Mining for Patterns in the Design of Systems for Synchronous Collaboration

Claudia Iacob
University of Milan
Via Comelico, 39/41
20135, Milan, Italy
+39 02 50316290
iacob@dico.unimi.it

ABSTRACT
This paper reports on the results of a set of design workshops during which participants were asked to design software systems to support people in performing activities such as drawing, database querying, text editing, puzzle, and crosswords solving collaboratively and in real-time. Techniques like scenario-based design, sketching, and mock-ups were used during the workshops for identifying recurring design problems and possible solutions to tackle them. Through the workshops and by analyzing a set of existing synchronous collaborative tools, a collection of design patterns for the design of systems to support synchronous collaboration was identified and it is documented by the paper.

Categories and Subject Descriptors
D.2.10. [**Software Engineering**]: Design – *Methodologies*.

General Terms
Design, Human Factors.

Keywords
Design patterns, synchronous collaborative systems, design workshops.

1. INTRODUCTION
The concept of design patterns was first introduced in the '70s by Alexander [1], who proposed a pattern language for architectural design. Later on, the concept was adopted in domains like software engineering and HCI [2]. In [3], a survey of 21 HCI pattern languages published between 1996 and 2007 shows that these collections target web user interface design, user interface design for desktop applications, interactive exhibits, user interface related programming, hypermedia applications, or ubiquitous computing. Moreover, several collections of patterns have been proposed for the design of social interfaces [4], groupware technology [5], and cross-culture collaboration [6].

Even if synchronous collaboration is common in various contexts (such as searching [7], or sketching [8]), little work has been done in identifying design patterns for the design of systems for such collaboration. This work aims to identify a set of design patterns in the design of software systems which support people in performing collaboratively and in real-time activities such as drawing, text editing, searching, and game solving. The paper reports on the results of a set of design workshops during which participants were asked to design software systems to support people in performing activities such as drawing, database querying, text editing, puzzle, and crosswords solving collaboratively and in real-time. Through the workshops and by analyzing a set of existing synchronous collaborative tools, a collection of design patterns for the design of systems to support synchronous collaboration was identified and it is documented by the paper.

AsianPLoP'11, March 17–19, 2011, Tokyo, Japan.

2. DESIGNING SOFTWARE FOR SYNCHRONOUS COLLABORATION
There are several issues of concern in the design of software systems which support synchronous collaboration. This work addresses a subset of them and looks at collaboration on two dimensions: the activities that might be performed in a collaborative process [9] and the contexts in which these activities can be performed (Figure 1).

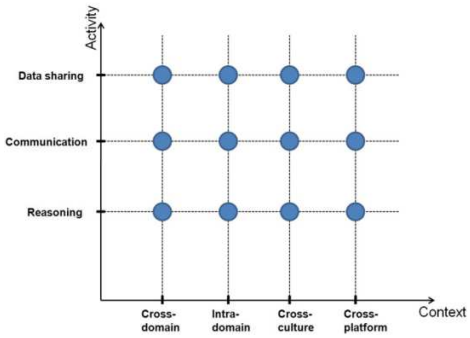

Figure 1 – Designing Collaborative Software Systems

The activities are categorized as reasoning activities, communication and coordination activities, and data sharing. This paper considers a small sample of reasoning activities, including drawing, searching, text editing and game solving. The contexts in which these activities can occur are among participants with different backgrounds (cross-domain), participants with similar backgrounds and expertise (intra-domain), participants belonging to different cultures (cross-culture), and participants using different devices (cross-platform).

A set of requirements for such collaborative systems includes: 1). The system needs to complement and enhance real spaces of collaboration, 2). Users should be supported in reasoning on the

problem at hand through appropriate interaction, 3). The system should support and enhance the interaction and common understanding among collaborating users, 4). Users should be able to exchange information, 5). Each user may perform domain-specific knowledge activities in real time, 6). Each user has available all and only the tools s/he needs for the activities he is interested in, 7). Each user should be able to interact with the system through his/her own system of signs, 8). The system should be materializable on diverse platforms, 9). A common shared knowledge base should be available to each user.

2.1 Concrete Examples

The subsection describes a set of concrete examples of software applications which support synchronous collaboration in the domains targeted by the paper.

2.1.1 Collaborative Drawing

Tools to support synchronous collaborative drawing target mainly architectural design, graphic design, or education. Synergo [8] is a synchronous collaborative tool used for building diagrammatic representations. It is targeted to small groups of students, but it also provides analyzing and supervision tools for the teachers. 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".

2.1.2 Collaborative Querying

Based on interviews with teachers, librarians, and researchers, it is concluded in [7] that there are many situations in which "groups of people gather around a single computer to jointly search for information online". CoSearch [7] is a collaborative searching tool which supports: a). keeping track of shared sessions' findings, b). identifying individual contributions to a search query by color cursors, c). visualizing what the other collaborators are searching for, d). adding notes about the websites collaboratively searched for, and e). saving the summaries of the searches and the history of the collaboration. During a search session, one collaborator is designated the "driver", being responsible for the execution of the query. Moreover, the tool targets desktops as well as mobile devices.

Similar goals are designed for Coagmento [10], a tool able to support collaborative information seeking. Coagmento provides its users with a chat for communication. It also displays the collaborators' information, making all collaborators aware of the actions the others are performing. All queries of a session are logged, and documents obtained through the query may be saved or flagged for further discussions. The tool can be used in a collaborative manner as well as individually.

2.1.3 Collaborative Text Editing

Twelve challenges in the design of synchronous collaborative editing software are identified in [11]: time and space, awareness, communication, private and shared work spaces, intellectual property, simultaneity and locking, protection, workflow, security, file format, platform independence, and user benefit. TellTable [11] tries to answer some of these challenges, 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". The full history of the development of a document is being maintained by a versioning system, 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.

Synchronous collaborative text editing is supported also by GoogleDocs [12]. 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.

2.1.4 Collaborative Games

"Mystery at the Museum" (M@M) [13] is a synchronous collaborative game meant to engage visitors in museum exhibits and to encourage them to collaborate in solving a detective problem. 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.

Collaborative Puzzle Game (CPG) [14] is a tabletop interactive game addressing children with Autism Spectrum Disorder (ASD). 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. Feedback is provided to the players through both visual and auditory media. Any player can, at any time, visualize what his collaborator is doing. The coordination of the players is enforced in that a piece of the puzzle can only be moved when both players drag it.

3. PATTERN MINING THROUGH DESIGN WORKSHOPS

In order to identify recurring design problems and their solutions in the design of software systems for synchronous collaboration, teams of professional designers, graduate and undergraduate students in Computer Science were asked to design software applications for synchronous collaboration. The problems the applications would have to be a solution for were collaborative drawing, collaborative database querying, collaborative text editing, and collaborative games (puzzle and crosswords solving). The participants were encouraged to externalize their design ideas through techniques such as scenario-based design [15, 16], sketches, and mockups. Their design processes were observed.

3.1.1 Participants

The total number of participants was 50, out of which 80% were male, and 20% female. 19 participants (38%) were Master students in a "Multimedia databases" class (DB), so they were divided in five teams and worked on the collaborative database querying 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. Lastly, 4 of the participants (8%) were professional designers. They worked as a team in designing an application for collaborative drawing.

Problem	Participants		Male	Female	Designers	Master students HCI course	Master students DB course	Undergrad students TC course
	Nr.	Teams						
Collaborative Drawing	11	3	8	3	4	4	0	3
Database querying	19	5	14	5	0	0	19	0
Text editing (summarization)	4	1	4	0	0	4	0	0
Puzzle solving	7	2	7	0	0	4	0	3
Crosswords solving	9	2	7	2	0	5	0	4

Table 1 – Workshop participants' statistics

Out of the 50 participants, 22% worked on the design of systems for collaborative drawing, 8% on collaborative text editing, 38% on collaborative querying, 14% on collaborative puzzle solving, and 18% on collaborative crosswords solving (Table 1).

3.1.2 Procedure

Participants worked in teams of 3-5 people and the duration of a workshop was 2 hours. Each team was presented with the list of problems and was encouraged to choose one problem for which to design a software application. Participants were initially presented with the techniques they would be using: scenario-based design [15, 16], sketching, and mock-ups.

During the first phase, participants were asked to define as many scenarios as they could consider for the application under design. In defining a scenario, they would consider answering the questions: a). who are the users?, b). what are they allowed to do with the application?, c). how could they achieve their goals using the application?, d). what is the motivation of their collaboration?, and e). when and where could the application be used?.

The second phase asked participants to choose another problem from the list and to find as many 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 was to identify commonalities and major differences between collaborative systems 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, 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. For that, they were strongly encouraged to sketch their ideas, express all the design problems they encounter and, possibly, create a mock up of their overall design. They were observed throughout the process.

3.1.3 Problems and Designs

Collaborative Drawing

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. Participants envisioned possible scenarios of use for the application, such as: a). seeing the drawing grow in a public space during a city event, b). networks of friends get together online and draw as in a game, or c). educational tool allowing masters to draw together with their apprentices.

One of the main issues discussed throughout the design of this type of application was associating it with some concrete, specific goals which would allow users to find their own motivation in using the application. Another long debated problem was related to the coordination mechanism the application would need for users to be able to synchronously collaborate. One of the solutions proposed was to "*link the application to a community and to the way this community is coordinated*".

Collaborative Database Querying

The application for the collaborative querying had a concrete context of application: movie database querying. 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 possible criteria for the query consisted of: a). the reviews of the movie (e.g. all movies with good reviews this year), b). the soundtrack of the movie (e.g. all movies which contain as soundtrack songs of a specific band), c). the cast of the movie (e.g. all movies in which the actor in the photo plays a main role), and d). the movies broadcasted next to a point of interest. The query would be run only after all the users who edit it would agree on it and each participant is free to edit his/her contribution to the query at any time (before the query is run).

Collaborative Text Editing

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 possible scenarios considered by the participants included: a). students taking notes during a lecture on the tablet PC and using one common document, b). creating a mash-up between document editing and chatting, c). allowing the collaborative creation of mental maps.

One of the design issues considered for this particular problem was maintaining the consistency of the document while synchronously editing it. Also, the participants focused on including in their design several social tools such as tagging and ranking systems, comments, and mail notifications of changes.

Collaborative Games

The set of games considered for collaborative solving consisted of puzzles and crosswords. Separate teams worked on these problems, each team focusing on one of the problems. The common requirement for both was that more users solve one game in the same time.

As expected, several similar issues were reported by the teams. All the teams considered important having the possibility of tracking the history of the game, possibly identifying each individual's contribution in order to support competition between different teams. Moreover, for both problems, participants considered adapting the game to the user's profile and to the device s/he is using. However, while the puzzle solving application was also considered to be used in a medical context, the crosswords solving application was thought to be used during a city event.

3.1.4 Results

Throughout their design processes, participants were encouraged to externalize any design problem, solution, or decision they considered relevant to their problem. In addition to that, the observer took notes of their conversations. This lead 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. For each design issue, its degree of recurrence (DoR) was computed as the percentage of recurrence of the issue throughout the workshops. As example, a design issue that was discussed by all the teams would have a DoR = 100, while an issue discussed by half of the teams would have a DoR = 50.

A list of the design issues mostly recurring during the design workshops is presented in Table 2. The only issue discussed by all teams is related to coordination and to deciding who should be the coordinator in a synchronous collaboration. A set of solutions were identified, described later on in this paper. Several important issues such as supporting collaborators' communication and adapting the application to several devices have been discussed by most of the teams.

No.	Design issue	DoR
1	Who is the coordinator in a real-time collaboration? How do collaborators coordinate themselves?	100
2	Support collaborators' communication; integrate instant messaging features in the application	76.92
3	Adapt application to several devices; make sure collaborators using different devices can work together in real time	69.23
4	Design the application for the web; web-based collaboration	69.23
5	Provide each user with the possibility of performing the activity supported by the application in a private area, individually	53.84
6	Support collaborators in providing feedback, comments, annotations, rankings	46.15
7	Visualize what the others are doing in real time	46.15
8	Allow each user to choose his/her collaborators	46.15
9	Support competition among various teams collaborating	38.46

Table 2 – List of the most recurring design issues in the conducted workshops

A further step conducted was to identify which of the design issues discussed during the workshops are mostly considered in the implementation of existing applications for synchronous collaboration (Table 3). The set of applications analyzed are the concrete cases described in section 2.1: Synergo, CoSearch, GoogleDocs, Coagmento, Mystery at the Museum, Collaborative Puzzle Game, TellTable.

No.	Issue Workshop	Design issue	Synergo	CoSearch	GoogleDocs	Coagmento	M@M	CPG	TellTable	DoR
1	1	Who is the coordinator in a real-time collaboration? How do collaborators coordinate themselves?	■	■	■	■		■	■	85.71
2	-	Support the visualization of the history of the collaboration through log files and/or timelines	■	■	■	■			■	71.42
3	-	Support the tracking of each collaborator's individual contribution		■	■	■	■		■	71.42
4	2	Support collaborators' communication; integrate instant messaging features in the application	■		■	■	■			57.14
5	7	Visualize what others are doing in real time		■	■	■		■		57.14
6	3	Adapt application to several devices; make sure collaborators using different devices can work together in real time		■			■	■		42.85
7	6	Support collaborators in providing feedback, comments, annotations, rankings	■	■		■				42.85
8	4	Design the application for the web; web-based collaboration		■	■				■	42.85

Table 3 – The design issues identified through the workshops and their implementation in concrete cases of applications

For each of the design issues, the DoR was computed with respect to the software analysis as the percentage in which the specific issue was considered in the implementation of the applications. As example, an issue considered in the implementation of all of the concrete cases of applications would have a DoR=100, while an issue considered in half of the cases would have a DoR=50.

The analysis of the concrete cases of applications revealed several design issues that were slightly considered throughout the workshops, but nonetheless relevant to the synchronous collaboration aspect of the application. Five out of the seven concrete existing applications considered would allow each user involved in the collaboration to track his/her contribution at all times. 71% of the cases considered allowed the visualization of the history of the collaboration by any collaborating user. Also, 57% of the concrete cases of applications allowed each user to visualize in real time the contribution of his/her collaborators.

4. The Patterns Identified

The design issues with the highest DoR were considered for being documented through design patterns. This section describes 8 of the design patterns identified.

4.1 Who is the coordinator?

Context

A group of collaborators work together on the same shared resource. Each of them contributes to the creation and the evolution of the resource, making sure that together they reach to a final version of the resource, version agreed by all those involved in the process.

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.

Forces

Collaborative drawing and collaborating text editing support groups of users in working on a single resource (the canvas or the text area) in the same time. It may be that the group is a community with unwritten rules which would not ask for an enforced coordination mechanism for the system. However, this might not always be the case.

Collaborative database querying asks for a (small) group of users to create a query on a data base using visual virtual tools. 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.

Collaborative puzzle or crosswords solving are particular cases in that they are games, hence competitions could be supported by applications targeting to solve these problems. This fact adds a new dimension to the coordination problem: time.

Examples

Collaborative drawing asks for more users to collaborate in the creation of one diagrammatic representation. The resource in this case is the canvas which must be available to all the collaborators and, in the same time, remain in a consistent state during the entire process.

In collaborative text editing the issue of resource consistency is highly important, hence requiring the enforcement of a coordination mechanism able to support efficient collaboration. TellTable is an example of collaborative editing tool which supports coordination by locking the editing area when a collaborator starts editing it and unlocking it after the contribution is saved.

Collaborative games such as puzzles or crosswords may require for an enforced coordination mechanism in order to allow the participation of all collaborators in a sequential manner.

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 querying, 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.

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

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.

Forces

Real-time communication allows collaborators with different expertise to share and clarify any misunderstandings that might come up throughout their process.

Moreover, 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.

Examples

Users trying to collaboratively solve a puzzle can only use the pieces of the puzzle to communicate. They would be highly supported by an instant messaging feature through which they can exchange ideas on the solution of the puzzle. One concrete example of application which addresses this issue is GoogleDocs where an instant messaging feature is available.

Solution

Therefore: Integrate an instant messaging feature in the design of the application. In doing that, either link the application to an existing IM 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.

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 in order to allow participants' coordination.

Forces

Collaborative activities such as drawing or text editing would be highly interrupted by notifications of all updates on the shared board. 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, mining for updates on the board should be fast, and straightforward.

Examples

Crosswords players sharing a large board would benefit from being notified of the addition of a new word on the board.

Users drawing together might not need an explicit notification every time some collaborator draws something on the board.

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.

4.4 Choose your collaborators

Context

Co-workers get connected to the synchronous collaborative application and they would be 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.

Forces

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.

Examples

A network of friends using the application plans to solve a puzzle together. They want to be able to form a group and work together for a collaborative solution to the game.

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.

4.5 Collaboration, always social

Context

Groups of collaborators are working together through web applications. 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. 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.

Forces

Collaboration triggers the forming 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.

Examples

Collaboration in text editing can be supported by allowing all collaborators to annotate parts of or the entire shared document they are editing.

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.

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. One member of the group might want to know his/her contribution to the editing process.

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 collaboration.

Forces

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.

Examples

In order to support competition in game applications, it is necessary to identify the individual contribution of each collaborator, and so to facilitate the creation of charts.

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 colors (hence, defining a color scheme for the application) to each collaborator is a solution. In this way, each user's contribution is highlighted in a different color.

❖ 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 tooltips containing information on the author of that particular part.

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.

Forces

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.

Examples

In the case of text editing, users might want to go back to previous versions of the document.

Moreover, tracking a game's history allows replaying, supporting strategy 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.

4.8 With or without collaboration

Context

Collaborating users share a public area of the application. Some might find it useful to be allowed to sketch their ideas within a private area of the application before making them public to their collaborators.

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.

Forces

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, hence those collaborators who cannot edit the resource at a given time due to it being locked might feel their creativity is being hindered.

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.

Examples

One might want to try out solutions for matching pieces of a puzzle. For that, s/he needs a private board where s/he can put together pieces in order to check if they do match the final solution.

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.

5. TOWARDS A PATTERN LANGUAGE

The collection of patterns identified can be placed in the context of definition for collaboration, as depicted in Section 2. Hence, reasoning activities are supported by awareness (addresses in *Eyes wide open*), and brainstorming (discussed in *With or without collaboration*). Communication and coordination activities are discussed in *Integrated chat* and *Who is the coordinator?;* moreover, they are addressing communities (*Choose your collaborators* and *Collaboration, always social*). Lastly, data sharing includes data identification and data storage (addressed in *My contribution* and *Track history of the collaboration*).

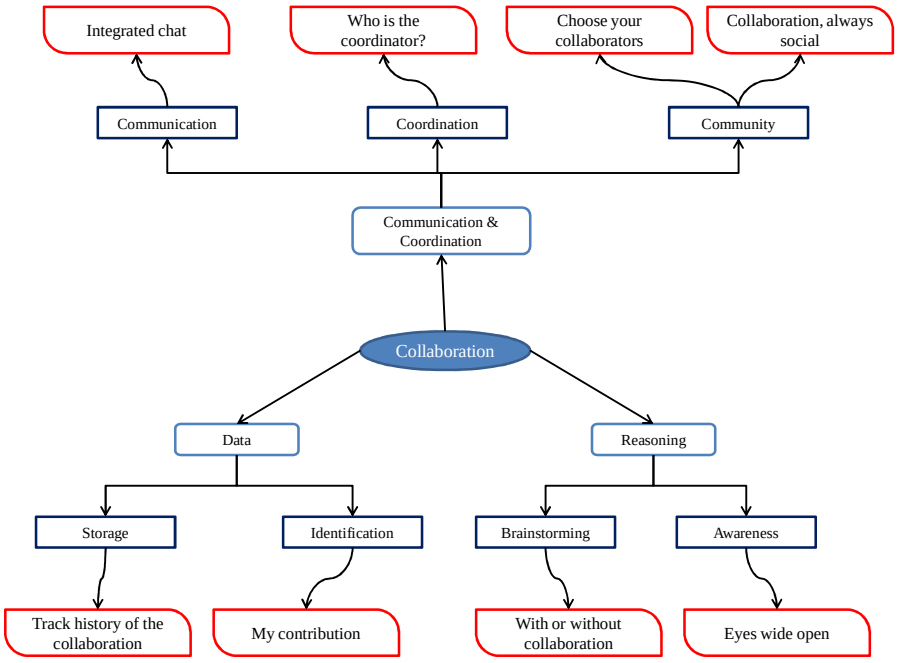

Figure 1 – The patterns identified in the context of collaboration

Moreover, the collection of patterns can be enhanced by a set of relationships existing between the patterns. Figure 2 depicts the collection together with the identified relationships. The pattern language, however, is not complete. As future work, the author is planning on extending the language.

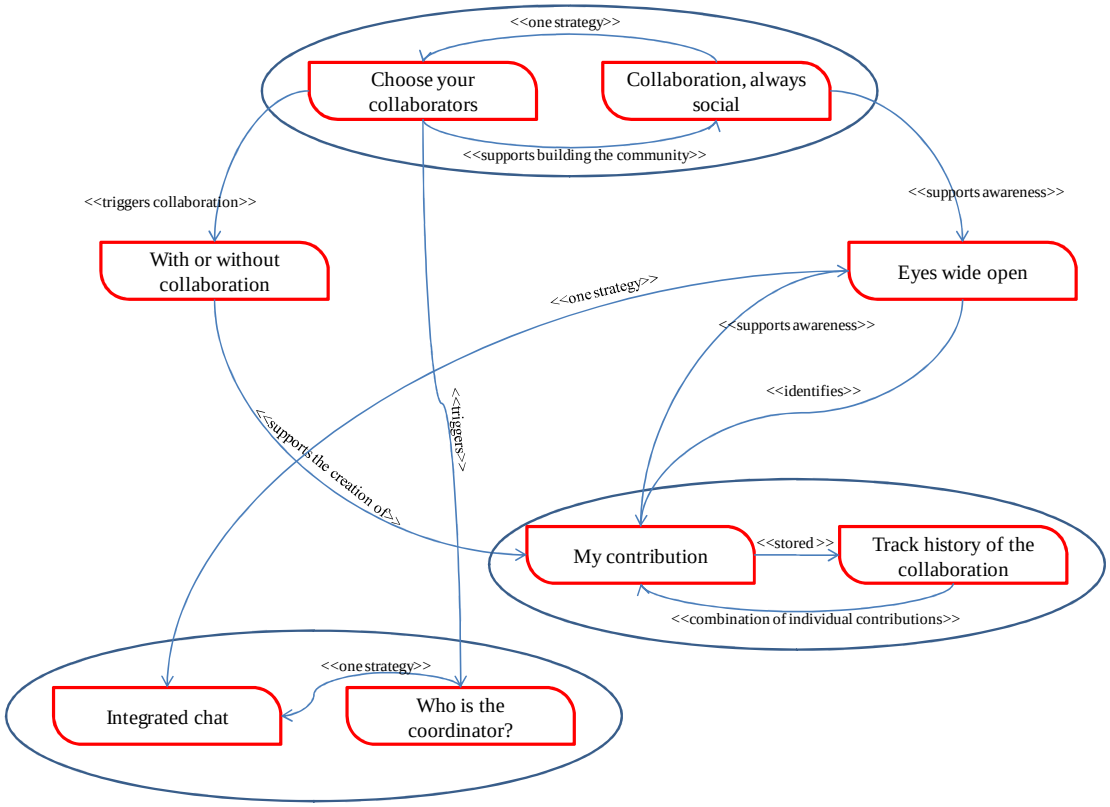

Figure 2 – Towards a pattern language for the design of synchronous collaborative systems

6. CONCLUSIONS

The contribution of the paper consists in identifying 8 design patterns for the design of software systems for synchronous collaboration. The domains targeted for collaboration are drawing, text editing, database querying, and game solving. The pattern mining process consisted of two phases.

During the first phase, a set of design workshops were held. Participants were asked to design applications for synchronous collaboration in the domains targeted using techniques like scenario-based design, sketching, and mock-ups. Throughout these workshops, the design issues the participants identified were collected. For each design issue, its DoR was computed as the percentage of recurrence of the issue throughout the workshops.

The second phase of the mining process was the analysis of a set of existing tools for synchronous collaboration in drawing, text editing, querying (information searching), and games. A list of all the design issues considered in the implementation of these applications was identified and for each issue its DoR was computed.

The design issues with a higher DoR in both the results obtained from the workshops and the results obtained through the applications analysis were considered to be documented through design patterns. As result, 8 design patterns were identified and are documented by the paper: *Who is the coordinator?*; *Integrated chat*; *Eyes wide open*; *Choose your collaborators*; *Collaboration, always social*; *My contribution*; *Track history of collaboration*; *With or without collaboration.*

7. ACKNOWLEDGMENTS

The author would like to thank Hanyuda Eiiti for the feedback on the paper and Alessandra Agostini, Giuseppe Boccignone, and Stefano Valtolina for their support. Special thanks to all the participants in the workshops for the inspiring ideas and great designs. This work was supported by the *Initial Training Network* "Marie Curie Actions", funded by the FP 7 – People Programme entitled "DESIRE: Creative Design for Innovation in Science and Technology".

8. REFERENCES

[1] Alexander, C. 1977. A pattern language: Towns, buildings, construction. New York: Oxford University Press.

[2] Borchers, J. 2001. A Pattern Approach to Interaction Design. John Wiley & Sons, Inc.

[3] Kruschitz, C., Hitz, M. 2010. Analyzing the HCI Design Pattern Variety, Proceedings of AsianPLoP2010, GRACE-TR-2010-01

[4] Crumlish, C., Malone, E. 2009. Designing Social Interfaces. O'Reilly Media, Inc.

[5] Lukosch, S., Schümmer, T. 2004. Communicating Design Knowledge with Groupware Technology Patterns: The Case of Shared Object Management. CRIWG 2004, 223-237.

[6] Schadewitz, N. 2009. Design Patterns for Cross-cultural Collaboration. International Journal of Design, 3(3).

[7] Saleema Amershi and Meredith Ringel Morris. 2008. CoSearch: a system for co-located collaborative web search. *Proceeding of CHI '08*. ACM, New York, NY, USA, 1647-1656.

[8] Margaritis, M., Avouris, N., Kahrimanis, G. 2006. On Supporting Users' Reflection during Small Groups Synchronous Collaboration. 12th International Workshop on Groupware, CRIWG 2006. LNCS 4154. Springer.

[9] Ellis, C. A. et al. 1991. Groupware: some issues and experiences. *Commun. ACM* 34(1), 39-58.

[10] Chirag Shah, Gary Marchionini, and Diane Kelly. 2009. Learning design principles for a collaborative information seeking system. *Proceedings* of *CHI '09*. ACM, New York, NY, USA, 3419-3424.

[11] Andy Adler, John C. Nash, and Sylvie Noel. 2006. Evaluating and implementing a collaborative office document system. Interact. Comput. 18(4), 665-682.

[12] GoogleDocs http://www. docs.google.com/

[13] Eric Klopfer, Judy Perry, Kurt Squire, Ming-Fong Jan, and Constance Steinkuehler. 2005. Mystery at the museum: a collaborative game for museum education. *In Proceedings of the 2005 conference on Computer support for collaborative learning: learning 2005: the next 10 years! (CSCL '05).* International Society of the Learning Sciences 316-320.

[14] A. Battocchi, F. Pianesi, D. Tomasini, M. Zancanaro, G. Esposito, P. Venuti, A. Ben Sasson, E. Gal, and P. L. Weiss. 2009. Collaborative Puzzle Game: a tabletop interactive game for fostering collaboration in children with Autism Spectrum Disorders (ASD). Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS '09). ACM, New York, NY, USA, 197-204.

[15] John M. Carroll. 1999. Five Reasons for Scenario-Based Design. *Proceedings of the Hawaii International Conference on System Sciences* (HICSS '99), Vol. 3. IEEE Computer Society, Washington, DC, USA, 3051-.

[16] John M. Carroll (Ed.). 1995. *Scenario-Based Design: Envisioning Work and Technology in System Development*. John Wiley & Sons, Inc., New York, NY, USA.