Design Patterns For Collaborative Learning: from practice to theory and back

Design Patterns For Collaborative Learning: From Practice To Theory And Back

Rune Baggetun
InterMedia, University of Bergen
Norway
rune.baggetun@intermedia.uib.no

Ellen Rusman
Open University of the Netherlands
The Netherlands
Ellen.Rusman@ou.nl

Caterina Poggi
Milano Politechnico
Italy
poggic@lu.unisi.ch

Introduction

We see that the concept of design patterns and the idea of a pattern language (Alexander et al. 1977) can help in capturing and communicating recurrent instructional design problems in e-learning. E-learning, (learning partly or fully supported by ICT and digital resources), is becoming more and more widespread in educational institutions. Knowledge and experience about e-learning are growing as research is conducted and educational institutions acquire more practical experience in developing and running e-learning courses. This knowledge exists in various formats: as both tacit and implicit knowledge inside the heads of practitioners, in written manuals ('best practice' documents, etc.), in research papers and books. The knowledge (tacit, implicit and explicit) comprises know how and research findings about a whole range of issues, from organisational and didactical/pedagogical issues to pure technical knowledge. Another important trait is the multidisciplinary (and multi-profession) characteristics of the people researching and working with e-learning: the creators and owners of e-learning knowledge are from various disciplines such as computer science, psychology, teacher education and social science. They each have their own perspectives on e-learning.

We contest that the knowledge and perspectives from different fields would be more easily communicable with a more shared and interdisciplinary means of communication at hand. We need a means of communication that will enable us to exchange, talk about, integrate and leverage existing knowledge from different disciplines and professions. We see this as important for both researchers and practitioners.

Practitioners need access to research results in an understandable format;researchers could benefit from the knowledge that exists among practitioners to a higher degree if such a reference mean was available. This is where we envisage design patterns as a powerful communication tool. Patterns can capture effective learning designs, allow newcomers in education to learn from more experienced developers, provide solutions that can be applied in many circumstances, function as a communications medium between experts of (different) disciplines and be a means to enhance knowledge management and knowledge transfer (Brouns 2004, Jones & Stewart, Bergin, et al. 2004).

A design pattern seeks to capture the essential bits of a problem - solution couple in a specific context, and presents it in a way so that it can be applied and adapted in different settings by non-expert practitioners, and even users (Dearden, Finlay, Allgar, McManus 2000). Thus patterns can mediate knowledge and transmit 'expert' expertise. In a design pattern special attention is given to the forces which are acting on the problem and the rational for choosing a particular solution. Some patterns are more mature than others (benefitting from many years of experience and research), while others are just in their 'infant years', ready to be more thoroughly explored and elaborated. To indicate this difference between more proven patterns and initial ones, Alexander used the notion of asterixes (*) to annotate mature or immature patterns. This categorisation of patterns into mature patterns and new 'infant' patterns

fits nicely with the field of e-learning, where we, on the one hand, see mature knowledge about e-learning design problems, and where, on the other hand, there's still much to explore and research.

CSCL Patterns

Computer supported collaborative learning (CSCL) is a field of research and design that focuses on the use of ICT as a mediational tool within collaborative methods of learning. From this perspective, technological tools are seen as artefacts that mediate the way we think, work and learn. In addition, technology and infrastructure do not exist in isolation, but in ecological relation to other artefacts such as language and symbolic systems (e.g., graphical representations, schema, diagrams, and illustrations).

The complexity inherent in the design of CSCL is strongly demonstrated in the research field of CSCL and how designers struggle to make genuine collaborative learning a part of e-learning courses. Conceptual tools are needed to refine and combine research findings and current practices in order to make them applicable for both researchers and practitioners.

The adaptation of the design pattern approach to object oriented programming (OOP) started with 'The Gang of Four's famous book (Gamma, Helm, Johnson, Vlissides 1995). This has been seen as a success and has led to the development of a pattern language for object oriented programming that enables software designers to talk about problems by the names of their pattern (e.g. the singleton pattern or the MVC pattern). In this way they have build a language that makes communication and the generation of quality software easier among both novices and experts.

E-learning patterns are different from OOP patterns. E-learning patterns concern how technology mediates learning that is based on social interaction. This is completely different from the work of OOP patterns and even HCI patterns, which also are a more narrow interpretation of Alexander's original ideas. In addition, CSCL involves interrelations between the social and the technical. E-learning and CSCL correspond, in our opinion, more closely to Alexander's notion of design patterns than how they have been used in the field of OOP. Design patterns for CSCL must include and take into account both material (including technological) issues and social issues. (This is not a criticism of the DP work in OOP, but just an observation that e-learning design patterns have to take into account more of the aspects and ideas of Alexander.) How students interact and live in and with the material is an important design problem when designing for CSCL; it's not only the technical aspects that matter - it's about social-technical design. As Preece (2000) states: 'In sociotechnical systems design, system refers to the whole network of users, technology and environments in which the system will be used. The design process explicitly acknowledges that the design has a social and political cachet. Design, therefore, cannot be independent of the social system in which it will be implemented'.

The first step in making an e-leaning pattern language, and in particular a CSCL pattern[1], is to identify problem areas (or issues). The rest of this paper will discuss how CSCL problems can be found and developed into patterns. This work is drawn from experiences gained while participating in the E-LEN project (see http://www2.tisip.no/E-LEN/) when we were trying to compose different patterns for CSCL. The E-LEN project consists of both researchers and practitioners and we have been able to start cataloging patterns by investigating both the practice and research within e-learning. We have seen that we have been using different methods to identify and distill patterns inside the project.

Pattern Identification and Destillation

Although a lot has been written about the concept of design patterns in software engineering and their possible uses, less has been written on the actual identification and deriviation ('pattern mining') processes which are necessary to construct a pattern. Some articles focus on the notation of a pattern once it has been identified (e.g. Meszaros & Doble 2003) and others give general tips on how to start with constructing patterns (e.g. Lea 2001, Cunningham 1994), and evaluate them by means of a shepherding process (Harrison 2000). But how can we first identify a pattern, in particular patterns for socio-technical systems where system features have to suit the social organisation?

[1]Alexander et al. operates with patterns at different levels in a pattern language: from regions, to towns to squares and houses.

Whereas a pattern is always a 'description of a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over' (Alexander 1977) it would be logical to expect that methods of pattern generation are mostly inductive: the deriviation of general principles from particular facts or examples related to the problem. Nevertheless, in practice, as experienced in the E-LEN project, deductive thinking methods also seem to help as a mind-tool supportive to the generation of patterns.

Next to this, there's another distinction in pattern identification and deriviation: the one of non- automatic pattern construction (pattern mining that is heavily based on thinking (mental) processes, observation and expert experience) and the one of semi-automatic pattern construction, in which the shared elements of often-applied solutions to particular problems are destilled out of a large set of instances and then completed by expert experience with these patterns. Within this broad specification of methods, several methods can be distinguished. Each of these pattern mining methods is decribed separately below to raise awareness of it's use, but this does not imply that they cannot be combined to produce patterns. Also, almost all methods involve 'questioning': posing questions and trying to get answers which help to solve the problem. Because this is a technique that is used within almost all methods, it's not described separately.

Inductive Pattern Mining – From Specifics To Generalisations

Generally, the construction process of these patterns is heavily dependent on expert experience and expert judgement of practices. Recently, there have also been some ideas to derive patterns in a more objective way, by combining automated deriviation of patterns with human experience (Brouns, e.a. 2004).

From instance/example to pattern	An expert thinks of a good or bad example and tries to pinpoint the reason for this judgement by posing questions, e.g.: *Why is this specific solution so good or bad in this situation? What factors contribute to my judgement? Are this factors transferable to other situations?. What forces and rationales have probably contributed to exactly this affordance?*
From observation of human behaviour to pattern	An expert tries to explain human behaviour in an environment by relating the behaviour to attributes of the environment, and tries to express them in positive or negative statements, e.g.: *The 'high interaction of people' is probably partly due to environmental attribute A, D and X because it improves/stimulates/supports/… or surpresses/breaks down/disturbs… underlying principle I (e.g. trust)* This principle can be applied for any kind of environment which is designed to support human behaviour and wellbeing.
From interrelations to pattern	An expert demarcates a possible pattern within a domain but discovers that this pattern actually consists of several subpatterns: the actual problem consists of subproblems, each of which has a separate solution within the defined context. These patterns should be distinguished and separately described. In this way, a pattern language can emerge (a relational network between patterns).
From multiple instances to pattern	As Brouns et.al. (2004) state, patterns are not solely created by human cognitive processing, but can be partly detected from patterns in the data: an abstraction of a set of best practices to fulfill a recurrent design problem. The condition for an automated approach of this kind of deriviation is that the courses must be structured in a machine interpretable way, e.g. IMS Learning design specification (IMS LD, 2003) to visualize the data and/or detect patterns in the data. Several techniques might be used, like latent semantic analysis, matrixes and the visualisation of XML tree algorithms (Brouns et.al, 2004). Forces and rationales acting upon these patterns are described additionaly to these 'automated patterns'. A comparable concept is mentioned in Derntl and Motschnig-Pitrik (2004), although no automated derivation methods are described, but patterns are derived 'manually' from comparison of UML-models of several implemented, succesful courses (selection based on evaluation results).

Deductive Pattern Mining - From Generalisations To Specifics

From metaphor to pattern

An expert thinks of the attributes of one general, familiar environment and the functions of these attributes and tries to translate these functions (by mechanisms of similarity and analogy) towards functions of another type of environment, e.g., to stay close to the origin of design patterns as a concept of the architect Alexander (1977), the 'building metaphor':

> *Each building has doors. What is the function of doors? It closes spaces for outsiders and helps to keep spaces private and/or enclosed, it's a barrier to enter, one needs allowance (e.g. by means of a key, the doorbell, knocking) to enter when it is closed, it enables people to go from one space to another, it's part of a separation of spaces (in combination with walls).*

Supposing that most of these functions are also necessary in a virtual environment, how can these functions be 'translated'. Examples of patterns which seem to be constructed in this way are the patterns 'door', 'bell', 'room' from the 'Patterns 4 groupware' initiative (Schümmer et.al, 2004).

From mindmap to pattern

An expert explores a concept by drawing a mindmap. A mindmap consists of a central concept and relates associated concepts to this centralized one. These mindmaps might be organised in several ways. Some possible ways are:

- free exploration of a central topic - draw all associations related to the central concept and organise them.
- identify a problem and draw a mind map with this problem in the centre and the six universal questions (what, where, how, when, why, how) as nodes on the map (Cave, C., 2003). From there on, explore the problem with the questions as trigger for associations.
- As Cunningham described (1994), make a list or map of little things you have learned throughout the years (subject matter that is practical but seldom explored in a textbook). Cunninham: 'Imagine that your kid brother has just taken responsibility for this area on his first big job. You are going to introduce him. Cast each item on your list as a solution and write a sentence with 'therefore' in the middle and figure out the forces that bear on your solutions'. Each solution can be a central concept in a mindmap.

> *With these maps as guidance an expert reflects on them: is this concept a solution? What is it a solution for? Is it a problem? Is it already solved? How is it solved? Why is this solution chosen?*

In this way an information base for a new pattern is emerging.

From experience to pattern

An expert confronts him/herself with a recurrent problem and tries to write down all considerations, factors and concepts which he/she is taking into account while solving the problem, purely based on his/her experience. After this, they try to organise these considerations into the following rubrics:

- context
- forces
- rationales
- solution

Then they attempt to 'abstract' the constructs mentioned by asking 'is this construct suitable while solving a comparable problem? If not, why not? If yes, why?' Only transferable, abstracted concepts are written down in a pattern.

It is probable that the above-mentioned pattern mining methods are not complete, but they are a first collection derived mainly from reflection upon pattern mining processess used within the E-LEN project and additional articles. This collection of methods is meant to support inexperienced pattern writers who are trying to define patterns.

For more information on the pattern writing process itself, and examples of possible formats that can be used, several sources can be mentioned (see references), mostly derived from pattern production within the software community:

- a pattern definition from Coplien (2003) at the Hillside site

- a pattern for pattern writing from Meszaros and Doble (2003)
- example patterns from The pedagogical patterns project (2004)
- example patterns for cooperative interaction from the Pointer project (2004).

Once a pattern has been identified it has to be verified and evaluated. This can be done in different ways, e.g. by evaluation and review by other practitioners (designers/developers), by implementation of the pattern in practise and measurements of its success (is it solving what it is meant to solve or is the typified problem still occuring?), by evaluation of its use by novice designers. But, in this article, we concentrate on the identification and creation of a pattern. Part of this creation process can be feedback by fellow practitioners. This feedback is mostly organised in a particular way, e.g. by writer workshops, expert review, conferences (e.g. PLOP: Pattern languages of programs) and a system of role division such as 'sheep' and 'shephards' in the writing process.

For more information on the processess of feedback and evaluation of patterns by fellow practitioners several sources can be mentioned:
- a pattern language for writer's workshops from Coplien (1999)
- the language of shepherding from Harrison (2000).

If you discover new methods of producing patterns while trying to identify and construct them yourself, the authors of this paper will appreciate your feedback.

References

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). A pattern language : towns, buildings, construction. New York : Oxford University Press, 1977.

Brouns, F., Koper, R., Manderveld, J., van Bruggen, J., Sloep, P., Rosmalen, P., Tattersall, C., Vogten, H. (2004). An exploration of technologies for the inductive analysis of Learning design patterns. [in press].

Cave, C (2003). The Creativity Web. Resources for creativity and innovation. Retrieved April 22, 2004, from http://members.optusnet.com.au/~charles57/Creative/index2.html

Coplien, J.O. (1999). A pattern language for writers' workshops. Lucent Technologies. Retrieved April 22, 2004, from http://www.bell-labs.com/user/cope/Patterns/WritersWorkshops

Coplien, J.O. (2003). A pattern definition- software patterns.

Cunningham, W. (1994). Tips for writing pattern languages. Retrieved April 22, 2004, from http://c2.com/cgi/wiki?TipsForWritingPatternLanguages

Dearden, A., Finlay, J., Allgar, L., McManus, B. (2002). Using Pattern Languages in Participatory Design. Position Paper at CHI2002 Patterns Workshop. Available at http://www.welie.com/patterns/chi2002-workshop/Dearden-CHIWorkshopPaper.pdf

Derntl, M., Motschnig-Pitrik, R. (2004). A pattern approach to person-centered e-Learning based on theory-guided action research. In Conference Proceedings of the Networked Learning conference 2004.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (January 15, 1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Pub Co; 1st edition

Harrison, Neil, B (2000). The Language of Shepherding: A Pattern Language for Shepherds and sheep. 7th. Pattern Languages of Programs Conference August 13 to 16, 2000 Allerton Park Monticello, Illinois, USA Available at http://hillside.net/patterns/EuroPLoP2001/shepherding.doc

Hermann, T. e.a. (2000). Concepts for usable patterns of groupware applications. ACM.
IMS (2003). IMS Learning Design Specification. Retrieved April 22, 2004, from http://www.imsglobal.org/learningdesign/index.cfm

IMS LD (2003). IMS Learning Design Information Model. Version 1.0. Final Specification. Retrieved June 10, 2003, from http://www.imsglobal.org/learningdesign/Idv1p0/imsle_info1p0.html

Lea, D. (2001). How do I go about writing a pattern? Retrieved April 22, 2004, from http://gee.cs.oswego.edu/dl/pd-FAQ/pd-FAQ.html#qwrite

Meszaros, G., Doble, J. (2003). A pattern language for pattern writing. Retrieved April 22, 2004, from http://hillside.net/patterns/writing/patterns.htm

Pointer project (2004). Patterns of interaction: a pattern language for CSCW. Patterns of cooperative interaction. Lancaster University. Retrieved April 22, 2004, from http://www.comp.lancs.ac.uk/computing/research/cseg/projects/pointer/patterns.html

Preece, J.(2000). Chp.6. Research spreaks to practice: groups - Social informatics. p 194-196. In: Online communities, designing usability, supporting sociability. Chichester: Wiley & Sons.

Schümmer, T (2004). Patterns 4 Groupware Swiki. Retrieved April 22, 2004, from http://wwwpi6.fernuni-hagen.de:8080/gw-patterns/

The pedagogical patterns project (2004). Retrieved April 22, 2004, from http://www.pedagogicalpatterns.org/

Weinberger, A. Fischer, F. & Mandl, H. (2001) Scripts and scaffolds in problem-based CSCL environments: Fostering participation and transfer. EARLI 2001, Retrived September 4 2003 from http://home.emp.paed.uni-muenchen.de/~weinberg/download/EARLI01Weinberger.pdf

Acknowledgements

We would like to thank the members of the E-LEN project for valuable collaboration, and in particular Sonia Bartoluzzi for her assistance. The E-LEN project is funded by the European Commision under their Socrates programme.