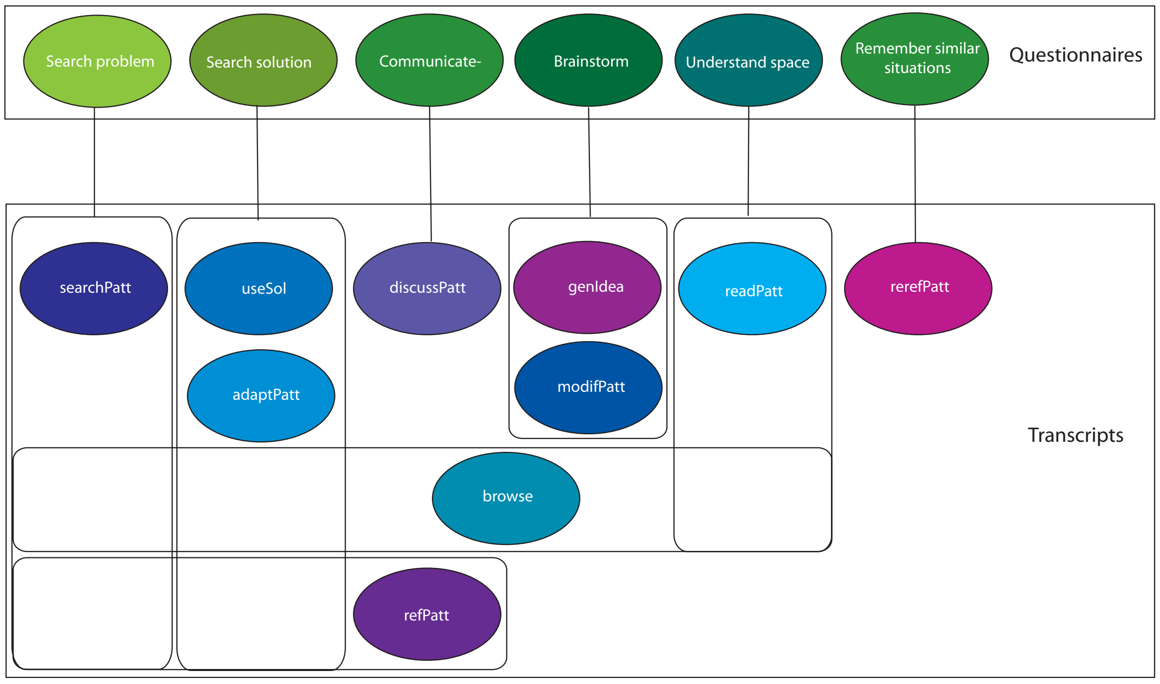


Figure 6.9: Mapping the questionnaire actions to the coding actions

What the questionnaires portrayed as searching for problems could be associated to the searchPatt, refPatt and browse codes used for the transcripts. On a similar line of thought, communicating with other members of the team can be directly associated to discussPatt. Brainstorming is the action represented in the coding scheme by genIdea, and modifPatt; whereas, understanding the design space of the application is associated to readPatt and browse. In the context of this mapping, remembering similar design situations is associated to rerefPatt. Lastly, searching for solutions is associated to usedSol, adaptSol, browse and refPatt.

I used the definition of the mapping described above in establishing the relationship between the two variable: rates computed based on the questionnaires, code percentages computed based on the transcripts' coding. Consider:

$$X = 4.28, 4.13, 3.93, 3.79, 3.48, 3.32$$

as the variable representing the questionnaires' rates computed for the following actions, respectively search problems, search solutions, communicate, brainstorm, un-

Var	Mean	StDev	Variance	Sum	Min	Max	Range
X	3.82	0.37	0.13	22.93	3.32	4.28	0.96
Y	9.91	6.64	44.19	59.46	3.2	20.6	17.4

Table 6.1: Perceived vs. actual behaviour - Descriptive statistics

derstand, remember. Moreover, consider the mapping between the two sets of actions defined above (Figure 6.9):
Search problems $\rightarrow$ searchPatt, browse, refPatt,
Search solution $\rightarrow$ useSol, adaptPatt, browse, refPatt,
Communicate $\rightarrow$ discussPatt,
Brainstorm $\rightarrow$ genIdea, modifPatt,
Understand $\rightarrow$ readPatt, browse,
Remember $\rightarrow$ rerefPatt.

For representing the second variable, consider:

$$\text{Y} = 11.36, 13.85, 20.6, 4.7, 5.8, 3.2$$

, where
y_1 = avg(%searchPatt + %browse + %refPatt) = 11.36,
y_2 = avg(%useSol + %adaptPatt + %browse + %refPatt) = 13.85,
...
y_6 = avg(%rerefPatt) = 3.2

The descriptive statistics for the two variables, X and Y, are illustrated in Table 6.1. The correlation coefficient, r is 0.64745. Surely, any generalization requires more empirical work and this result can not lead to any firm conclusive statement. However, this is a timid indication of the fact that the participants interpretation of their use of the patterns differed from the way they actually used the collection. Therefore, further implications of this case study will be inferred with a closer look at the transcripts, the facilitator's notes and the participants qualitative feedback.

6.4.2 Strategies Identified

As a further step in analysing the results of the case study, I tried to identify possible overall strategies the participants developed (willingly or not) while using the patterns. Abstracting from:

- the sequences of actions the teams performed on the collection of patterns in isolated contexts of their design processes
- the ratio of each category of actions the teams performed
- the facilitator's notes on the participants' interactions,

I identified a set of strategies the participants developed while using the design patterns.

6.4.2.1 Customize Pattern Identification

In going through the patterns and trying to get familiar with the problems addressed by them, the teams often tried to associate each pattern with a characteristic word. Having done that, their dialogues would contain references to the patterns through the words associated to them (e.g. "*We can decide on a fixed time for all the game and during the game one can take maximum 2 breaks, and then we look into the solution for the pause one* [the pattern Resume collaboration]"). Interesting enough, these words were not always consciously chosen from the list of keywords provided in the description of the patterns. However, with the exception of one case, all the words the teams associated with the patterns already belonged to the list of keywords provided by the cards. Two of the teams filtered the collection of patterns after going through it and discussing it once and chose a subset of these patterns they considered fundamental for their design process. Throughout their work, they referred mostly to these patterns.

6.4.2.2 Signal Patterns

Often times, while some of the members of a team were focusing on the design task, the other(s) browsed the collection of patterns and tried to relate the team's design decisions to the solutions proposed by the patterns. When the team member(s) browsing the patterns identified a useful pattern at a specific moment, s/he signalled this pattern to the team. Some examples of such references are: "*Ok, there is a thing I read here [My contribution]: for understanding who has placed a certain piece*", or "*Look at this, this is interesting [points to pattern With or without collaboration] When you solve a puzzle you should have a private area where you try out the pieces and when a piece works well where it is placed, you just add it to the whole puzzle*". Signalling a pattern would bring the pattern to the attention of the entire team for further discussions. In a larger scale project, signalling a pattern could help in both:

- dividing the labour so that part of a team makes the rest of the team aware of the issues already documented by the repository of patterns used;
- recommending patterns among collaborators in a recommender system fashion;

6.4.2.3 Search – Analyse - Apply

The most common strategy the teams were expected to choose consisted in: a) initiate by writing down possible problems they would face, b) browse the collection of patterns searching for those patterns documenting the problems they considered, c) point to a pattern once found and read it, d) analyze the solutions proposed by the pattern and assess which solution to apply. Contrary to the expectations, less than half of the teams adopted this precise path of actions.

6.4.2.4 The Pattern Collection as a Checklist

Ten out of the 18 teams used the collection of patterns also as a checklist. They initiated their work after going through the patterns, but initially ignored them. After reaching an idea for the application they were designing and sketching a draft of it, they went through all the patterns, one by one, in order to make sure that they covered all the issues addressed by the collection. For each of the patterns, they analysed whether they considered the issue addressed by the pattern or not. In the affirmative case, they identified the solution they used or adapted. Such an example is: "*We used this one with the updates [pattern Eyes wide open], and we sent notifications*". In the negative case, they explained the reasons for which the pattern did not apply to their design context. An example of such a reference is: "*The pause one [pattern With or without collaboration], we skip it?/ Yes, we have included that in the chat feature*".

6.4.2.5 Patterns as Startup Tools

Four of the teams initiated their design processes by going through the patterns, one by one, and identifying how each pattern could be applied in the context of their application's design. Then, when faced with a problem during their design process, the teams tried to remember which of the patterns addressed that problem. Examples of such references are: "*Yes, there was a pattern on that*", or "*There was one [pattern] that was mentioning the saving because if we are 5 and we decide to save, we should be able to do that*". Moreover, specific situations faced during the design process reminded the teams of the patterns they browsed at the beginning of the process. As example, consider "*Exactly, this was one of the issues in the patterns. If one clicks on the piece and drags it, in that moment that piece is locked*".

6.4.2.6 Patterns as Source of Inspiration

A common behaviour of all the teams was to consult the patterns ever so often during their design processes. This helped them explore their design options and take informed decisions on the solutions to consider applying. Moreover, once going through the patterns, the teams considered problems and design ideas they wouldn't have considered otherwise. The patterns inspired the teams in adding elements to their designs, and some example of such situations are: "*Let's add something about notifications [after reading Eyes wide open]*", or "*How do they choose the collaborators? [pointing to the pattern Choose your collaborators]*".

6.4.2.7 Mark the Use

The final result provided by each team was a sketch or a mockup of their overall design. No strategy was suggested to the participants for marking the patterns used. However, there were three ways they decided to address this. The majority of the teams grouped together all the patterns they used, putting them aside. Others have decided to arrange the patterns in the order they used them throughout the process. A more systematic approach was adopted by two of the teams which annotated their sketches with the identifiers of the patterns they used, marking the use of each pattern in a specific context of the application's design.

6.4.2.8 What do you mean?

Patterns were often used as means of making oneself understood. The teams used the patterns in order to explain each other concepts or to discuss open issues or misunderstandings. For example, one of the most challenging concepts to grasp was reverting changes, the teams making use of the Collaborative undo pattern to explain each other the concept and the way it can be addressed in the context of the applications they were designing. Similar results have been identified in [35].

6.4.2.9 Beyond Patterns

During their work with the patterns, some of the teams went beyond the definition provided by the cards and pointed out examples of applications of the patterns in software systems commonly used or identified associations between the issues addressed by the patterns and concrete implementations of software applications. Moreover, one of the teams identified possible relationships existing between patterns. For example, they considered the patterns Track history of collaboration, Collaborative undo, and Support versioning related to each other, even if they did not specify exactly in which way these patterns are related. A similar association was identified among the patterns Collaboration, always social, Annotate, and Customize collaboration.

6.5 Implications

The strategies described above trigger a set of implications to the use of design patterns in collaborative design processes. Such a discussion addresses both:

- The design of future tools to support the use of patterns in collaborative processes. Identifying possible strategies of use supports the definition and implementation of scenarios of use for tools addressing pattern-driven design.

- Teaching design patterns to novices and supporting them in grasping the full efficiency of the concept.

Customize Pattern Identification. A collection of patterns helps the communities using it in creating a pattern-oriented vocabulary, customized to the community and able to associate each pattern with a set of keywords. This is a simplification meant to support members of the same community in better understanding each other and in getting to more efficient communication processes. As observed throughout the study, the situations in which the participants would refer to a pattern by its full name were extremely rare, most of the times they either directly pointed to a pattern or used associated keywords to refer to it. A tool able to support designers in working with design patterns should consider such issues and allow its users to both reference a pattern by a set of keywords and contribute to a vocabulary able to represent the collection of patterns managed by the tool. Such a vocabulary would be a dynamic entity able to evolve according to the community managing it.

Signal Patterns. A collection of design patterns is a repository of knowledge. Judging by the collections of patterns available today, such a repository could have sizes larger than one individual can manage. In such cases, it is required that more than one individual would browse the repository and mark in some way those patterns relevant to the collaborative design process as a whole. The study showed that even when small size collections (i.e. 15 patterns) are used, such division of labour seamlessly occurs even in small size teams. The design of a pattern-oriented tool support should include mechanisms of signalling patterns among collaborating designers. Flagging relevant patterns is one way to do it. Recommending patterns based on one's expertise is another solution.

Search - Analyse - Apply. Searching in a pattern repository proved to be common. Even more, the participants in the study perceived searching as the most common action they performed. Surely, such a task can be more cumbersome and give the impression it is more often performed. For larger repositories of inter-related patterns especially, but also for more restricted collections, searching can be customized such that different types of queries can be performed on the repository. A more detailed discussion on such queries is presented in Section 5.6.1. Analysing a pattern in order to understand how it can be applied proved to be very common. Surely, supporting discussions among designers on the basis of the patterns they are using should be one of the goals of any tool addressing this area. A specialized tool should also support workflows such as: a) querying the pattern repository in search for patterns relevant in a specific context, b) reasoning on the applicability of the results of the query, c) reiterating the query based on the results of such reasoning, d) applying the solution best suited.

The Pattern Collection as a Checklist. A collection of patterns basically describes a set of fundamental problems in a specific domain and best practices to tackle such problems. Having such a collection at hand allows the evaluation of design results on the basis of the patterns. In other words, a collection of patterns can ensure that no fundamental issue in the design domain targeted had been missed. Surely, the goal in design is not making use of all such patterns, but ensuring that those relevant are used and that those unused are not suited for that particular design. Results above point out that novice designers view such checklists as powerful support tools for evaluating their own work and make extensive use of them.

Patterns as Startup Tools. Getting familiar with a set of patterns targeting a domain prior to initiating a design process supports getting an overall picture with respect to issues to be addressed further on in the process. This would particularly help novices or designers joining an already working team. In the case described above, the majority of the participants were novices and those with more than 3 years experience have never designed applications targeting synchronous collaboration. For them this domain was a novelty and therefore they considered the patterns to be an efficient introduction to it. Surely, larger repositories of patterns can only be browsed (and not thoroughly studied) and it is for this reason that any tool supporting designers working with patterns should provide its users with visualization techniques able to allow browsing such structures.

Mark the Use. Documentation is often portrayed as a cumbersome yet needed design activity. Seen from this perspective, patterns can be used as documenting tools in that design results can be enhanced with information on the patterns used throughout the process. Marking the use of a pattern on the design result points to the detailed documentation of the problem faced at that specific point in the design and at the complete description of the solution applied. Surely, the efficiency of such types of documentations needs to be studied over a longer period of time and involving designers, but judging by the results presented above such a technique looks promising. Participants who marked the patterns used on their design, found it much easier to go back to previous steps of their work and understand the implications of past decisions on the current context.

What do you mean? Collaborative design processes are often marked by misunderstandings among the collaborating designers. Such misunderstandings come from various reasons, such as collaborators have different backgrounds and expertise [88], the domain addressed by the design process is new to them, they do not make use of any shared representation of knowledge [98], [13]. As the study described above indicates, design patterns are forms of knowledge representation to which the participants often referred, and which were the support of extensive negotiations and debate. Integrating such support in collaborative design processes not only allows the representation of shared knowledge, but it also makes it available to others in a form which allows the creation of communication bridges among different collaborating designers.

Beyond Patterns. The application of a pattern might lead to the need of considering other related patterns. Designing a tool supporting the work with patterns should consider such an issue and include a mechanism able to recommend related patterns once a specific pattern has been used. Related patterns can be organized and managed as pattern languages, a detailed discussion on that being provided in Chapter 5.

6.6 Threats to Validity

Even if they have been sporadically criticized for "offering a poor basis for generalization" [34], case studies are powerful empirical methods used mainly for exploratory investigations. Using them at their full potential implies defining their objectives, the criteria for interpreting the findings, and their limitations [44], [102]. The latter consist in exploring and identifying the validity of the design and the results of the case study. One of the limitations of the case study presented above refers to not involving a sufficiently large number of professional designers. The small percentage of experienced designers (12% of the participants had more than 3 years experience in software design) is not convincing enough for any generalization of the results to professional designers. Therefore, all the implications the results trigger address mainly novice software designers. The patterns provided to the designers addressed a particular area – the design of synchronous collaborative applications. Moreover, the collection contained a relatively small number of patterns – 15. However, as support for the findings brought to light by the case study, the results presented in [35] identify several similar points even if the collection of patterns used by the authors addressed web design and contained 22 design patterns. In [35], Diaz et al. identified the "Read one-by one" browsing strategy and defined it as "participants went through all the patterns as a first strategy to identify candidates and look for ideas". Also, the web patterns proved to be intuitive and easily understood by the designers involved.

Some of the implications of the results described above are addressed through the design of CACE [62], a tool to support "pattern-based design of collaboration processes following the Collaboration Engineering approach". The tool is mainly addressing collaboration engineers and it supports the analysis of a collaborative task, the decomposition of a collaborative process, the visualization and the validation of a collaboration process flow. The goal of the tool however focuses on thinkLets - "a codified packet of facilitation skill that can be applied by practitioners to achieve predictable, repeatable patterns of collaboration, such as divergence or convergence"[27]. However, as opposed to thinkLets, design patterns are design tools meant for externalizing and sharing best practices in various areas of design.

Strong conclusions with respect to generalizations of the use of design patterns in collaborative design processes ask for further empirical work. Nevertheless, this work aims at bringing more knowledge to the matter and provides a starting point for further understanding and investigation.

—

In this chapter, I described a case study designed to bring some understanding on the ways design patterns are used in collaborative contexts. Having to complete a design task collaboratively, teams of novice designers were provided with a collection of design patterns and were asked to use the collection through the design process. The chapter discusses the results obtain from several data sources including the transcripts of the conversations the participants had, the questionnaire each participant filled in at the end of each workshop, the notes takes by the facilitator present during the workshops, and the design results of the teams processes.

The teams made use of the patterns even if not all the teams considered them fundamental to their work. They found the patterns' representation accessible and somehow fun to use which, in addition to the team work, motivated them to get engaged in the whole process. According to their feedback, searching was the most commonly performed action with respect to the patterns. In reality, they mostly discussed the patterns, most of the knowledge they've build around them coming from exchanging ideas and communicating. During the participants' working with the patterns, some common strategies they've developed came to light. As also described in [32], the patterns were seen as checklists by some of the teams, being walked-through to evaluate whether they were all considered. Other teams decided to mark the use of the patterns on their design results and this proved to be an efficient way to document their design decisions. The teams turned to the patterns when in doubt, or when searching for ideas. The patterns helped the teams validate their decisions or get more confidence in the path they were following. Some of the teams were somehow selective in terms of the patterns, filtering those they considered fundamental and only using those. Others, on the other hand, were completely driven by the patterns and made extensive use of them.

Chapter 7

Conclusions: Summary, Contributions, Future Directions

The goal of this thesis has been twofold. On one hand, I aimed at bringing methodological support to design pattern research in answer to the scarce landscape of methods and techniques for both identifying design patterns in interaction design and generating pattern languages based on existing collections of patterns. I focused mainly on one area of interaction design, i.e. the design of applications addressing synchronous collaboration, and I targeted four domains in the area, i.e. drawing, text editing, searching, and game solving. On the other hand, I was interested in better understanding how design patterns are used and what is the impact of using them in collaborative design processes. I first focused on the collaborative processes involving novice designers, aiming to correlate the findings from this initial study with those obtained after investigating similar processes involving experienced users of patterns.

7.1 Summary of the Thesis

Related work and background. I started by framing the landscape I am looking into and this comprises three major areas: CSCW, design pattern research, and creativity in design. CSCW helped me understand what synchronous collaboration refers to, how the concept evolved over time and how it differs from other collaboration modes. Also, I looked into the documented challenges faced in the design of software applications which support synchronous collaboration, classifying these challenges into several categories, such as technology supporting CSCW, coordination and conflict, communication, notifications and awareness, interruptions, the social side of CSCW, annotations, and roles in CSCW.

My interest in design pattern research led me to first investigate how the concept of design pattern appeared and how it evolved. Getting aware of the existence of multiple design pattern collections, I proceeded to reviewing these collections, trying to understand what design areas they address and how they are defined and described. The results of this review are synthesised and classified, nine areas being identified as associated with several collections of design patterns. Moreover, several templates for defining design patterns are used across collections, a synthesis of these templates and where they are used is described in Section 2.2.3. Going through the available collections and consulting literature in the matter, I searched for methodological support in identifying patterns only to discover that methods and techniques addressing such a goal are scarce and somewhat lacking the possibility of being generalized.

The concept of design pattern is more than often related to the concept of pattern language, since design patterns are never described independently from each other. Reviewing the literature in the matter, I addressed pattern languages, pointing out their documented advantages over plain collections of patterns. Similarly to the identification of design patterns, the methodological landscape with respect to pattern languages proved to be scarce. Pattern authors describe pattern languages without any indication to the way such languages are generated. Surely, experience plays an important role in first identifying patterns and then finding relationships between them, connecting them in a pattern language structure. However, such generation processes can not be replicated or even evaluated in the absence of a method describing them.

Lastly, having in mind the second goal of this thesis (i.e. understanding the behaviour around the collaborative use of design patterns), I went through some documented uses of patterns. The results of this review pointed out that patterns are specifically useful in teaching, design and more particularly participatory design, supporting the involvement of users in design processes. Even if these documented studies brought to light specific behavioural features observed during the use of patterns, many questions have been left unanswered or have not even been asked.

Much of the work described in this thesis revolves around interaction design. Therefore, I was particularly interested in models and creative techniques to support interaction design processes. I needed such models and techniques as tools for structuring and organizing what is left to be described.

—

Synchronous Collaborative Processes and Tools. I am particularly interested in design patterns for the design of applications which support synchronous collaboration. It is for this reason that I dedicate the third chapter to the domains of synchronous collaboration I am addressing throughout this thesis. I am mainly referring to drawing, text editing, searching, and game solving, providing for each of these domains the motivation for choosing it and the detailed descriptions of the tools existing out there which support users working together in each of these domains.

—

Identifying patterns. I further asked how are patterns identified in interaction design and I defined a structured method for such a purpose, making use of the limited results of the literature review as well. The method is defined as a two-step process which uses both a) the results of a series of workshops organized with designers and b) the results of the analysis of a set of applications addressing the area of the pattern mining. Both sets of results are considered in identifying the most recurring design issues in such design processes. These recurring issues are further documented in the form of design patterns, being validated by similar patterns described in the literature. To summarize, I identified 15 patterns after running 9 workshops with 50 participants and analysing 20 software applications.

—

Relating patterns. Next, I defined a method for identifying relationships between the patterns in an existing collection. The main idea behind refers to representing the domain the patterns address in the form of an ontology, identifying the set of concepts defining it and the relationships between these (less abstract) concepts. Such a representation further triggers the generation of a pattern language structure. In my attempt of applying the method, I realized that much of the process is subject to automation. Therefore, I proceeded to the design and the implementation of a supporting tool able to automatically apply the method and output as result a pattern language. This tool also allows querying the pattern language structure, resembling a search engine only that a search engine localized to the repository of patterns represented by the language. In testing the tool I used two collections of patterns: the one described in Chapter 4 and the collection proposed by Jenifer Tidwell addressing web design [103].

—

Evaluating Pattern. The results of the case study described in Chapter 6 provide some insight into the matter of using design patterns. The study aimed at measuring how understandable patterns are for novice designers and then, investigate how they use patterns in collaborative design processes. 18 teams participated in the study, using the collection of patterns described in Chapter 4. The description of the results focuses on all the data sources used:

- A facilitator observed all the teams and took notes of their interactions. Direct observation brought to light several issues to be further explored, such as the possibility of identifying common strategies participants used in working with the patterns.

- Each participant provided his/her feedback through a questionnaire. The results from the questionnaires show that the patterns were easy to understand and that the participants perceived searching for patterns as the most common action they performed.

- All the conversations of the teams were recorded, transcribed and further coded according to a defined coding scheme. The coding results come to point out that communication was of major importance in the processes recorded and also allowed the identification of specific action patterns in the participants' behaviour with respect to the use of the design patterns.

In the sections that follow, I discuss the contributions of this work and outline future research directions.

7.2 Contributions and Discussion

This work primarily impacts design pattern research at a methodological, theoretical, practical and empirical level. Secondarily, the findings described throughout the thesis inform behavioural research and human-computer interaction. Table 7.1 summarizes these contributions.

This work aims at describing the overall landscape in design pattern research through reviews of the available collections of design patterns, of the template definitions used for describing patterns, and of previous work in identifying and evaluating design patterns (Section 2.2). In addition, being particularly interested in patterns for the design of synchronous applications, I focused on the CSCW literature addressing synchronous collaboration and specifically on the documented challenges and concerns in the design of applications to support such collaboration mode. Multiple domains are subject to synchronous collaboration, those of interest to this work being drawing, searching, text editing, and game solving. Even if most of the tools targeting collaborative processes are designed to support asynchronous modes of collaboration, present efforts have acknowledged the need for applications addressing users working together in the same time (i.e. synchronously). A brief review of such applications is described in Chapter 3.

There are four questions this thesis addresses and the contributions brought to each question's area are discussed below:

Ch.	Contribution	Type
2	Review of documented challenges in designing synchronous applications	theoretical
2	Review of available collections of design patterns	theoretical
2	Review of previous work in identifying and evaluating patterns	theoretical
3	Review of tools supporting synchronous collaboration in drawing, searching, text editing, and game solving	theoretical
4	Development and application of a method for identifying design pattern for interaction design	methodological, empirical
4	A collection of patterns for the design of synchronous collaboration	theoretical
5	Development and application of a method for pattern language generation	methodological, practical
5	Development of a proof-of-concept tool able to support both the method's application and the resulting pattern language's querying	practical
5	Testing the tool using as input two available collections of patterns	practical
6	Investigation of the impact of using design patterns in collaborative design involving novices	empirical
6	Identification of behavioural strategies developed by novices while using patterns collaboratively	empirical
6	Discussion of the implications of the evaluation and of its results	theoretical, practical

Table 7.1: Contributions brought by this thesis

- *How to identify design patterns for interaction design?*
  Methods for identifying design patterns in interaction design are classified as inductive and deductive methods [15]. Whether moving from generalizations to specifics or vice-versa, their goal is to identify recurring design problems and document their proven solutions. The literature in the matter points to some examples of such methods, most of them however based on the experience of the designers involved. A systematic pattern development cycle targeting the design of e-learning systems is described in [83] and makes use of reverse engineering techniques. Since no generalization of this method is described and considering the fact that interaction design does not always allow the application of reverse engineering techniques, the present efforts address the definition of a design pattern mining method to be used for interaction design processes. Chapter 4 describes such a method defined by the correlation of results collected from both design workshops involving designers and from the analysis of existing applications used in the area of the pattern mining. Surely, the evaluation of such a method needs a large number of application cycles. This thesis illustrates some of them, the application of the pattern mining method leading to the identification of 15 design patterns for the design of synchronous applications, documented in 4.4. These patterns are not entirely novel, but they are validated by and help validating similar work done in this area [94].

- *How to generate pattern languages from collections of design pattern?*
  The advantages of a pattern language over a collection of patterns is largely admitted, pattern authors structuring their patterns in pattern languages [68], [90]. Even so, pattern authors rarely describe the process by which they reach the languages, basing themselves mostly on experience. Classifications of possible relationships between patterns have been described in [25] and [105], but very little work has been done in actually framing a method which would support relating patterns based on such relationships. To answer this gap, chapter 5 illustrates a method for relating design patterns comprised in a collection for generating a pattern language structure. The application of the pattern language generation method uses two test cases - the collection of patterns described in Chapter 4

and the collection proposed by Jenifer Tidwell addressing web design [103].

- *What tool support is needed for the application of the methods described above?*
  My work also comprises the implementation of a prototypical tool designed to support both pattern writers interested in automatically generating a pattern language out of the collection of patterns they wrote and pattern users. The tool follows the pattern language generation method described in Chapter 5 outputting a pattern language in a format interpretable by the Medusa 5.2 graph visualization tool. Moreover, the tool allows the execution of six types of queries, acting as a search engine on the repository of patterns represented by the language.

- *How are design patterns used in collaborative design processes?*, *How do designers perceive the use of design patterns in collaborative design processes?*
  Previous findings with respect to the ways design patterns are used point out things such as patterns are useful tools for teaching design principles [23], patterns help designers get familiar to a new domain [29], a collection of patterns can be used as a checklist [31], or browsed in a read one-by-one manner [35]. The aim of this work is to build on these findings, the case study described in Chapter 6 contributing to a better understanding of the way patterns are used by novice designers in collaborative contexts. As a first general observation, the patterns pose no difficulty to the participants in the study in terms of understandability, validating that the simplified format chosen to present the patterns provided just enough information. When working with patterns and especially when novice designers are involved, less (information) is more. Also because results from the case study showed that reading a pattern completely was rare, most of the participants' understanding of the patterns coming from them discussing the patterns together and reading fragments of what the description of each pattern provided.

  The participants considered searching as the most recurring action they performed. This, however, comes to contradict the results drawn from the actual transcripts which point out that communication had a far more important role in their processes. Now, how is this relevant? On one hand, this points out a gap between what the participants perceived they were doing and what they actually did. Getting a better understanding of the strategies they used in working with the patterns definitely needs a closer look at the actual process and less attention to the feedback from the participants. On the other hand, the feedback from the participants indicates that searching is probably the most cumbersome action when using patterns since it was perceived as the most often performed. This leads to the belief that efficiently searching in a repository of patterns should be a main consideration for any tool designed to support pattern-oriented design.

Results also showed that once a pattern is referenced, in most cases it is further discussed. Brought to the attention of the entire team, the pattern becomes an object used for negotiation, common understanding, debate. It supports collaborating designers in co-creating a shared representation of a common ground to be used further on in their process. Also, encouraging enough is the indication that generating ideas based on the use of design patterns leads to generating more ideas relevant to the design process. Limited as it is, this is an indication that patterns support brainstorming processes, allowing their users to come up with new ideas on the basis of the issues addressed by the patterns.

Several strategies were brought to light by abstracting from the set of data sources gathered during the study. These strategies are a first attempt to model and describe the behaviour of novices while collaboratively using design patterns. Surely, more empirical studies are needed for deepening such aspects. However, the results of this study indicate several valuable findings, further pointing to new questions to ask and new paths to follow. Some of the implications brought by the current results address both the design of tools to support the use of patterns and teaching processes involving patterns.

7.3 Future Research Directions

Several aspects presented in this thesis are pointing to future research direction worth exploring. First, the methodological contributions require more empirical evaluation. Applying the pattern mining method in other areas with an eye on the efficiency and the validity of the obtained results is one of those directions. The method is, of course, subject to evolution. It might be improved or enhanced upon being further applied. Moreover, replicating the process of applying it in the area of synchronous collaborative design choosing different domains of collaboration would enhance the confidence in the method's efficiency and power and would probably lead to the identification of other patterns as well.

Similar observations hold true for the second method described by this work. Evaluating the method using as input other available collections of design patterns is a future goal. In parallel to such evaluations, the supporting tool would need to go through other iterations since any modification of the method asks for the tool to be updated accordingly. Efforts are required for the design of a more intuitive GUI, as well. Such designs ask for the involvement of users interested in providing feedback on the tool's efficiency and intuitiveness. At this point, the tool is meant as a proof of concept; it is, however, suited for other contexts of use as depicted in the scenarios describing its possible uses.

A critical limitation of the evaluation of the use of design patterns is the external validity of the laboratories studies. The design task participants were confronted with was constructed to simulate real-world design processes. Exploring the questions addressed by the case study using this artificial design task pointed out some interesting and valuable findings. However, it remains somewhat unclear how these questions would be answered in a real work environment, involving professional designers. In the field, professional designers base their design decisions on experience and intuition and might think of the patterns as best practices they are familiar with and apply even at an unconscious level (without specifically pointing out they do so). The questions would therefore translate into:

- How does experience impact the use of design patterns? Are experienced designers more prone to using patterns without making this explicit in any way?

- Are patterns more suited to be used in an organizational context? What benefits would they bring when used in such a context?
- What feedback would experienced designers provide when using design patterns in a collaborative context? Would their process be helped in any way? What kind of disadvantages would the enforced use of patterns bring?

Observing both novice and experienced designers would surely require different research methods. Investigating the behaviour of novices would require observing them in action, hence the workshops approach seems to be better suited. Experienced designers, having a vaster experience, are able to reflect on their own actions and behaviour. Therefore, interviews could prove more efficient in their case.

I described several implications of the results of the evaluation case study, each of these implications needing further investigation. As few hints on such investigations, consider:

- Identifying the efficiency of design patterns acting as documentation tools.
- Analysing the efficiency of using a collection of patterns for evaluating design results.
- Informing design processes of the usefulness of design patterns in training designers joining an already formed team or novice designers.

Glossary

association function function which directly maps a design issue to a set of keywords, hence implicitly to each keyword in the set.

collection group of design patterns without the specification of any relationships between these patterns.

CSCW Computer-Supported Cooperative Work.

design issue design idea, problem, concern, solution found useful, or/and any issue relevant to the design of the application.

design pattern way to document proven solutions to recurring design problems.

guideline structure which helps capturing design knowledge and supports the establishment of a clear design process.

HCI Human-Computer Interaction.

mockups very early prototypes made of cardboard or otherwise low-fidelity materials.

pattern language structure representing a collection of design patterns together with all the relationships between them.

problem design task provided during the workshops.

scenario-based design technique of using scenarios during design processes.

sketches tools for capturing preliminary observations and ideas.

synchronous collaboration collaboration mode which requires that geographically distributed or co-located work group members work together - supported by software - in developing and refining one commonly shared resource in the same time.

task description details on a problem for which a solution is needed.

use case description of the dialogue between the user and the system, modelling the interaction provided by the system.

References

[1] http://www.cs.kent.ac.uk/people/staff/saf/patterns/plml.html. 168

[2] http://quince.infragistics.com/html/home.aspx. 38

[3] http://tagsea.sourceforge.net/. 28

[4] Atwood M. E. Abraham, G. Patterns or claims: do they help in communicating design advice? *Proceedings of OZCHI '09*, 2009. 56

[5] Bailey B. P. Adamczyk, P. D. If not now, when?: the effects of interruption at different moments within task execution. *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '04)*, 2004. 24

[6] Nash J. C. Noel S. Adler, A. Evaluating and implementing a collaborative office document system. *Interact. Comput*, 2006. xv, 2, 85, 86, 87, 89

[7] C. Alexander. *The Oregon Experiments*. Oxford University Press, 1975. 4, 30

[8] C. Alexander. *The Timeless Way of Building*. Oxford University Press, 1979. 5, 30, 33

[9] Ishikawa S. Silverstein M. Alexander, C. *A pattern language: Towns, buildings, construction*. New York: Oxford University Press, 1977. 4, 30, 49, 51, 56, 166

[10] T.M. Amabile. *The Social Psychology of Creativity*. Springer-Verlag, New York, 1983. 59

[11] Ringel Morris M. Amershi, S. Cosearch: a system for co-located collaborative web search. *Proceeding of CHI '08*, 2010. xiv, 73, 75, 76

[12] Ringel Morris M. Moraveji N. Balakrishnan R. Toyama K. Amershi, S. Multiple mouse text entry for single-display groupware. *Proceedings of ACM conference on Computer supported cooperative work (CSCW '10)*, 2010. 20

[13] E. G. Arias and G. Fischer. Boundary objects: their role in articulating the task at hand and making information relevant to it. *Proceeding of International ICSC Symposium on Interactive and Collaborative Computing (ICC 2000)*, 2000. 268

[14] Fussell S. R. Hudson S. E. Avrahami, D. Im waiting: timing and responsiveness in semi-synchronous communication. *Proceeding of ACM conference on Computer supported cooperative work (CSCW '08)*, 2008. 23

[15] Rusman E. Poggi C. Baggetun, R. Design patterns for collaborative learning: From practice to theory and back. *Proceedings of International Conference on Educational Multimedia, Hypermedia and Telecommunications*, 2004. 53, 277

[16] L. Bannon. Cscw: Four characters in search for a context. 1989. 14, 17, 25

[17] N. Bjørn-Andersen Bannon, L. and B. Due-Thomsen. Computer support for co-operative work: An appraisal and critique. *EURINFO '88, Information Systems for Organizational Effectiveness*, 1988. 14

[18] M. Baskinger. Pencils before pixels: a primer in hand-generated sketching. *Interactions*, 2008. 62

[19] Heymann S.-Jacomy M. Bastian, M. Gephi: An open source software for exploring and manipulating networks. *International AAAI Conference on Weblogs and Social Media*, 2009. 169

[20] D. Tomasini M. Zancanaro G. Esposito P. Venuti A. Ben Sasson E. Gal Battocchi, F. Pianesi and P. L. Weiss. Collaborative puzzle game: a tabletop interactive game for fostering collaboration in children with autism spectrum disorders. *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS '09)*, 2009. xv, 91, 93, 95

[21] D. Beyer. Ccvisu: Automatic visual software decomposition. *ICSE'08*, 2008. 170

[22] J. Borchers. *A Pattern Approach to Interaction Design.* John Wiley Sons, Inc., 2001. 5, 31, 42, 51, 52, 56, 102, 166, 167, 168, 170

[23] J. Borchers. Teaching hci design patterns: Experience from two university courses. patterns in practice: A workshop for ui designers. *CHI2002*, 2002. 54, 278

[24] Civica R. Levialdi S. Orso L. Panizzi E. Trinchese R. Bottoni, P. Madcow: a multimedia digital annotation system. *Proceedings of the working conference on Advanced visual interfaces (AVI '04)*, 2004. 26

[25] Guerra E. de Lara J. Bottoni, P. A language-independent and formal approach to pattern-based modelling with support for composition and analysis. *Information and Software Technology*, 2010. 57, 277

[26] Star S. L. Bowers, G. Sorting things out: Classification and its consequences. 1999. 5

[27] Vreede G. Nunamaker J. Briggs, R. O. Collaboration engineering with thinklets to pursue sustained success with group support systems. *Journal of Management Information Systems*, 2007. 269

[28] J.M. Carroll. *Scenario-Based Design: Envisioning Work and Technology in System Development.* John Wiley Sons, Inc., 1995. 61, 103, 104, 211

[29] Hong J. I. Lin J. Prabaker M. K. Landay J. A. Liu A. L. Chung, E.S. Development and evaluation of emerging design patterns for ubiquitous computing. *Proceedings of DIS '04*, 2004. 42, 55, 278

[30] Malone E. Crumlish, C. *Designing Social Interfaces.* O'Reilly Media, Inc., 2009. 5, 40, 51, 142, 150

[31] Finlay J. Dearden, A. Pattern languages in hci: A critical review. *Human Computer Interaction Journal*, 2006. 206, 278

[32] Finlay J. Allgar E. Mcmanus B. Dearden, A. Using pattern languages in participatory design. *Proceedings of the Participatory Design Conference (PDC 2002)*, 2002. 55, 270

[33] Herbsleb J. D. Dekel, U. Pushing relevant artifact annotations in collaborative software development. *Proceedings of ACM conference on Computer supported cooperative work (CSCW '08)*, 2008. 26, 28

[34] S. Easterbrook S. M. Dewayne E. Perry, Elliott Sim. Case studies for software engineers. *Proceedings of the 26th International Conference on Software Engineering (ICSE '04)*, 2004. 269

[35] Rosson M.B. Aedo I. Carroll J.M. Diaz, P. Web design patterns: Investigating user goals and browsing strategies. *Proceedings of the International Symposium on End-User Development (IS-EUD '09)*, 2009. 44, 206, 265, 269, 278

[36] Pandolfo A. Bender W. DiMicco, J. M. Influencing group participation with a shared display. *Proceedings of ACM conference on Computer supported cooperative work (CSCW '04)*, 2004. 20

[37] Zhao Y. Peng T. Dong, J. A review of design pattern mining techniques. *International Journal of Software Engineering and Knowledge Engineering*, 2009. 53

[38] Moore R. J. Ducheneaut, N. The social side of gaming: a study of interaction patterns in a massively multiplayer online game. *Proceedings of ACM conference on Computer supported cooperative work (CSCW '04)*, 2004. 26

[39] C. A. et al. Ellis. Groupware: some issues and experiences. *Commun. ACM*, 1991. 60

[40] Golovchinsky et al. Cerchiamo: A collaborative exploratory search tool. *Proceedings of CSCW'08*, 2008. 81

[41] G. Fischer. Social creativity, symmetry of ignorance and meta-design. *Knowledge-Based Systems Journal*, 2000. 5

[42] G. Fischer. Social creativity: turning barriers into opportunities for collaborative design. *Proceedings of PDC'04*, 2004. 32, 60

[43] Ostwald J. Fischer, G. Knowledge communication in design communities. *Barriers and Biases in Computer-Mediated Knowledge Communication*, 2005. 2, 60, 61

[44] B. Flyvbjerg. Five misunderstandings about case-study research. *Qualitative Inquiry*, 2006. 269

[45] L. Bernareggi C. Fogli, D. Parasiliti Provenza. A design pattern language for accessible web sites. *Proceedings of AVI 2010*, 2010. 5, 43, 170

[46] R. Helm R. Johnson Vlissides J. Gamma, E. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995. 31, 34, 51, 57

[47] I. Graham. *A Pattern Language of Web Usability*. Addison-Wesley, 2003. 41, 51

[48] S. T. Gregory. On prototypes vs. mockups. *SIGSOFT Softw. Eng. Notes*, 1984. 63

[49] I. Greif. Remarks in panel discussion on "cscw: What does it mean?". *CSCW'88*, 1988. 13

[50] J. Grudin. Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM*, 1994. xiv, 14, 15, 17

[51] Graham T. C. N. Wolfe C. Wong N. de Alwis B. Gutwin, C. Gone but not forgotten: designing for disconnection in synchronous groupware. *Proceedings of ACM conference on Computer supported cooperative work (CSCW '10)*, 2010. 25

[52] Hoffmann M. Jahnke I. Kienle A. Kunau G. Loser K. Menold N. Herrmann, T. Concepts for usable patterns of groupware applications. *Proceedings of the ACM SIGGROUP conference on Supporting group work (GROUP '03)*, 2003. 206

[53] Jahnke I. Loser K. Herrmann, T. The role concept as a basis for designing community systems. *Proceedings of the ACM SIGGROUP conference on Supporting group work (GROUP '03)*, 2004. 28

[54] T. Herrmann. Support of collaborative creativity for co- located meetings, from cscw to web 2.0. *European Developments in Collaborative Design Computer Supported Cooperative Work*, 2010. 18

[55] Bork P. Hooper, S.D. Medusa: a simple tool for interaction graph analysis. *Bioinformatics Applications Note*, 2005. 169

[56] Frøkjær E. Hornbæk, K. Reading patterns and usability in visualizations of electronic documents. *ACM Trans. Comput.-Hum. Interact.*, 2003. 31, 52

[57] Papadopoulou S. Oster G. Norrie M. C. Ignat, C. L. Providing awareness in multi-synchronous collaboration without compromising privacy. *Proceedings of the ACM conference on Computer supported cooperative work (CSCW '08)*, 2008. 24

[58] Horvitz E. Iqbal, S. T. Notifications and awareness: a field study of alert usage and preferences. *Proceedings of the ACM conference on Computer supported cooperative work (CSCW '10)*, 2010. 24

[59] Whittaker S. Kalnikaite, V. Social summarization: does social feedback improve access to speech data? *Proceedings of the ACM conference on Computer supported cooperative work (CSCW '08)*, 2008. 25

[60] Kraut R. E. Kittur, A. Beyond wikipedia: coordination and conflict in online production groups. *Proceedings of the ACM conference on Computer supported cooperative work (CSCW '10)*, 2010. 21, 23

[61] Perry J. Squire K. Jan M.F. Steinkuehler C. Klopfer, E. Mystery at the museum: a collaborative game for museum education. *Proceedings of CSCL '05*, 2005. xv, 93, 94

[62] De Vreede G. J. Briggs R. O. Kolfschoten, G. L. Computer aided pattern-based collaboration process design: a computer aided collaboration engineering tool. *13th international conference on Groupware: design implementation, and use (CRIWG'07)*, 2007. 269

[63] Lukosch S. Verbraeck A. Valentin E. de Vreedea G.J. Kolfschoten, G. Cognitive learning efficiency through the use of design patterns in teaching. *Computers and Education*, 2010. 54

[64] Hitz M. Kruschitz, C. Human-computer interaction design patterns: Structure, methods, and tools. *International Journal on Advances in Software*, 2010. 5, 44

[65] Myers B. A. Landay, J. A. Sketching interfaces: Toward more human interface design. *Computer*, 2001. 63

[66] Kuhail M. Lauesen, S. Task descriptions versus use cases. *Springer Requirements Eng*, 2011. 4

[67] Schümmer T. Lukosch, S. Communicating design knowledge with groupware technology patterns: The case of shared object management. *Proceedings of the CRIWG 2004*, 2004. 39

[68] Schümmer T. Lukosch, S. The role of roles in computer-mediated interaction. *13th European Conference on Pattern Languages of Programs*, 2008. 28, 39, 160, 277

[69] Memisoglu M. Engelke T. Streitz N. Magerkurth, C. Towards the next generation of tabletop gaming experiences. *Proceedings of the GI '04*, 2004. 91, 97

[70] Johnston L. J. Mahemoff, M. J. Pattern languages of usability: An investigation of alternative approaches. *Proceedings of the APCHI 98*, 1998. 41, 170

[71] Baker A. Dempsey M. Navarro E. van der Hoek A. Mangano, N. Software design sketching with calico. *Proceedings of the IEEE/ACM international conference on Automated software engineering (ASE '10)*, 2010. 59, 62

[72] Avouris N. Kahrimanis G. Margaritis, M. On supporting users' reflection during small groups synchronous collaboration. *Proceedings of the 12th International Workshop on Groupware, CRIWG 2006*, 2006. 3, 68, 69

[73] Borges M.R.S. Araujo R.M. Meire, A.P. Supporting collaborative drawing with the mask versioning mechanism. *Proceedings of 9th International Workshop on Groupware*, 2003. xiv, 70, 72

[74] Neu D. Shi Q. Menkov, V. Antworld: A collaborative web search tool. *Proceedings of the Workshop on Distributed Communities on the Web (DCW '00)*, 2000. xv, 83, 84

[75] Doble J. Meszaros, G. A pattern language for pattern writing. *Proceedings of International Conference on Pattern languages of program design (1997)*, 1997. 131, 164

[76] Weir C. Noble, J. *Small Memory Software: Patterns for systems with limited memory (Software Patterns Series)*. Addison-Wesley Professional, 2000. 158

[77] A. Osborn. *Applied Imagination: Principles and Procedures of Creative Problem Solving*. New York, New York: Charles Scribner's Sons, 1953. 59

[78] O'Brien E. Ringel Morris M. Winograd T. Piper, A.M. Sides: a cooperative tabletop computer game for social skills development. *Proceedings of CSCW '06*, 2006. xv, 95, 96

[79] Winckler M. Limbourg Q. Pontico, F. Organizing user interface patterns for e-government applications. *Proc. Engineering Interactive Systems (EIS 2008)*, 2008. 40, 52

[80] M.D. Qian. Collaborative design with netdraw. *Proceedings of Computer Aided Architectural Design Futures '99*, 1999. xiv, 68, 69, 71

[81] T.M.H. Reenskaug. The model-view-controller (mvc) - its past and present. *JavaZONE, Oslo*, 2003. 31

[82] Mackay B. Watters C. R. Inkpen K. M. Reilly, D. F. Small details: using one device to navigate together. *Proceedings of the ACM conference on Computer supported cooperative work (CSCW '08)*, 2008. 20

[83] Georgiakakis P. Dimitriadis Y. Retalis, S. Eliciting design patterns for e-learning systems. *Computer Science Education*, 2006. 53, 277

[84] Horvitz E. Ringel Morris, M. Searchtogether: an interface for collaborative web search. *Proceedings of UIST '07*, 2007. xiv, 2, 73, 75, 76, 79, 80

[85] Lombardo J. Wigdor D. Ringel Morris, M. Wesearch: supporting collaborative search and sensemaking on a tabletop display. *Proceedings of the ACM conference on Computer supported cooperative work (CSCW '10)*, 2010. xv, 2, 83, 85

[86] Morris D. Winograd T. Ringel Morris, M. Individual audio channels with single display groupware: effects on communication and task strategy. *Proceedings of the ACM conference on Computer supported cooperative work (CSCW '04)*, 2004. 20, 21

[87] Ryall K. Shen C. Forlines C. Vernier F. Ringel Morris, M. Beyond "social protocols": multi-user coordination policies for co-located groupware. *Proceedings of the ACM conference on Computer supported cooperative work (CSCW '04)*, 2004. 22

[88] H. Rittel. Second-generation design methods. *Developments in Design Methodology*, 1984. 268

[89] Prabaker M. K Abowd G.D. Landay J. A. Saponas, T.S. The impact of pre-patterns on the design of digital home applications. *Proceedings of DIS '06*, 2006. 55

[90] Jachna T. Schadewitz, N. Comparing inductive and deductive methodologies for design patterns identification and articulation. *International Design Research Conference*, 2007. 5, 54, 277

[91] N. Schadewitz. Design patterns for cross-cultural collaboration. *International Journal of Design*, 2009. 39

[92] D. Schön. *The reflective practitioner: How professionals think in action.* NY: Basic Books, 1983. 62

[93] D. Schuler. A pattern language for living communication. *Proceedings of PDC'02*, 2002. 40, 51, 54

[94] Lukosch S. Schummer, T. *Patterns for Computer-Mediated Interaction.* John Wiley Sons, Ltd, 2007. xiv, 5, 38, 45, 46, 51, 133, 135, 137, 140, 142, 146, 148, 150, 152, 277

[95] Grant K. D. Mandryk R. L. Scott, S. D. System guidelines for co-located, collaborative work on a tabletop display. *Proceedings ECSCW'03*, 2003. 19

[96] A. Sellen and R. Harper. *The Myth of the Paperless Offices*. MIT Press, MA, 2002. 62

[97] Marchionini G. Kelly D. Shah, C. Learning design principles for a collaborative information seeking system. *Proceedings of CHI '09*, 2009. xiv, 2, 76, 78

[98] B. Shneiderman. Creating creativity: user interfaces for supporting innovation. *ACM Trans. Comput.-Hum. Interact*, 2000. 60, 268

[99] B. Shneiderman. Creativity support tools: Accelerating discovery and innovation. *CACM*, 2007. 5

[100] S. L. Star. The structure of ill-structured solutions: Boundary objects and heterogeneous distributed problem solving. *Distributed Artificial Intelligence*, 1990. 5

[101] J. Surowiecki. *Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*. Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations, 2004. 76

[102] W. Tellis. Introduction to case study. *The Qualitative Report*, 1997. 269

[103] J. Tidwell. *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly Media, 2005. 5, 36, 51, 197, 274, 278

[104] S. G. Turner. A case study using scenario-based design tools and techniques in the formative evaluation stage of instructional design: Prototype evaluation and redesign of a web-enhanced course interface. *Ph.D. Thesis*, 1998. 61, 62

[105] van der Veer G. C. van Welie, M. Pattern languages in interaction design: Structure and organization. *Proceedings of Interact'03*, 2003. 51, 57, 277

[106] Sapp C. Mathews M. Verplank, B. A course on controllers. 2001. 62

[107] Heylen D. Nijholt A. Vyas, D. and G. van der Veer. Collaborative practices that support creativity in design. *Proc. of ECSCW '09*, 2009. 62

[108] G. Wallas. *The Art of Thought*. New York, Harcourt, Brace and Company, 1926. 59, 104

[109] O'Neill E. Warr, A. Understanding design as a social creative process. *Proceedings of the 5th conference on Creativity cognition (C C '05)*, 2005. 63

[110] Gennari J. H. Weng, C. Asynchronous collaborative writing through annotations. *Proceedings of the 2004 ACM conference on CSCW'04*, 2004. 26

[111] Mor Y. Winters, N. Dealing with abstraction: Case study generalization as a method for eliciting design patterns. *Computers in Human Behavior*, 2009. 53, 54

[112] L. Young-Jin. Vissearch: a collaborative web searching environment. *Comput. Educ*, 2005. xiv, 81, 82

[113] U. Zdun. Capturing design knowledge. *IEEE Software*, 2009. 4

[114] W. Zimmer. Relationships between design patterns. *Pattern Languages of Program Design*, 1995. 57

Appendix 1 - Urban and architectural design patterns proposed by Christopher Alexander

1. *INDEPENDENT REGIONS*
2. *THE DISTRIBUTION OF TOWNS*
3. *CITY COUNTRY FINGERS*
4. *ARCHITECTURAL VALLEYS*
5. *LACE OF COUNTRY STREETS*
6. *COUNTRY TOWNS*
7. *THE COUNTRYSIDE*
8. *MOSAIC OF SUBCULTURES*
9. *SCATTERED WORK*
10. *MAGIC OF THE CITY*
11. *LOCAL TRANSPORT AREA*
12. *COMMUNITY OF 7000*
13. *SUBCULTURE BOUNDARY*
14. *IDENTIFIABLE NEIGHBORHOOD*
15. *NEIGHBORHOOD BOUNDARY*
16. *WEB OF PUBLIC TRANSPORTATION*
17. *RING ROADS*
18. *NETWORK OF LEARNING*
19. *WEB OF SHOPPING*
20. *MINI-BUSES*
21. *FOUR-STORY LIMIT*
22. *NINE PER CENT PARKING*
23. *PARALLEL ROADS*

24. *SACRED SITES*
25. *ACCESS TO WATER*
26. *LIFE CYCLE*
27. *MEN AND WOMEN*
28. *ECCENTRIC NUCLEUS*
29. *DENSITY RINGS*
30. *ACTIVITY NODES*
31. *PROMENADE*
32. *SHOPPING STREET*
33. *NIGHT LIFE*
34. *INTERCHANGE*
35. *HOUSEHOLD MIX*
36. *DEGREES OF PUBLICNESS*
37. *HOUSE CLUSTER*
38. *ROW HOUSES*
39. *HOUSING HILL*
40. *OLD PEOPLE EVERYWHERE*
41. *WORK COMMUNITY*
42. *INDUSTRIAL RIBBON*
43. *UNIVERSITY AS A MARKET-PLACE*
44. *LOCAL TOWN HALL*
45. *NECKLACE OF COMMUNITY PROJECTS*
46. *MARKET OF MANY SHOPS*
47. *HEALTH CENTER*
48. *HOUSING IN BETWEEN*
49. *LOOPED LOCAL ROADS*
50. *T JUNCTIONS*
51. *GREEN STREETS*
52. *NETWORK OF PATHS AND CARS*
53. *MAIN GATEWAYS*
54. *ROAD CROSSING*
55. *RAISED WALK*
56. *BIKE PATHS AND RACKS*
57. *CHILDREN IN THE CITY*
58. *CARNIVAL*
59. *QUIET BACKS*
60. *ACCESSIBLE GREEN*
61. *SMALL PUBLIS SQUARES*
62. *HIGH PLACES*
63. *DANSING IN THE STREET*
64. *POOLS AND STREAMS*

65. *BIRTH PLACES*
66. *HOLY GROUND*
67. *COMMON LAND*
68. *CONNECTED PLAY*
69. *PUBLIC OUTDOOR ROOM*
70. *GRAVE SITES*
71. *STILL WATER*
72. *LOCAL SPORTS*
73. *ADVENTURE PLAYGROUND*
74. *ANIMALS*
75. *THE FAMILY*
76. *HOUSE FOR A SMALL FAMILY*
77. *HOUSE FOR A COUPLE*
78. *HOUSE FOR ONE PERSON*
79. *YOUR OWN HOME*
80. *SELF-GOVERNING WORKSHOPS AND OFFICES*
81. *SMALL SERVICES WITHOUT RED TAPE*
82. *OFFICE CONNECTIONS*
83. *MASTER AND APPRENTICES*
84. *TEENAGE SOCIETY*
85. *SHOPFRONT SCHOOLS*
86. *CHILDREN'S HOME*
87. *INDIVIDUAL OWNED SHOPS*
88. *STREET CAFÉ*
89. *CORNER GROCERY*
90. *BEER HALL*
91. *TRAVELER'S INN*
92. *BUS STOP*
93. *FOOD STANDS*
94. *SLEEPING IN PUBLIC*
95. *BUILDING COMPLEX*
96. *NUMBER OF STORIES*
97. *SHIELDED PARKING*
98. *CIRCULATION REALMS*
99. *MAIN BUILDING*
100. *PEDESTRIAN STREET*
101. *BUILDING THOROUGHFARE*
102. *FAMILY OF ENTRANCES*
103. *SMALL PARKING LOTS*
104. *SITE REPAIR*
105. *SOUTH FACING OUTDOORS*
106. *POSITIVE OUTDOOR SPACE*
107. *WINGS OF LIGHT*

108. *CONNECTED BUILDINGS*
109. *LONG THIN HOUSE*
110. *MAIN ENTRANCE*
111. *HALF-HIDDEN GARDEN*
112. *ENTRANCE TRANSITION*
113. *CAR CONNECTION*
114. *HIERARCHY OF OPEN SPACE*
115. *COURTYARDS WHICH LIVE*
116. *CASCADE OF ROOFS*
117. *SHELTERING ROOF*
118. *ROOF GARDEN*
119. *ARCADES*
120. *PATHS AND GOALS*
121. *PATH SHAPE*
122. *BUILDING FRONTS*
123. *PEDESTRIAN DENSITY*
124. *ACTIVITY POCKETS*
125. *STAIR SEATS*
126. *SOMETHING ROUGHLY IN THE MIDDLE*
127. *INTIMACY GRADIENT*
128. *INDOOR SUNLIGHT*
129. *COMMON AREAS AT THE HEART*
130. *ENTRANCE ROOM*
131. *THE FLOW THROUGH ROOMS*
132. *SHORT PASSAGES*
133. *STAIRCASE AS A STAGE*
134. *ZEN VIEW*
135. *TAPESTRY OF LIGHT AND DARK*
136. *COUPLE'S REALM*
137. *CHILDREN'S REALM*
138. *SLEEPING TO THE EAST*
139. *FARMHOUSE KITCHEN*
140. *PRIVATE TERRACE ON THE STREET*
141. *A ROOM OF ONE'S OWN*
142. *SEQUENCE OF SITTING SPACES*
143. *BED CLUSTER*
144. *BATHING ROOM*
145. *BULK STORAGE*
146. *FLEXIBLE OFFICE SPACE*
147. *COMMUNAL EATING*
148. *SMALL WORK GROUPS*

149. *RECEPTION WELCOMES YOU*
150. *SMALL MEETING ROOMS*
151. *HALF-PRIVATE OFFICE*
152. *ROOMS TO RENT*
153. *TEENAGER'S COTTAGE*
154. *OLD AGE COTTAGE*
155. *SETTLED WORK*
156. *HOME WORKSHOP*
157. *OPEN STAIRS*
158. *LIGHT ON TWO SIDES OF EVERY ROOM*
159. *BUILDING EDGE*
160. *SUNNY PLACE*
161. *NORTH FACE*
162. *OUTDOOR ROOM*
163. *STREET WINDOWS*
164. *OPENING TO THE STREET*
165. *GALLERY SURROUND*
166. *SIX-FOOT BALCONY*
167. *CONNECTION TO THE EARTH*
168. *TERRACED SLOPE*
169. *FRUIT TREES*
170. *TREE PLACES*
171. *GARDEN GROWING WILD*
172. *GARDEN WALL*
173. *TRELISSED WALK*
174. *GREENHOUSE*
175. *GARDEN SEAT*
176. *VEGETABLE GARDEN*
177. *COMPOST*
178. *ALCOVES*
179. *WINDOW PLACE*
180. *THE FIRE*
181. *EATING ATMOSPHERE*
182. *WORKSPACE ENCLOSURE*
183. *COOKING LAYOUT*
184. *SITTING CIRCLE*
185. *COMMUNAL SLEEPING*
186. *MARRIAGE BED*
187. *BED ALCOVE*
188. *DRESSING ROOM*
189. *CEILING HEIGHT VARIETY*
190. *THE SHAPE OF INDOOR SPACE*
191. *WINDOWS OVERLOOKING LIFE*

192. *HALF-OPEN WALL*
193. *INTERIOR WINDOWS*
194. *STAIRCASE VOLUME*
195. *CORNER DOORS*
196. *THICK WALLS*
197. *CLOSETS BETWEEN ROOMS*
198. *SUNNY COUNTER*
199. *OPEN SHELVES*
200. *WAIST-HIGH SHELF*
201. *BUILT-IN SEATS*
202. *CHILD CAVES*
203. *SECRET PLACE*
204. *STRUCTURE FOLLOWS SOCIAL PLACES*
205. *EFFICIENT STRUCTURE*
206. *GOOD MATERIALS*
207. *GRADUAL STIFFENING*
208. *ROOF LAYOUT*
209. *FLOOR AND CEILING LAYOUT*
210. *THICKENING THE OUTER WALLS*
211. *COLUMNS AT THE CORNERS*
212. *FINAL COLUMN DISTRIBUTION*
213. *ROOF FOUNDATION*
214. *GROUND FLOOR SLAB*
215. *BOX COLUMNS*
216. *PERIMETER BEAMS*
217. *WALL MEMBRANES*
218. *FLOOR-CEILING VAULTS*
219. *ROOF VAULTS*
220. *NATURAL DOORS AND WINDOWS*
221. *LOW SILL*
222. *DEEP REVEALS*
223. *LOW DOORWAY*
224. *FRAMES AS THICKENED EDGES*
225. *COLUMN PLACE*
226. *COLUMN CONNECTION*
227. *STAIR VAULT*
228. *DUCT SPACE*
229. *RADIANT HEAT*
230. *DORMER WINDOWS*
231. *ROOF CAPS*
232. *FLOOR SURFACE*

233. *LAPPED OUTSIDE WALLS*

234. *SOFT INSIDE WALLS*

235. *WINDOWS WHICH OPEN WIDE*

236. *SOLID DOORS WITH GLASS*

237. *FILTERED LIGHT*

238. *SMALL PANES*

239. *HALF-INCH TRIM*

240. *SEAT SPOTS*

241. *FRONT DOOR BENCH*

242. *SITTING WALL*

243. *CANVAS ROOFS*

244. *RAISED FLOWERS*

245. *CLIMBING PLANTS*

246. *PAVING WITH CRACKS BETWEEN THE STONE*

247. *SOFT TILE AND BRICK*

248. *ORNAMENT*

249. *WARM COLORS*

250. *DIFFERENT CHAIRS*

251. *POOLS OF LIGHT*

252. *THINGS FROM YOUR LIFE*

Appendix 2 - A collection of patterns for usability of web applications proposed by Ian Graham

- Patterns for getting started on a site design: *ESTABLISHING THE BUSINESS OBJECTIVES*, *BUSINESS PROCESS MODEL*, *ESTABLISH THE USE CASES*, *TIMEBOXES*, *GRADUAL STIFFENING*, *AUTOMATE TESTING*, *USABILITY TESTING*, *GET-IT?*, R*ETEST WHEN CONTENT UPDATED*, *TWO-YEAR-OLD BROWSER*, *CLASSIFY YOUR SITE*, *SITE MAP*, *USER-CENTERED SITE STRUCTURE*, *SEARCH BOX*, *SENSE OF LOCATION*, *AESTHETICS*, *CONTEXT-SENSITIVE HELP*.

- Patterns for improving usability: *PRIMING AND INTERFERENCE*, *STRUCTURED MENUS*, *THE RHETORIC OF ARRIVAL AND DEPARTURE*, *CANONICAL LOCATION*, *SYMMETRY AND IDEMPOTENCE*, *BREADCRUMBS*, *SITE LOGO AT TOP LEFT*, *NAVIGATION BAR*, *THREE-REGION LAYOUT*, *NO FRAMES ON PUBLIC SITES*, *HOME PAGE*, *TRITE FONTS*, *THE HUMAN TOUCH*, *LINKS TO MANY SITES*, *AVATAR*, *CONTEXT-SENSITIVE CONTACT LINK*, *GO BACK TO A SAFE PLACE*, *BACK BUTTON*, *FOLLOW STANDARDS*, *PRISONER OF WAR*.

- Patterns for adding detail to a design to enhance usability even further: *KEEP IT SIMPLE*, *EXPLOIT CLOSURE*, *NO MODES*, *FEEDBACK*, *DOWNLOAD TIME*, *DESIGN PAGES FOR SCANNING*, *SHORT TEXTS*, *ANCHORS AWAY*, *NO UNPLEASANT SURPRISES*, *SEPARATE PRINT PAGES*, *SENSE OF PROGRESS*, *FINAL SLASH ON URLs*, *ACCEPTABLE WORDING*, *THE HALT AND THE LAME AND THE STRANGER AT THE DOOR*, *INTERNATIONALIZATION*, *USE OF COLOR*, *TESSELLATE GRAPHICS*, *CONTENT BEFORE GRAPHICS*, *NATURAL METAPHORS*, *WORDS BEFORE ICONS*, *WHITE SPACE SEPARATES CONTENT*, *BROKEN BUTTONS*, *MAGIC MARGINS*, *TRACK MULTIPLE IDENTICAL REQUESTS*, *UNIQUE NAMES FOR PAGES TITLES AND META-TAGS*, *CONTEXT-DEPENDENT SEARCH CATEGORIES*, *STORE CONTENT IN A DATABASE.*

- Patterns for dealing with workflow and security issues: *EQUAL OPPORTUNITY*, *AVOID PRE-EMPTION*, *CACHE TRANSACTIONS*, *RETURN VISITORS*, *SUPPORT COLOR WITH SPATIAL METAPHORS*, *WHAT YOU SEE IS WHAT YOU CAN USE*, *MANDATORY FIELDS*, *SECURITY AND ENCRYPTION*, *OBLIQUE LANDMARKS*, *PARANOID SECURITY*, *SENSE OF LOCATION IN WORKFLOW*, *CONTENT IS LINKED TO NAVIGATION*, *BUTTON GRAVITY*, *PIPELINE INTERACTION*, *DISPLAY THE OPTIONS.*

Appendix 3 - Language Generation Test Case 1 - Synchronous Collaboration

.1 The Design Issues Map, DIM

- 1: separate_layers, non_collaborative, collaborative, brainstorming;
- 2: web-based;
- 3: rewarding, collaborators;
- 4: choose, collaborators, invites, groups;
- 5: transform, game, entertainment, goals;
- 6: communication, IM, chat;
- 7: visualization, awareness;
- 7.1: instant_echos, notifications;
- 8: playback, logfiles, save_data;
- 8.1: record, snapshots;
- 9: tagging, ranking, comments, social, community;

- 10: data, search, data_filter;
- 11: teaching, education, goals;
- 12: social data;
- 13: cenzorship, making_trouble;
- 14: coordinator, coordination, mechanism;
- 14.1: timers;
- 14.2: separate_blocks;
- 14.3: community, community_application;
- 14.4: voting, agreement, rate;
- 14.5: workflows;
- 14.6: first_editor, locking, unlock after_save;
- 14.7:first_editor, decides, majority_agreement;
- 14.8: initiator, coordinator;
- 15: usability, appropriation, usage;
- 15.1: users_decision;
- 15.2: multilanguage;
- 16: composition, individual_contributions, non_collaborative, blocks;
- 17: media, transmitting_data, capturing_data;
- 17.1: visual_cues, audio_cues, visualization, awareness;
- 18: competition, teams;
- 19: medical, goals;
- 20: registered_users, priviledges;

- 21: social_networks_integration;
- 21.1: share_data, social_networks;
- 22: adapt_application, device, data, brainstorming;
- 22.1: shared_device;
- 22.2: inter_connected_devices;
- 22.3: mobile_devices;
- 22.4: tablet_pc;
- 22.5: script, automatic_identification_of_device;
- 22.6: tabletop;
- 22.7: handheld_computer, java;
- 23: save_data, history, timelines, logfiles;
- 23.1: versioning, history;
- 23.2: revert, changes;
- 23.3: track, changes;
- 23.4: interactive_history, redirect;
- 23.5: timeline;
- 24: track, contribution, awareness;
- 24.1: individual_contribution, color scheme;
- 24.2: individual_contribution, highlight_contribution;
- 25: import, export, other_formats;
- 25.1: publish, wiki;
- 25.2: import, export, pdf, jpg;

- 26: communication, similar_tools;
- 26.1: share, online;
- 26.2: similar_tools, file_transfer;
- 27: annotate, shared_resource, community, discussion;
- 28: collaborators, groups, leave_collaboration, join_collaboration;
- 28.1: notifications, status;
- 29: color_scheme, object_status, locked, unlocked;
- 30: list, available, collaborators, connected_collaborators;
- 31: templates, community, library;
- 32: conflict_resolution;
- 33: customize_collaboration, community;
- 33.1: roles, rights, collaborators;
- 34: color_scheme, individual_contribution, collaborators, identity;
- 35: search_topic, query_history;
- 36: searched_page, metadata, shared_resource;
- 37: division_of_labor, community;
- 38:recommendation_mechanism;
- 39: shared_summary, collaboration;
- 39.1: summary, search_findings;
- 40: similar_resources, shared_resource, collaborators;
- 41: session_state, store_data, resume_session, session;
- 42: project_shared_resource, awareness;

- 43: mind_map;
- 44: street_art;
- 45: city_event, citizens;
- 46: ;
- 47: recruiting_place, goals;
- 48: different_expertise, collaborators;
- 49: connect_people, goals;
- 50: layers;
- 51: perpetual_beta, prototype, versions;
- 52: feedback, corrections;
- 53: elderly_people, goals;
- 54: translation;
- 55: help;
- 56: a;
- 57: a;
- 58: a;
- 59: a;

.2 The set of Keywords, K

- adapt
- agreement_rate
- annotate
- appropriation
- audio_cues
- availability

- awareness
- brainstorm
- capturing_data
- cc
- cenzorship
- changes
- chat
- choose
- citizens
- city_event
- collaboration
- collaborative
- collaborators
- color_scheme
- comments
- communication
- community
- community_application
- competition
- conflict
- connect_people
- connected_collaborators
- coordination
- coordinator
- corrections
- customization
- data
- device
- device_identification
- different_expertise
- division_of_labor
- education
- elderly
- entertainment
- export
- features
- feedback
- file_transfer
- filter_data
- first_editor
- formats
- game
- goals
- groups
- handheld
- help
- highlight
- history
- identify_data
- identity
- IM
- import
- individual_contribution
- initiator
- instant_echoes
- integration
- inter_connected_devices
- interactive
- invites
- java
- join
- jpg
- layers
- leave
- library
- list
- locked

- locking
- logfiles
- making_trouble
- mechanism
- media
- medical
- metadata
- mind_map
- mobiles
- multilanguage
- non_collaborative
- notifications
- online
- pdf
- perpetual_beta
- playback
- priviledges
- prototype
- publish
- query_hostory
- ranking
- reasoning
- recommendation
- records
- recruiting
- redirect
- registered_users
- resolution
- resource_status
- result
- resume_session
- revert
- rewarding
- rights
- roles
- save_data
- script
- search_data
- search_results
- search_topic
- searched_page
- separate_blocks
- separate_layers
- session
- session_state
- share_data
- shared_device
- shared_resource
- similar_resources
- similar_tools
- snapshots
- social
- social_data
- social_networks
- street_art
- summary
- tablet
- tabletop
- tagging
- teaching
- teams
- templates
- timelines
- timers
- track
- transformation
- translation

- transmitting_data
- unlocked
- unlocking
- usability
- usage
- user_status
- versioning
- versions
- visual_cues
- visualization
- voting
- web_based
- wiki
- workflows

.3 The Keywords Map, KM

- *Equivalence*
  - *chat* ≡ *IM*;
  - *community* ≡ *groups*;
  - *groups* ≡ *teams*;
  - *collaborators* ≡ *connected_collaborators*;
  - *availability* ≡ *user_status*;
  - *filter_data* ≡ *search_data*;
  - *initiator* ≡ *first_editor*;
  - *cenzorship* ≡ *making_trouble*;
  - *web_based* ≡ *online*;
  - *searched_page* ≡ *search_topic*;
- *Specialization*
  - *handheld* ISA *device*;
  - *inter_connected* ISA *device*;
  - *java* ISA *device*;
  - *web_based* ISA *device*;
  - *mobiles* ISA *device*;

- *tablet* ISA *device*;
- *shared_device* ISA *device*;
- *tabletop* ISA *device*;
- *annotate* ISA *social*;
- *comments* ISA *social*;
- *corrections* ISA *social*;
- *feedback* ISA *social*;
- *ranking* ISA *social*;
- *recommendation* ISA *social*;
- *registered_users* ISA *social*;
- *social_networks* ISA *social*;
- *tagging* ISA *social*;
- *voting* ISA *social*;
- *wiki* ISA *social*;
- *coordination* ISA *mechanism*;
- *recommendation* ISA *mechanism*;
- *city_event* ISA *goals*;
- *competition* ISA *goals*;
- *education* ISA *goals*;
- *elderly* ISA *goals*;
- *entertainment* ISA *goals*;
- *game* ISA *goals*;
- *medical* ISA *goals*;
- *recruiting* ISA *goals*;
- *street_art* ISA *goals*;
- *teaching* ISA *goals*;

- *connect_people* ISA *goals*;
- *identity* ISA *social*;
- *citizens* ISA *collaborators*;
- *first_editor* ISA *coordinator*;
- *help* ISA *features*;
- *import* ISA *features*;
- *export* ISA *features*;
- *locking* ISA *features*;
- *unlocking* ISA *features*;
- *file_transfer* ISA *features*;
- *highlight* ISA *features*;
- *layers* ISA *features*;
- *playback* ISA *features*;
- *publish* ISA *features*;
- *redirect* ISA *features*;
- *resume_session* ISA *features*;
- *translation* ISA *features*;
- *versioning* ISA *features*;
- *pdf* ISA *format*;
- *jpg* ISA *format*;
- *filter_data* ISA *data*;
- *identify_data* ISA *data*;
- *save_data* ISA *data*;
- *search_data* ISA *data*;
- *share_data* ISA *data*;
- *transmit_data* ISA *data*;

- *capturing_data* ISA *data*;
- *multilanguage* ISA *usability*;
- *query_history* ISA *history*;

- *Composition*
  - *visualization* HASA *shared_resource*;
  - *session* HASA *session_state*;
  - *library* HASA *templates*;
  - *community* HASA *library*;
  - *search_results* HASA *summary*;
  - *versioning* HASA *versions*;
  - *shared_resource* HASA *resource_status*;
  - *resource_status* HASA *unlocked*;
  - *resource_status* HASA *locked*;
  - *collaborators* HASA *different_expertise*;
  - *history* HASA *interactive*;
  - *registered_users* HASA *priviledges*;
  - *device* HASA *resolution*;
  - *collaborators* HASA *rights*;
  - *collaborators* HASA *roles*;
  - *groups* HASA *collaborators*;

- *Association*
  - *connect_people* $RELATEDTO_{by}$ *adapt*;
  - *adapt* $RELATEDTO_{influences}$ *data*;
  - *device_identification* $RELATEDTO_{seq}$ *adapt*;
  - *device* $RELATEDTO_{needs}$ *adapt*;

- $community\ RELATEDTO_{needs}\ social$;
- $visualization\ RELATEDTO_{supports}\ awareness$;
- $color\_scheme\ RELATEDTO_{used-for}\ visualization$;
- $notifications\ RELATEDTO_{used-for}\ visualization$;
- $audio\_cues\ RELATEDTO_{used-for}\ visualization$;
- $visual\_cues\ RELATEDTO_{used-for}\ visualization$;
- $user\_status\ RELATEDTO_{used-for}\ visualization$;
- $instant\_echoes\ RELATEDTO_{used-for}\ visualization$;
- $identity\ RELATEDTO_{used-for}\ visualization$;
- $collaborative\ RELATEDTO_{non}\ non\_collaborative$;
- $chat\ RELATEDTO_{used-for}\ communication$;
- $annotate\ RELATEDTO_{used-for}\ communication$;
- $file\_transfer\ RELATEDTO_{used-for}\ share\_data$;
- $agreement\_rate\ RELATEDTO_{used-for}\ coordination$;
- $chat\ RELATEDTO_{used-for}\ coordination$;
- $community\ RELATEDTO_{used-for}\ coordination$;
- $division\_of\_labor\ RELATEDTO_{used-for}\ coordination$;
- $workflows\ RELATEDTO_{used-for}\ coordination$;
- $timers\ RELATEDTO_{used-for}\ coordination$;
- $first\_editor\ RELATEDTO_{used-for}\ coordination$;
- $locking\ RELATEDTO_{used-for}\ coordination$;
- $separate\_blocks\ RELATEDTO_{used-for}\ coordination$;
- $customization\ RELATEDTO_{applies-to}\ collaboration$;
- $appropriation\ RELATEDTO_{supports}\ brainstorm$;
- $track\ RELATEDTO_{applies-to}\ changes$;
- $revert\ RELATEDTO_{applies-to}\ changes$;

– $cenzorship\ RELATEDTO_{used-for}\ filter\_data$;

– $voting\ RELATEDTO_{used-for}\ filter\_data$;

– $filter\_data\ RELATEDTO_{applies-to}\ summary$;

– $identify\_data\ RELATEDTO_{supports}\ awareness$;

– $identify\_data\ RELATEDTO_{supports}\ versioning$;

– $identify\_data\ RELATEDTO_{supports}\ revert$;

– $identify\_data\ RELATEDTO_{applies-to}\ individual\_contribution$;

– $highlight\ RELATEDTO_{used-for}\ identify\_data$;

– $save\_data\ RELATEDTO_{applies-to}\ shared\_resource$;

– $save\_data\ RELATEDTO_applies-to\ comments$;

– $save\_data\ RELATEDTO_{applies-to}\ changes$;

– $save\_data\ RELATEDTO_{applies-to}\ metadata$;

– $save\_data\ RELATEDTO_{applies-to}\ results$;

– $logfiles\ RELATEDTO_{used-for}\ save\_data$;

– $library\ RELATEDTO_{used-for}\ save\_data$;

– $records\ RELATEDTO_{used-for}\ save\_data$;

– $save\_data\ RELATEDTO_{applies-to}\ session\_state$;

– $snapshots\ RELATEDTO_{used-for}\ save\_data$;

– $save\_data\ RELATEDTO_{applies-to}\ social\_data$;

– $file\_transfer\ RELATEDTO_{used-for}\ share\_data$;

– $audio\_cues\ RELATEDTO_{used-for}\ transmit\_data$;

– $visual\_cues\ RELATEDTO_{used-for}\ transmit\_data$;

– $mind\_map\ RELATEDTO_{used-for}\ share\_data$;

– $join\ RELATEDTO_{applies-to}\ groups$;

– $leave\ RELATEDTO_{applies-to}\ groups$;

– $track\ RELATEDTO_{applies-to}\ history$;

- $history\ RELATEDTO_{used-for}\ save\_data$;
- $invites\ RELATEDTO_{trigger}\ join$;
- $media\ RELATEDTO_{used-for}\ transmit\_data$;
- $media\ RELATEDTO_{used-for}\ capturing\_data$;
- $search\_topic\ RELATEDTO_{trigger}\ search\_results$;
- $timelines\ RELATEDTO_{used-for}\ save\_data$;
- $rights\ RELATEDTO_{applies-to}\ shares\_resource$;
- $import\ RELATEDTO_{applies-to}\ similar\_tools$;
- $export\ RELATEDTO_{applies-to}\ similar\_tools$;
- $script\ RELATEDTO_{used-for}\ adapt$;
- $rewarding\ RELATEDTO_{applies-to}\ collaborators$;
- $separate\_blocks\ RELATEDTO_{used-for}\ composition$;
- $prototype\ RELATEDTO_{used-for}\ usage$;
- $perpetual\_beta\ RELATEDTO_{used-for}\ usage$;
- $versions\ RELATEDTO_{used-for}\ usage$;
- $separate\_layers\ RELATEDTO_{trigger}\ adapt\_application$;

Appendix 4 - Language Generation Test Case 2 - Web Design

.4 The Design Issues Map, DIM

1. *Two-Panel Selector*: two_panel_selector, selectable_lists, easy_access;
2. *One-Window Drilldown*: one_window, option_menu, space_navigation;
3. *Wizard*: wizard, ordered_navigation, branched_tasks, chunking_the_task, physical_structure;
4. *Extras On Demand*: on_demand, hide_features, simplification, dropdown;
5. *Intriguing Branches*: additional_branches, original_flow, clear_return, resume_task;
6. *Clear Entry Points*: entry_points, clear_start, initiate_navigation, hide_flow;
7. *Global Navigation*: global_navigation, support_navigation;
8. *Color-Coded Sections*: color_codes,distinguishing_sections;
9. *Animated Transition*: animated, transition,transformation, connect_states;
10. *Visual Framework*: visual, framework, flexibility, overall_look_and_feel;
11. *Center Stage*: center_stage, cluster_secondary_tools, visual_hierarchy, clear_focus;
12. *Titled Sections*: separate_sections, strong_title, memorable_names, chuncking_the_content;

13. *Card Stack*: card_stack, separate_panels, panel_selector, memorable_names, chuncking_the_content;

14. *Closable Panels*: closable_panels, flexibility, sections;

15. *Movable Panels*: movable_panels, custom_layout;

16. *Diagonal Balance*: balance, visual_weight;

17. *Responsive Disclosure*: complex_task, single_page, unfolding_task, dynamically_growing_UI, hide_content;

18. *Responsive Enabling*: unfolding_task, disable_content, single_page;

19. *Liquid Layout*: liquid_layout, resizing, synchronisation, page_filled, wrap_content;

20. *Action Panel*: action_panel, group_actions, label_actions;

21. *Smart Menu Items*: smart_menu, synchronisation;

22. *Progress Indicator*: progress_indicator, time_consuming_task, animated_indicator, cancel_operation;

23. *Multi-Level Undo*: undo, revert_changes;

24. *Command History*: command_history, running_list_actions;

25. *Overview Plus Detail*: overview_detail, zoomed_area, inset_panel, additional_panel, viewport, magnified_projection, synchronisation;

26. *Row Striping*: row_striping, color_backgrounds, alternate_shades;

27. *Sortable Table*: sortable_table, clickable_header;

28. *Jump to Item*: jump_to_item, continous_filter;

29. *Cascading Lists*: cascading_lists, hierarchy, selectable_lists;

30. *Tree-Table*: tree_table, hierarchical_data, indented_structure, support_sorting;

31. *Forgiving Format*: forgiving_format, interpret_input;

32. *Fill-in-the-Blanks*: fill_in_the_blanks, format_input, self_explanatory_UI;
33. *Input Hints*: input_hints, suggest_input, example_input, short_hint;
34. *Input Prompt*: input_prompt, prefill_input;
35. *Dropdown Chooser*: dropdown_chooser, hide_content, complex_value_selection;
36. *Illustrated Choices*: illustrated_choices, available_choices, images;
37. *Good Defaults*: defaults, prefill_forms;
38. *Edit-in-Place*: dynamic_text_editor, edit_in_place;
39. *Smart Selection*: smart_selection, automatic_selection;
40. *Composite Selection*: composite_selection;
41. *One-Off Mode*: switch_mode, automatic_transition, one-off mode;
42. *Constrained Resize*: constrained_resize, resize_modes, resizing;
43. *Deep Background*: deep_background, soft_focus, color_gradients, depth_cues, no_strong_focal_poin
44. *Few Hues Many Values*: max_color_hues, color_pallette;

.5 The set of Keywords, K

- two_panel_selector
- transition
- additional_panel
- prefill_input
- transformation
- viewport
- dropdown_chooser
- easy_access
- connect_states
- dynamically_growing_UI
- magnified_projection
- one_window
- visual
- complex_value_selection
- option_menu
- framework
- row_striping
- illustrated_choices
- space_navigation
- flexibility
- disable_content

- color_backgrounds
- available_choices
- wizard
- overall_look_and_feel
- single_page
- alternate_shades
- images
- ordered_navigation
- center_stage
- liquid_layout
- sortable_table
- defaults
- branched_tasks
- cluster_secondary_tools
- resizing
- clickable_header
- prefill_forms
- chunking_the_task
- visual_hierarchy
- jump_to_item
- dynamic_text_editor
- physical_structure
- clear_focus
- page_filled
- continous_filter
- edit_in_place
- on_demand
- separate_sections
- wrap_content
- cascading_lists
- smart_selection
- hide_features
- strong_title
- action_panel
- hierarchy
- automatic_selection
- simplification
- memorable_names
- group_actions
- selectable_lists
- composite_selection
- dropdown
- label_actions
- tree_table
- switch_mode
- additional_branches
- card_stack
- smart_menu
- hierarchical_data
- automatic_transition
- original_flow
- separate_panels
- synchronisation
- indented_structure
- one-off mode
- clear_return
- panel_selector
- progress_indicator
- support_sorting
- constrained_resize
- resume_task
- time_consuming_task
- forgiving_format
- resize_modes
- entry_points
- chuncking_the_content
- animated_indicator

- interpret_input
- clear_start
- closable_panels
- cancel_operation
- fill_in_the_blanks
- deep_background
- initiate_navigation
- undo
- format_input
- soft_focus
- hide_flow
- sections
- revert_changes
- self_explanatory_UI
- color_gradients
- global_navigation
- movable_panels
- command_history
- input_hints
- depth_cues
- support_navigation
- cutom_layout
- running_list_actions
- suggest_input
- no_strong_focal_points
- color_codes
- balance
- overview_detail
- example_input
- max_color_hues
- distinguishing_sections
- visual_weight
- zoomed_area
- short_hint
- color_pallette
- animated
- complex_task
- inset_panel
- input_prompt
- prefill_input
- dropdown_chooser
- hide_content
- complex_value_selection
- illustrated_choices
- available_choices
- images
- defaults
- prefill_forms
- dynamic_text_editor
- edit_in_place
- smart_selection
- composite_selection
- switch_mode
- automatic_transition
- one-off mode
- constrained_resize
- resize_modes
- resizing
- deep_background
- soft_focus
- color_gradients
- depth_cues
- no_strong_focal_points
- max_color_hues
- color_pallette

.6 The Keywords Map, KM

- *Equivalence*

  - *hide_features* ≡ *hide_flow*;
  - *distinguishing_sections* ≡ *separate_sections*;
  - *strong_title* ≡ *memorable_names*;
  - *hide_content* ≡ *disable_content*;
  - *single_page* ≡ *one_window*;
  - *physical_structure* ≡ *visual_hierarchy*;
  - *time_consuming_task* ≡ *complex_task*;
  - *group_actions* ≡ *chunking_the_task*;
  - *cluster_secondary_tools* ≡ *group_actions*;
  - *magnified_projection* ≡ *zoomed_area*;
  - *fill_in_the_blanks* ≡ *suggest_input*;

- *Specialization*

  - *ordered_navigation* ISA *space_navigation*;
  - *two_panel_selector* ISA *panel_selector*;
  - *closable_panels* ISA *separate_panels*;
  - *separate_sections* ISA *sections*;
  - *movable_panels* ISA *separate_panels*;
  - *liquid_layout* ISA *custom_layout*;
  - *action_panel* ISA *separate_panels*;
  - *animated_indicator* ISA *animated*;
  - *alternate_shades* ISA *color_codes*;
  - *visual_hierarchy* ISA *hierarchy*;
  - *hierarchical_data* ISA *hierarchy*;

  - *dropdown_chooser* ISA *smart_menu*;
  - *illustrated_choices* ISA *available_choices*;
  - *defaults* ISA *short_hints*;
  - *prefill_forms* ISA *example_input*;
  - *dynamic_text_editor* ISA *dynamically_growing_UI*;
  - *deep_background* ISA *color_backgrounds*;
  - *automatic_transition* ISA *transition*;
  - *constrained_resize* ISA *resizing*;
  - *prefill_forms* ISA *prefill_input*;

- *Composition*
  - *one_window* HASA *option_menu*;
  - *physical_structure* HASA *distinguishing_sections*;
  - *ordered_navigation* HASA *transition*;
  - *physical_structure* HASA *framework*;
  - *complex_task* HASA *branched_tasks*;
  - *unfolding_task* HASA *additional_branches*;
  - *dynamically_growing_UI* HASA *visual_hierarchy*;
  - *single_page* HASA *physical_structure*;
  - *action_panel* HASA *cancel_operation*;
  - *color_codes* HASA *color_backgrounds*;
  - *resizing* HASA *resize_modes*;
  - *color_codes* HASA *color_gradients*;
  - *soft_focus* HASA *no_strong_focal_points*;
  - *color_pallette* HASA *color_codes*;

- *Association*

- $separate\_sections\ RELATEDTO_{used-for}\ separate\_panels$;
- $initiate\_navigation\ RELATEDTO_{applies-to}\ space\_navigation$;
- $global\_navigation\ RELATEDTO_{applies-to}\ space\_navigation$;
- $support\_navigation\ RELATEDTO_{applies-to}\ space\_navigation$;
- $connect\_states\ RELATEDTO_{used-for}\ space\_navigation$;
- $clear\_focus\ RELATEDTO_{used-for}\ clear\_return$;
- $hide\_content\ RELATEDTO_{applies-to}\ hide\_flow$;
- $page\_filled\ RELATEDTO_{applies-to}\ single\_page$;
- $liquid\_layout\ RELATEDTO_{used-for}\ simplification$;
- $resume\_task\ RELATEDTO_{applies-to}\ complex\_task$;
- $balance\ RELATEDTO_{used-for}\ resizing$;
- $memorable\_names\ RELATEDTO_{used-for}\ label\_actions$;
- $smart\_menu\ RELATEDTO_{used-for}\ easy\_access$;
- $progress\_indicator\ RELATEDTO_{visualize}\ transitions$;
- $undo\ RELATEDTO_{applies-to}\ complex\_task$;
- $resizing\ RELATEDTO_{triggers}\ zoomed\_area$;
- $command\_history\ RELATEDTO_{includes}\ initiate\_navigation$;
- $revert\_changes\ RELATEDTO_{triggers}\ clear\_return$;
- $resizing\ RELATEDTO_{triggers}\ magnified\_projection$;
- $row\_striping\ RELATEDTO_{triggers}\ chuncking\_the\_content$;
- $jump\_to\_item\ RELATEDTO_{supports}\ flexibility$;
- $cascading\_lists\ RELATEDTO_{supports}\ ordered\_navigation$;
- $indented\_structure\ RELATEDTO_{applies-to}\ physical\_structure$;
- $support\_sorting\ RELATEDTO_{by}\ sortable\_table$;
- $input\_hints\ RELATEDTO_{support}\ forgiving\_format$;
- $jump\_to\_item\ RELATEDTO_{support}\ complex\_value\_selection$;

- $illustrated\_choices\ RELATEDTO_{applies-to}\ visual\_hierarchy$;
- $dynamic\_text\_editor\ RELATEDTO_{support}\ flexibility$;
- $memorable\_names\ RELATEDTO_{support}\ self\_explanatory\_UI$;

Appendix 5 - Design Pattern Evaluation Questionnaire

General Information

1. Level of study:
2. Your experience in software design (in number of years):
3. Have you used/worked with design patterns before? If yes, please specify the domain of the patterns.

Understanding the Patterns

1. Which of the elements defining a pattern helped you more in understanding the patterns?
   ID Name Keywords Problem Solution Illustration
2. Which of the elements defining a pattern was the least useful for you?
   ID Name Keywords Problem Solution Illustration
3. How would you order the elements defining each pattern from the most useful to the least useful?

Pattern	Grade	Pattern	Grade
Who is the coordinator?		Support versioning	
Integrated chat		Support reverting changes	
Eyes wide open		With or without collaboration	
Choose your collaborators		Adapt application to device	
Collaboration, always social		Customize collaboration	
Annotate		Shared summary	
My contribution		Resume collaboration	
Track history of collaboration			

Table 2: Participants distribution across workshops

ID Name Keywords Problem Solution Illustration

4. How would you grade the understandability (the easiness to understand) of each pattern?
   1 – not at all understandable – 5 – very easy to understand
   See Table 2.

Using the Patterns

1. In what extent were the patterns useful for
   1 – not at all useful – 5 – the most useful

   - Understanding the design space of the application?
   - Searching for the problems?
   - Searching for solutions for identified problems?
   - Communicating with your group members?
   - Remembering similar design situations you have encountered?

- Brainstorming for design ideas for the application?
- Other:

2. What patterns did you use? Assign a letter from above (a – g) for each pattern used.
   See Table 2.

3. How many times did you search for a problem?

4. How many times did you apply a solution proposed by the patterns?

5. Did you search for solutions to problems which were not documented by the patterns? If yes, please mention these problems.

Understanding the Patterns

1. What information would you add in the definition of a pattern?

2. What representation would best fit for working with pattern?

   - Paper cards.
   - Wiki application.
   - Search engine application.
   - Personalized application with the following features.

Overall Assessment

1. How would you rate the usefulness of each pattern in the overall process?
   1 – not at all useful – 5 – the most useful
   See Table 2.

2. Did the patterns support the overall group work? If yes, how?

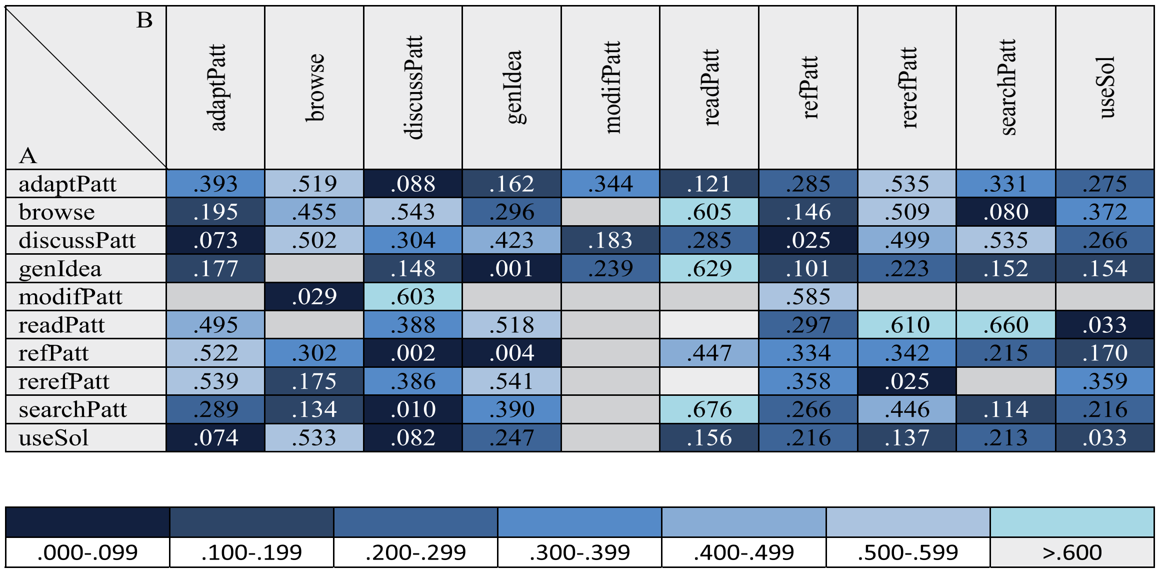
3. Any other comments or suggestions

Appendix 5 - Statistics in the evaluation workshops

A \ B	adaptPatt	browse	discussPatt	genIdea	modifPatt	readPatt	refPatt	rerefPatt	searchPatt	useSol
adaptPatt	0.40	0.11	-1.47	1.18	0.89	1.47	-0.73	-0.34	-0.71	0.73
browse	-1.08	-0.45	0.03	-0.82		-0.18	1.23	0.24	1.73	-0.55
discussPatt	-1.54	-0.20	-0.63	0.30	1.39	0.73	2.12	-0.31	-0.12	-0.75
genIdea	1.13		-1.21	3.68	1.40	-0.12	-1.43	1.09	1.30	-1.20
modifPatt		3.29	0.22				0.27			
readPatt	0.22		0.47	-0.38			-0.80	0.08	-0.04	2.21
refPatt	-0.10	-0.71	3.17	-2.51		-0.40	-0.55	-0.71	-0.97	1.07
rerefPatt	-0.33	1.33	-0.62	0.18			0.59	3.00		-0.73
searchPatt	-0.82	-1.36	2.64	-0.60		0.00	-0.83	0.41	1.51	-1.00
useSol	1.62	0.05	-1.49	-0.88		-1.24	0.90	-1.34	-1.00	2.03

🟨	🟧	🟥	🟥	🟩	🟩	🟩	🟩
<-2.00	-2.00 - -1.00	-0.99- -0.50	-0.50-0.00	0.00–0.50	0.51-1.00	1.01-2.00	>2.01

Figure 1: Frequency Matrix - Z-values for the sequences of codes considered in the evaluation workshops



A \ B	adaptPatt	browse	discussPatt	genIdea	modifPatt	readPatt	refPatt	rerefPatt	searchPatt	useSol
adaptPatt	.393	.519	.088	.162	.344	.121	.285	.535	.331	.275
browse	.195	.455	.543	.296		.605	.146	.509	.080	.372
discussPatt	.073	.502	.304	.423	.183	.285	.025	.499	.535	.266
genIdea	.177		.148	.001	.239	.629	.101	.223	.152	.154
modifPatt		.029	.603				.585			
readPatt	.495		.388	.518			.297	.610	.660	.033
refPatt	.522	.302	.002	.004		.447	.334	.342	.215	.170
rerefPatt	.539	.175	.386	.541			.358	.025		.359
searchPatt	.289	.134	.010	.390		.676	.266	.446	.114	.216
useSol	.074	.533	.082	.247		.156	.216	.137	.213	.033

■	■	■	■	■	■	■
.000-.099	.100-.199	.200-.299	.300-.399	.400-.499	.500-.599	>.600

Figure 2: Frequency Matrix - Probability values for the sequences of codes considered in the evaluation workshops