Electronic Communications of the EASST
Volume 31 (2010)

Proceedings of the
Second International Workshop on
Visual Formalisms for Patterns
(VFfP 2010)

Towards a Pattern Language for the Design of Collaborative Interactive Systems

Claudia Iacob, Piero Mussio, Li Zhu and Barbara Rita Barricelli

12 Pages

Guest Editors: Paolo Bottoni, Esther Guerra, Juan de Lara
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Towards a Pattern Language for the Design of Collaborative Interactive Systems

Claudia Iacob, Piero Mussio, Li Zhu and Barbara Rita Barricelli

Department of Computer Science, University of Milan

Abstract: Nowadays, the design of interactive systems addresses diverse communities of end users, each belonging to a certain culture, having a role in the context/domain and using a specific digital platform. More than often, they come together and collaborate in performing their work tasks and need to be supported by virtual interactive systems. This brings a set of challenges and design problems to be faced by interaction designers focused on the design of collaborative interactive systems. The present paper focuses on one approach to overcome these challenges – by making available the knowledge and wisdom within a team of designers to each and every designer in the team by the definition of pattern languages, organized as sets of multimedia, multimodal documents accessible and manageable in the Web. A design pattern language comprises a set of inter-related design patterns able to address interaction design problems and to allow the accumulation and use of knowledge within a team of designers. This paper identifies and describes a set of design patterns addressing the design of collaborative interactive systems together with the possible relationships among them and the operations made available to designers for managing and using the patterns.

Keywords: Collaboration, Design patterns, Interaction design, Localization

1 Introduction

Collaborative interactive systems focus on supporting a community of end users - i.e. people who are not computer science experts, but are supported by software systems in performing their everyday work activities [4] – to work together in achieving a common goal. The design of collaborative interactive systems, as all design processes, is a complex creative process which requires more knowledge than any single person can posses [11]. "The predominant activity in designing complex systems is that participants teach and instruct each other." [10] Therefore, the design of collaborative interactive systems requires the participation of multidisciplinary stakeholders constituting a design team and bringing the necessary experiences, skills and knowledge to the design process. The stakeholders becoming designers in the team need to communicate and collaborate with each other, sharing their common problems and understanding of the design issues they face. This leads to the need of tools for sharing, managing and enhancing a common knowledge base, accessible to all the participants in the design process and which gathers the common knowledge and wisdom within a team of designers. Making all voices heard and allowing all design problems to be shared enables social creativity which, as defined by Fischer, "explores computer technologies to help people work together" [11].
Design patterns are tools to support social creativity providing a way of capturing and sharing knowledge and wisdom related to design problems arising in the design of interactive systems.

Each design pattern is a multimedia document storing “a proven solution to a recurring design problem” [3]. Moreover, in design, problems are not isolated: they refer to each other, smaller problems arising in the context of larger ones. Therefore, patterns are not isolated, but linked according to the relationships among the problems whose solution they document. These links relate patterns together to form a pattern language [3], a shared repository of a finite number of problem-solution specifications. A design pattern language can be used for the design of collaborative interactive systems, accessed and managed by all the members of the team of designers.

The paper adopts and refines the definition of design pattern proposed in [3] and introduces a set of inter-related patterns addressing the design of collaborative interactive systems. These patterns form a pattern language seen as a continually evolving shared repository of knowledge and wisdom which may be modified and enriched at any time based on the designers’ experience.

2 Background and Related Work

Patterns and pattern languages were introduced by the architect Alexander in the seventies [1] as tools for capturing and making available and communicable knowledge and wisdom related to urban spaces. Alexander conceived urban spaces as artefacts, where “people enjoy living in” and which “have a certain, timeless ‘Quality without a Name’ that cannot be reduced to a single dimension” [3]. These environments must provide affordances which support “the patterns of events that frequently happen there” [1]. Patterns of events that frequently happen in a space and the relationships among them are not created by the architects themselves, but emerge by 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 and wisdom 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 multimedia documents to be used by architects: 1) as knowledge and wisdom 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 [13], 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 hypertext which organizes good design practices within 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. Paper, as communication support, constraints the way of navigation in the hypertext as well as its update and maintenance, i.e. how the evolution of design pattern languages occurs.

Alexander’s approach had a wide impact in several domains, including Computer Science and Human-Computer Interaction (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 [3].

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 [15]. 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” [3]. Several collections of design patterns [18, 20] for interface and interaction design are now available on the Web. A collection of patterns targeted for the design of social interfaces is introduced in [7]. The focus of these patterns is on the design of systems which support social activities like: broadcasting and publishing, collecting data, rating, or collaborative editing.

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 design of complex processes requires more knowledge than any single person can posses [11]. 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, Borchers defines the Pattern Language Mark-up Language (PLML) [14] 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. In this way, three hypertexts become available to the stakeholders in the design teams, making available concepts from the application domain, the HCI and SE domains.

Our focus is on the HCI problems in the design of collaborative interactive systems. We adopt Borchers view, refining and adapting methods and tools to the new domain based on our experiences [5, 6, 19]. We look at collaborative interactive systems as spaces where end users can develop their collaborative activities and at pattern languages as the repository of knowledge and wisdom gained by designers so far. In this paper we propose a pattern language addressing HCI problems and solutions within this context.

3 Pattern Mining in the Design of Collaborative Interactive Systems

Collaborative interactive systems have as goal supporting the collaborative work of end users working within communities in terms of: a). their reasoning on the problem at hand, and the knowledge creation and management to support their reasoning, b). their communication and common understanding through appropriate interaction and c). data sharing among them [9]. There are several issues to be faced in the design of collaborative interactive systems coming from both the diversity of these end users and the diversity of technology they use.

In order to support the mining of design patterns addressing issues in the design of collaborative interactive system design, we identify – based on literature review [9, 11, 12] and on previous design experiences [5, 6, 19] – two dimensions that affect the design process: *activity* and *context*. As activities, we consider reasoning, communication and data sharing. The contexts identified are cross-domain, intra-domain, cross-culture and cross-platform. Table 1 summarizes possible issues to be addressed within each class of situations.

	Reasoning	**Communication**	**Data sharing**
Cross-domain	Support the reasoning, knowledge creation and management of end users working in different domains	Enable end users working in different domains to communicate and reach a common understanding through appropriate interaction	Allow the sharing of data among end users working in different domains
Intra-domain	Support the reasoning, knowledge creation and management of end users working in the same domain, but having different roles	Enable end users working in the same domain, but having different roles to communicate and reach a common understanding through appropriate interaction	Allow the sharing of data among end users working in the same domain, but having different roles
Cross-culture	Support the reasoning, knowledge creation and management of end users belonging to different cultures	Enable end users belonging to different cultures to communicate and reach a common understanding through appropriate interaction	Allow the sharing of data among end users belonging to different cultures
Cross-platform	Support the reasoning, knowledge creation and management of end users who use different platforms	Enable end users who use different platforms to communicate and reach a common understanding through appropriate interaction	Allow the sharing of data among end users using different platforms

Table 1 – HCI issues in the design of collaborative interactive systems

To this, two software engineering related concerns can be added. On one hand, guaranteeing a consistency among the different instances of the same system; on the other hand, facilitating the maintenance and reuse of these systems.

3.1 Design Pattern and Pattern Language Definitions

Several different templates for defining design patterns have been proposed. These templates 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 [8].

Borchers [3] proposed a first definition of the template. This definition is refined in:

P = (id, n, pb, F, E, d, K, s, R, IN, OUT)

The description of these elements is defined below:

- The *identifier*, id is a string of characters that uniquely identifies a pattern and which respects the following regular expression format: Pn, where $n \in \mathbb{N}$.
- The *name*, n of the pattern is a string of characters which helps refer to the central idea of the pattern.
- The *problem*, pb is multimedia, multimodal description of the major issue the pattern is trying to solve. It may embed textual, graphical, audio and video content.
- The set of *forces*, F = $\{f_1, \ldots, f_i\}$ is a set of multimedia, multimodal descriptions which present the implications of the problem addressed by the pattern. The forces are defined as the cognitive, social or economical related issues which influence the problem described by the pattern [3].
- The set of *examples*, E = $\{e_i, \ldots, e_j\}$ presents the multimedia description of a set of existing situations in which the problem described by the pattern arises.
- The *diagram*, d is a graphical illustration of the pattern.
- The set of *keywords*, K = $\{k_1, \ldots, k_t\}$ 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 patterns. 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.
- The *solution*, s is the multimedia description of a possible method or process for solving the problem addressed by the pattern.
- The *references*, R = $\{r_1, \ldots, r_u\}$ set is a set of literature references related to the pattern.

We refine Borchers definition by making explicit three defining elements: K, IN, OUT. Moreover, we embed the definition of the "illustration" element from [3] in the definition of the elements p and IN.

Patterns are inter-related, allowing the definition of problems ranging on a scale of complexity – more general patterns may point to more specialized patterns. Possible relationships between patterns are: IS-A, HAS-A, RELATED-TO. Hence, two additional defining elements are identified for a design pattern:

- The *input*, IN = $\{in_1, \ldots, in_a\}$ is the set of ids of the design patterns which define a context, i.e. the design situations in which the pattern can be used.
- The *output*, OUT = $\{out_1, \ldots, out_b\}$ is the set of ids of the design patterns which define the design situations that refine the one in which the pattern is used.

A pattern language comprises a set of inter-related design patterns and it is a directed acyclic graph PL = $\{\Omega, \Delta\}$, where $\Omega = \{P_1, \ldots, P_n\}$ is a finite set of nodes representing design patterns and $\Delta = \{R_1, \ldots, R_m\}$ is the finite set of edges representing the relationships among the patterns. $P_1 \in \Omega$ is said to point to $P_2 \in \Omega$, hence a relationship between P_1 and P_2 can be identified, if and only if there is a directed edge $R_k \in \Delta$, leading from P_1 to P_2. In this case, $P_1 \in$ IN of P_2 (i.e. P_1 belongs to the input set of P_2) and $P_2 \in$ OUT of P_1 (i.e. P_2 belongs to the output set of P_1).

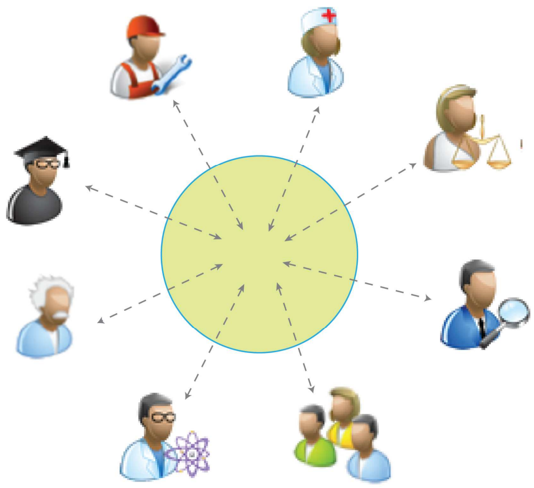
Each design pattern may be described with the help of a specialized XML-based language – PLML [14]. PLML supports the definition of inter-related design patterns, hence providing means of representation of pattern languages as directed acyclic graphs, which can be viewed, browsed and managed accordingly.

3.2 Patterns for the Design of Collaborative Interactive Systems

In what follows, the paper provides the brief description of a subset of the design patterns derived from previous collaborative design experiences [5, 6, 19] and which are part of an initial and under development design pattern language addressing communities of interaction designers, focused on designing collaborative interactive systems. The description of the top design pattern is complete. The descriptions of the other design patterns omit (due to space limitations) the forces, the set of references (available in Section 6), and the sets IN and OUT (illustrated in Figure 1).

P1: Enable Collaboration.
Problem. The growing complexity of design problems [2] and the expanding scale of design projects are moving beyond individual human capability and thus need multidisciplinary end users, owners of the problems and developers, to collaborate in order to solve them [5, 10].
Forces. The set of forces are described by: i).the challenge to provide all the end users with virtual tools, which are able to support them in their work and collaboration and ii). communication gaps arise in the collaborative design process, since end users with different cultural and contextual backgrounds use different systems of signs, languages and representations and may have different perceptions as well as different interpretations, even for the same images [5].
Examples. This problem is faced in the design of a system to support a community of mechanical engineers who come together and collaborate in the validation of an artifact [19]. In their collaboration, they use virtual tools which allow them to annotate domain documents. These annotations respect a formal, technical definition, developed over the years on the basis of the engineers' experience.
Another example is the design of a system to support the collaboration of a team of physicians in establishing a diagnosis [5]. Physicians with different roles (neurologists and radiologists)

collaborate in performing diagnostic activities on the basis of annotating (through virtual tools) the medical records they are reasoning on.
Diagram.



Keywords. K= {‘enable collaboration’, ‘diversity’, ‘design for collaboration’}.
Solution. The solution is to provide each of the end users involved in the collaboration with an instance of the system tailored to his/her own needs and background. The instance allows each end user to reason in his/her own system of signs and to use his/her preferred digital platform in the interaction with the system. The system is, therefore, localizable to the user’s domain, role and culture and to the platform in use. A shared knowledge base is made available to all the end users involved in the collaboration, to support a common ground of communication and understanding. The activity of managing the knowledge base is the annotation, which allows end users to update it by adding comments next to data in order to highlight its meaning [6].
In. IN = $\varnothing$. This is an empty set since it is associated to the top level pattern.
Out. OUT = {“*Cross-domain collaboration*”, “*Intra-domain collaboration*”, “*Cross-culture collaboration*”, “*Cross-platform collaboration*”, “*Guarantee consistency*”, “*Facilitate maintenance and reuse*”}.

P2: Cross-domain Collaboration.
Problem. End users working in different domains and using different systems of signs and notations need to collaborate in their everyday work and must be supported by virtual tools which enable their collaboration. In their collaboration, they bring together different expertise for the common goal of solving a problem.
Examples. A possible example of such situation is the design of a system to support architects or civil engineers who need to collaborate in the sketch of a blueprint. They use virtual tools able to support their communication, collaboration and information sharing.
Keywords. K = {‘cross-domain collaboration’, ‘different domains’, ‘cross-domain reasoning’}.

Solution. Designing systems which support cross-domain collaboration asks for providing each end user with an instance of a system localized to his/her domain and for an overall design model able to enable the communication among domain dependent systems. The common ground for cross domain communication consists of a boundary/bridge object language formed of inter-related boundary objects. Boundary objects are artifacts and can be utilized to support cross domain collaboration since they are adaptable to serve different domains' needs and maintain their common identities during the collaboration processes [17]. Boundary objects may be used by end users to interact with each other, reason on each other's work and exchange them.

P3: Intra-domain Collaboration.
Problem. End users working in the same domain, but having different roles and backgrounds need to collaborate in their everyday work and must be supported by virtual tools which enable their collaboration, communication and information sharing.
Examples. The design of a system to enhance the collaboration of a neurologist and a radiologist [5] in reaching a common diagnostic is an example of situation where the problem described above arises. They both belong to the same domain, but use different tools and ways of reasoning.
Keywords. K = {'intra-domain collaboration', 'different roles'}.
Solution. Designing systems to enhance intra-domain collaboration asks for providing each end user with an instance of a system localized to his/her role in the domain. We argue that specific role dependent tools should be provided for each end user. Moreover, end users belonging to the same domain should be supported in managing a common knowledge base associated to the domain. The communication among end users with various roles is made on the basis of the common knowledge base and by exchanging domain specific boundary objects.

P4: Cross-culture Collaboration.
Problem. End users belonging to different cultures, speaking different languages and using different systems of signs need to be supported in their collaborative work by virtual tools which enable their communication and common reasoning. The same system should address end users of different cultures, allowing them to come together, understand each other, share information and collaborate.
Examples. An example in this respect is the design of a system for a community of tourists which belong to different cultures and collaborate in the creation of a shared knowledge base related to a geographical region. They use virtual tools which allow them to communicate and share their feedback on their visits and, in this way, enrich the knowledge base.
Keywords. K = {'cross-culture collaboration', 'different languages', 'system of signs'}.
Solution. Designing systems which support and enhance cross-culture collaboration asks for allowing each end user to interact with an instance of a system localized to his/her culture in terms of language and system of signs. The design of such localized systems requires the

definition of a level of abstraction meant to decouple the culture-related properties of the system, allowing their specification by means of specialized languages and tools.

P5: Cross-platform Collaboration.
Problem. End users use different digital platform in interacting with virtual systems which support them in their work. They communicate, collaborate and share information through collaborative systems which may be materialized on different types of platforms.
Examples. An example of such a case is the design of a system to support the collaboration of end users using both laptops and mobile devices in their interaction with the same system.
Keywords. K = {'cross-platform', 'materialization'}.
Solution. Materializing a specific system on a set of different platforms asks for a level of abstraction which decouples the technical details characterizing the platform and the specification of how these details affect the materialization of the system on that platform. Specialized languages and tools capture information like the platform's display characteristics, the memory specification, the description of the input and output devices. Moreover, information related to the way the content and the behavior of the system are rendered on each platform must be described independently.

4 Towards a Pattern Language

The identification of a set of inter-related design patterns leads to the definition of a pattern language.

Figure 1 depicts an overall definition of the proposed pattern language addressing the design of collaborative interactive systems. The top level design pattern ("*Enable collaboration*") is refined into the 6 design patterns ("*Cross-domain collaboration*", "*Intra-domain collaboration*", "*Cross-culture collaboration*", "*Cross-platform collaboration*", "*Guarantee consistency*", "*Facilitate maintenance and reuse*"), a subset of which is described in Section 3. Each of these patterns can be further on refined into smaller granularity design patterns. The lower level patterns represented in Figure 1 are classified according to the activity dimension defined in Section 3 (reasoning, communication, data sharing).

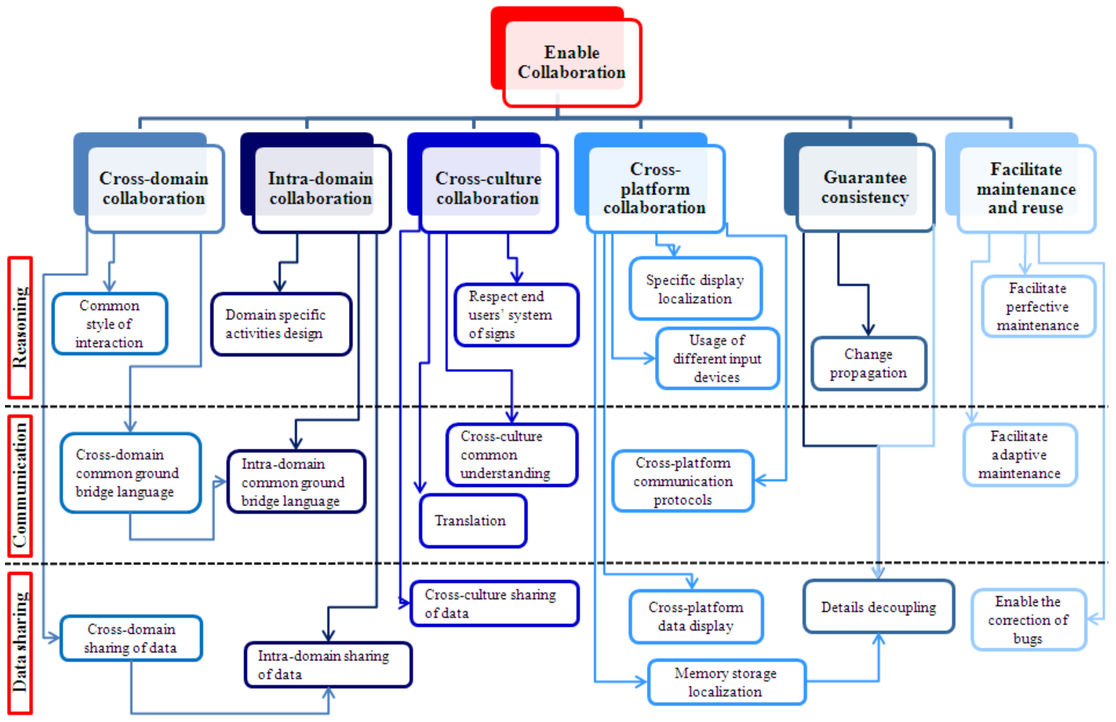

Enable Collaboration
Cross-domain collaboration
Intra-domain collaboration
Cross-culture collaboration
Cross-platform collaboration
Guarantee consistency
Facilitate maintenance and reuse
Reasoning
Communication
Data sharing
Common style of interaction
Domain specific activities design
Respect end users' system of signs
Specific display localization
Usage of different input devices
Change propagation
Facilitate perfective maintenance
Cross-domain common ground bridge language
Intra-domain common ground bridge language
Cross-culture common understanding
Translation
Cross-platform communication protocols
Facilitate adaptive maintenance
Cross-domain sharing of data
Intra-domain sharing of data
Cross-culture sharing of data
Cross-platform data display
Memory storage localization
Details decoupling
Enable the correction of bugs

Figure 1 – A pattern language for the design of collaborative interactive systems

The representation of all the relationships identified among the design patterns leads to a graph representation in which each node may point to a set of other nodes (set *OUT*) and in which a set of nodes may point to a specific node (set *IN*).

4.1 The Pattern Language in Use

The use of the pattern language described above is dedicated to teams of designers focused on the design of collaborative interactive systems. The initial set of design patterns is meant to provide a skeleton to support further creation of patterns. Based on their on-the-go experience, designers contribute to the pattern language, sharing their knowledge and wisdom. In this way, they not only make use of the solutions provided by other members of their team, but are also able to propose new solutions to existing problems or new problems with their associated solutions.

The management of the pattern language is supported by the definition of each design pattern, offering a template to support understanding and usage. Several operations are made available to the designers, like:
i). *searching* for design patterns, operation which is supported by the use of the set of keywords. Searching is needed when, faced with a design problem, a designer is interested in the solutions provided by its community. The keywords associated to each design pattern form a glossary of terms, based on which the search is being performed.
ii). *browsing* through the pattern language, which is allowed by the graph representation of the patterns and their relationships. PLML supports the design patterns representation as nodes of a graph in which the edges are the representation of the relationships among the patterns.

Browsing is highly facilitated by such a representation, helping designers get an overview of the design patterns available and the way they are related.
iii). *modifying* existing design patterns or *annotating* their description. The definition of a design pattern is comprised within a document which can be easily modified according to the designers' experience. Moreover, the content describing each design pattern can be annotated as answer to any misunderstandings or clarification requests within the community. Annotations support open discussions within the community, allowing each designer to leave his/her feedback on any edge and/or node of the graph associated with the pattern language.
iv). *creating* new design patterns by providing the elements of the description listed above. At each step, any designer may create a new design pattern according to his/her experience by filling in the description of the design pattern and by relating it to other already existing patterns.

5 Conclusions

Communities of interaction designers focusing on the design of collaborative interactive systems face a set of challenges and open issues coming from the diversity of users and technology. As answer to making designers' knowledge and wisdom available within the community they belong to, a design pattern approach is proposed. The paper identifies a set of design patterns, part of an under development pattern language, which answer the top level issues in the design of collaborative interactive systems. The extension of the pattern language follows an iterative approach in that at any time, new design patterns can be added to the language and linked to already existing patterns. As future work, we are focusing on the development of authoring and browsing tools which allow designers and end users to participate to the creation, management and sharing of a design pattern knowledge base. These tools are based on annotation, annotation indexing, knowledge base organization, and browsing support.

6 Acknowledgements

The work of Claudia Iacob and Li Zhu was supported by the *Initial Training Network* "Marie Curie Actions", funded by the FP 7 – People Programme with reference PITN-GA-2008-215446 entitled "DESIRE: Creative Design for Innovation in Science and Technology".

7 References

[1] Alexander, C. 1977. A pattern language: Towns, buildings, construction. New York: Oxford University Press.
[2] Arias, E., Eden, H., Fischer, G., Gorman, A., Scharff, E. 2000. Transcending the individual human mind—creating shared understanding through collaborative design. *ACM Trans. Comput.-Hum. Interact.* 7, 1 (Mar. 2000), 84-113.
[3] Borchers, J. 2001. A Pattern Approach to Interaction Design. John Wiley & Sons, Inc.
[4] Brancheau, J. C., Brown, C. V. 1993. The management of end user computing: status and directions. *ACM Comput. Surv.* 25, 4 (Dec. 1993), 437-482.
[5] Costabile, M. F., Fogli, D., Mussio, P., Piccinno, A. 2007. Visual Interactive Systems for End user Development: a Model-based Design Methodology, *IEEE Transactions on Systems Man and Cybernetics* 37, 6 (2007), 1029 – 1046.

[6] Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A. 2006. End user Development: The Software Shaping Workshop Approach. In Lieberman, H., Paternò, F., & Wulf, V. (Eds.): End user Development, pp. 183-205. Dodrecht: Springer.

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

[8] Díaz, P., Rosson, M. B., Aedo, I., and Carroll, J. M. 2009. Web Design Patterns: Investigating User Goals and Browsing Strategies. In *Proc.of the International Symposium on End user Development* (Siegen, Germany, March 02 - 04, 2009). V. Pipek, M. B. Rosson, B. Ruyter, and V. Wulf, Eds. Lecture Notes In Computer Science, vol. 5435. Springer-Verlag, Berlin, Heidelberg, 186-204.

[9] Ellis, C. A., Gibbs, S. J., and Rein, G. 1991. Groupware: some issues and experiences. *Commun. ACM* 34, 1 (Jan. 1991), 39-58.

[10] Fischer, G., 2000. Social Creativity, Symmetry of Ignorance and Meta-design. *Knowledge-Based Systems Journal*, Vol. 13, No. 7–8, pp. 527-37.

[11] Fischer, G. 2004. Social creativity: turning barriers into opportunities for collaborative design. In *Proc. of the PDC'04*, Toronto, Canada, July 27 - 31, 2004, ACM, New York, NY, 152-161.

[12] Guerrero, L.A., Fuller, D.A. 2001. A pattern system for the development of collaborative applications. *Information and Software Technology*, 43(7), 457-467.

[13] Hornbæk, K. and Frøkjær, E. 2003. Reading patterns and usability in visualizations of electronic documents. *ACM Trans. Comput.-Hum. Interact.* 10, 2 (Jun. 2003), 119-149.

[14] http://www.cs.kent.ac.uk/people/staff/saf/patterns/diethelm/plmlx_doc/index.html

[15] Reenskaug, T.M.H. 2003. The model-view-controller (MVC) - its past and present. 2003. JavaZONE, Oslo.

[16] Snow, C. P., 1959. The Two Cultures and the Scientific Revolution, Cambridge University Press, Cambridge.

[17] Star, S. L. and Griesemer, J. R., 1989. Translations' and Boundary Objects: Amateurs and Professionals in Berkley's Museum of Vertebrate Zoology, 1907-39, *Social Studies of Science*, Vol. 19, No. 3, pp. 387-420.

[18] Tidwell, J. 2005. Designing Interfaces: Patterns for Effective Interaction Design. O'Reilly Media

[19] Valtolina, S., Mussio, P., Barricelli, B. R., Bordegoni, M., Ferrise, F., Ambrogio, M. 2009. Distributed knowledge creation, recording and improvement in collaborative design. In *Proc. of KES IIMSS '09*. E. Damiani et al. (Eds.), SCI 226, Springer-Verlag, New York, 31-41.

[20] Welie, M. Patterns in interaction design. Retrieved: June, 20th 2010. Available at: http://www.welie.com