A Pedagogical Application Framework for Synchronous Collaboration

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

by

Aiman Turani

The University of Sydney

School of Electrical and Information Engineering

The University of Sydney

September 2007

Abstract

Designing successful collaborative learning activities is a new focus of research within the E-Learning community. The social dimension inside the traditional face-to-face collaborative learning is important and must be included in the online learning designs. In this thesis, we introduce the concept of Pedagogical Application Frameworks, and describe Beehive, a pedagogical application framework for synchronous collaborative learning. Beehive guides teachers in reusing online collaborative learning activities based on well-known pedagogical designs, to accomplish their educational objectives within a certain educational setting, and also simplifies the development of new pedagogical collaboration designs. Beehive's conceptual model has four abstraction layers: Pedagogical Techniques, Collaboration Task patterns, CSCL Components, and CSCL script. By following the framework's guidelines and specifications, developers will place the control of designing pedagogical collaboration tools in the teacher's hand rather than in the software designer's.

Acknowledgements

In the name of God most merciful and graceful, I would like to thank Rafael A. Calvo for his supervision of my thesis, and for giving me the opportunity to pursue this project. I would also extend my thanks to Peter Goodyear for his role in co-supervising my thesis, the Web Engineering group for its support and guidance: Nick Carroll, Sergio Freschi, Zhang Danyu, Zhang JingYu, Di Dong, Juan Jose Garcia Adeva, and Ernesto Ghiglione, the CoCo team for its assistance, especially, Peter Reimann, Dorian Peters, Adam Ullman, Miriam Weinel, and the NSW Institute of Sport for this research funding.

On a personal level, I dedicate my efforts on this work to my Mother and Father, Ammal and Ahmad, eternal supporters of my effort, to my beautiful and beloved wife, Heba, and to my children, Ahmad, Yaman, and Sarah, who keep me from sliding into some dark place.

Preface

This thesis is the culmination of a PhD project in the Web Engineering Group at the University of Sydney's School of Electrical and Information Engineering. The project has produced two products: one is this thesis and previous publications, describing a new framework for synchronous collaborative learning, and the other is the application software "Beehive", which can be used by teachers, trainers, researchers, etc. to design and conduct online collaborative activities.

In order to design such a framework, research was conducted on the current learning management systems, theories of learning design, pedagogical patterns, and CSCL, as well as research on the potential technologies that could be used in the implementation of such a framework. The discourse in this thesis does not assume deep prior knowledge in collaborative learning frameworks, but it assumes some familiarity with basic concepts of collaborative techniques, learning designs, and CSCL.

Availability

The latest version of Beehive is always available at: http://www.weg.ee.usyd.edu.au/projects/beehive/. All source code, documentation, and animated examples are included in the distribution. The animations include several use-case examples that contain interactive interfaces to simulate the designing and the run-time process. There is also a sample testing server section in which users are allowed to use our WEG server to test and try it with real users before deciding to install Beehive on their machines. For users who decide to install Beehive, there is a detailed section on how to download Beehive from the openacs repository along with some information on the Media Server installation.

After submission to the University of Sydney for examination, this thesis document will be available in electronic format at: http://www.weg.ee.usyd.edu.au/people/aimant/thesis.pdf, and in hardcopy format from the University of Sydney's Engineering Library.

Motivation

I was attracted to this project for various reasons. But one interesting reason was wondering what makes online chatting so popular. You can find people spending long hours chatting online with others who are next-door or reachable by cheap phone calls. Can we use such systems to encourage learners to communicate and learn in the same enjoyable atmosphere? When we look at the past and future, we expect that online teaching and learning will still continue to change. My own sense is that the active form of learning will be the most dominant. I hope that this work will be one of the significant milestones in facilitating collaborative learning in the online environment.

Licensing

Publications and Presentations

During the course of candidature on which this thesis is based, the following contributions were accomplished:

Peer-reviewed Papers:

- Turani, A. and Calvo, R.A. Beehive: a software application for synchronous collaborative learning. *Campus-Wide Information Systems*, *23* (3). 196-209.
- Turani, A. and Calvo, R., Sharing Synchronous Collaborative Learning Structures using IMS Learning Design. In *the International Conference on Information Technology Based Higher Education & Training*, (Sydney, 2006).
- Turani, A., Calvo, R. and Goodyear, P., An Application Framework for Collaborative Learning. In *the International Conference on Web Engineering* (Sydney, 2005), Springer-Verlag, 243-251.
- Turani, A. and Calvo, R., The Potential Use of Collaboration Scripts in Synchronous Collaborative Learning. In *the International Conference on Interactive Mobile and Computer Aided Learning* (Amman, Jordan, 2007) .
- Turani, A. and Calvo, R.A., Educational design patterns with Beehive. In the *Conference on Education and Technology in the Middle East* (Manama, Bahrain, 2007).

Workshops:

- A presentation on the implementation of the computer-supported collaboration scripts using Beehive was delivered during an invitation to attend the international workshop "The CSCL Alpine Rendez-vous workshop", 21-26 January 2007.
- A presentation on the effective interactive interfaces for CSCL was delivered at the national workshop "The Australasian Society for Computers in Learning in Tertiary Education workshop", 3–6 December, 2006.

- A presentation on how to support group-based training in sport settings was delivered at the Sport Technology workshop, 22 December 2006.
- A presentation on how to use CSCL innovations to support group-based training was delivered at the Safety Training in Australia Symposium, 31 March 2006.

Award

Beehive was the winner of the postgraduate prize for Next Generation Application from the Sydney University Research Conversazione in 2005.

Declaration

Except where noted in the text, this dissertation is the result of my own work.

I hereby declare that this dissertation is not substantially the same as any that I have submitted for a degree or diploma or other qualification at any other University. I further state that no part of my dissertation has already been or is being concurrently submitted for any such degree, diploma or other qualification.

Contents

List of Figures

List of Tables

1 Introduction

The learning opportunities provided by E-Learning technologies are encouraging universities to systematically invest in them to enhance the student experience. This trend has generated a renewed interest in reusing successful instructional designs. The quality improvement and cost reduction have driven researchers to study how to increase the reuse of learning content and instructional design. Today, a lot of the research by learning technologists emphasises the reuse of "learning objects" (a way of packaging content modules). Regrettably, this work might reinforce the idea of learning as information "transfer" (from the teacher or content to the student) which is a cause of low learner motivation, low engagement, and isolation [119]. Learning occurs when students are actively involved in solving a problem within a social environment [78]. Recent pedagogical research shows that learning [92] is not simply knowledge assimilated with the help of a more knowledgeable person or a computer system, but also jointly constructed via problem-solving with peers [133]. Learning through peers' conversations is one such form of learning and teaching. It has been the focus of relatively early theories of learning [105], and it has been closely associated with a quality approach to teaching and learning by a process of collaboration.

As a result, there has been a trend among university teachers, in which collaborative learning takes priority over content delivery [139]. This trend has significant implications for online teaching, whereby software systems must be built to support pedagogical designs. Companies and universities have addressed this new paradigm with a variety of approaches, but in most cases these institutions lack software

systems that embody these teaching strategies and support teachers in their designs. Where these systems exist, they tend to focus more on managing content rather than communication and collaborative learning [67].

There are two main categories of collaborative learning: asynchronous, and synchronous. The asynchronous collaborative learning can be performed at any time and at any place, providing more freedom [85]. Even though the asynchronous approach has been dominant in recent years, there are many challenges to this kind of collaboration, such as time delays between responses, lack of real-time sense for a group, etc.

We decided to focus in this thesis on synchronous collaboration, since it is closer to the face-to-face model and has the potential, if well designed, to substitute for face-to-face in blended learning due to several advantages (ability to record interactions, ability to maximise interactions, freedom of space requirements, etc.).

Nevertheless, designing synchronous collaborative activity is considered to be more resource intensive (facilitator's time, effective planning, appropriate technical background, etc.) than the traditional way of teaching [102]. General synchronous tools (chat, whiteboard, etc.) that are proposed to support informal synchronous collaborative learning are not always appropriate or sufficient to build a meaningful learning experience, while specialised tools that can be used to support specific synchronous collaborative learning methods are rare, crude and difficult to reuse [14].

Therefore, there is an increasing need for a system that can guide teachers in implementing their online activities based on well-known pedagogical techniques, and at the same time simplifies the development of pedagogical collaboration tools needed to carry out those techniques.

We propose in this thesis Beehive, a pedagogical application framework for synchronous collaborative learning, which enables instructors to reuse and implement a wide range of collaborative learning design patterns in the online environment.

Design patterns were introduced to explain urban design ideas by Christopher Alexander [4] in the 1970's. In the 80's, patterns were adopted by the software

engineering community as a way of communicating useful and proven software designs [50]. Around 10 years later, in 1995, the PPP (Pedagogical Pattern Project) [107] started to encourage the capture and dissemination of successful teaching/learning experience in education using the pattern approach. Educational design patterns are closer to Alexander's notion of design patterns [10] than software design patterns, since educational design patterns take into account both material and social issues. This came together with all the trends of reusing educational designs by using standards such as IMS LD.

After 10 years of using design patterns in software engineering, the software engineering community found it useful to combine that technique of reusing designs with software components that support reuse of source code. The software engineering researchers introduced the concept of application framework to benefit from both Design Patterns and Software Components [47].

After several years of introducing design patterns in education, this thesis introduces the concept of pedagogical application frameworks in education. This concept adapts what we believe most valuable from the applications frameworks, as used in software engineering, and uses it as a base for designing software applications needed to implement learning design patterns. The difference between the pedagogical application framework and software application framework is that the design patterns being reused are pedagogical not software. The pedagogical application framework's abstraction is driven by the educational discipline, while its implementation is driven by the software discipline, which makes it easier for teachers to design pedagogical tools that can be reused in other contexts.

We started abstracting our pedagogical application framework by first modelling some of the research-based pedagogical techniques (brainstorming, debate, and jigsaw). We found out that there are many common components embedded within them, such as forming groups, provisioning topic information, voting, group discussion, etc. We have presented these components as a common list of tasks, in which we successfully reused them to form many other well-known pedagogical techniques (group nomination, group

discussion, round-table discussion, role playing, etc.). Identifying these tasks has helped us in defining software components that can be automatically mapped to these tasks, hiding the technical difficulty from instructional designers.

Although tasks are a core component in designing collaboration activities, they are not sufficient to run the activities. A collaboration script is needed to organise these tasks in a certain order according to the activity's basic scenario. Identifying a collaboration script is not trivial, since pedagogical patterns take into account both material and social issues. Roles and group settings need to be specified, along with their tasks. There is a big potential benefit from applying this new paradigm in the ability of recording successful collaborative activity structures that can be applied to other teaching situations. In this framework, we have provided the basis of modelling the scripting language to describe, formalise, and synchronise the collaborative learning process.

Our proposed pedagogical application framework for synchronous collaborative learning is not based only on synchronous software components but also on other types of components: asynchronous components, static components, non-software component, etc. Synchronous collaboration primarily indicates that the system should synchronise a session's progress in a synchronised way. For instance, learners may first join a common session to review their tasks then they may break up to investigate these tasks. That might be done by, for instance, visiting a library, and it might be undertaken over a long period of time (for example, days). This process is still considered to be synchronous, since learners break and meet at specified times in a synchronised manner.

We also present Beehive's implementation that is based on the framework's conceptual model abstractions, including a number of important design elements that are common in any application framework, such as Modularity, Extensibility, Reusability, Representation, etc.

We suggest the division of the Beehive's implementation architecture into two main parts: authoring and run-time. We also suggest appropriate standards to enable these separate systems to communicate properly with each other. The Beehive's

authoring application publishes a schema that represents the collaborative learning designs according to these standards. The published schema will be interpreted by the Beehive's run-time application to request appropriate Application Program Interfaces (APIs) specified in the design.

For extensibility, the Beehive's authoring application has the capability of adding new components and the ability to integrate in any LMS (Learning Management System). This provides many potential benefits, such as it enables the authoring application to inherit various LMS features (user management, course managements, etc.), and it enables the presentation of the collaboration sessions' run-time interactions within LMS assessments and reports.

For reusability, Beehive has the capability of reusing software components in creating new CSCL (Computer Support for Collaborative Learning) components to support new designs.

The representation of related components needed to drive the desired activity is undertaken automatically by the Beehive's authoring application. Depending on (a) the learning objectives, (b) what kind of problem students are asked to solve and (c) the context of the problem, teachers have two design options. First, the teacher may select an existing design from a pedagogical design repository. In this case, Beehive automatically presents a default list of tasks, a default group structure (how students would be grouped), and a default session script. It also asks teachers to provide the session's resources. At this point, a teacher's participation in the design process is complete and Beehive maps each task to its software component internally and then assembles all selected components into a single collaboration tool. The second option is for the teacher to design an entirely new pedagogical technique. In this case, the teacher needs to 1) select tasks to be performed by both teacher and learners, 2) define the group structure, 3) define the session script, and 4) provide the session's resources.

In this project, we have presented various contributions to the CSCL field. We have introduced the concept of pedagogical application frameworks that combine collaborative learning design patterns with software components. Also in the process of

designing Beehive, we have first proposed a list of common tasks that can form a wide range of collaborative techniques [126]. Then we have proposed a collection of CSCL components that can be mapped automatically to facilitate various collaborative tasks [127]. Since collaborative learning is a structured approach, we have proposed a collaboration script language that can be used to sequence the CSCL components according to designs' structures and at the same time be easily managed by instructors [124]. To achieve interoperability with other systems, we have proposed three sets of IMS LD extensions to enable IMS LD to specify how a group would interact inside a synchronous collaborative learning session [125]. We also presented the Beehive's design elements and the architecture of both environments (authoring and run-time) [20]. We have also proposed a communication protocol for launching collaborative learning sessions and exchanging data between the collaboration authoring application and the run-time application. Lastly, we have modelled 16 well-known face-to-face collaborative techniques [128] and stored them in "Beehive's Repository".

This thesis is divided into six chapters, including this one. Chapter 2 covers the background research on which we have based our own ideas on E-Collaboration. This chapter starts by introducing the research literature related to collaborative learning and its relation to E-Learning. It also introduces CSCL as a new field that aims to facilitate the adoption of collaborative learning in the online environment. Then this chapter points to the various CSCL challenges that are limiting its popularity and the Design Patterns ideas, as a way to overcome some of these challenges. Lastly it provides the literature on scripting and standards, and how they can be used to represent collaborative designs in a systematic way.

In Chapter 3, Beehive's conceptual model is described in detail presenting Beehive's various theoretical aspects and components. This chapter also describes the Beehive's evolution from the 2-layered to the current 4-layered model. It explains how the four layers (Collaborative Patterns, Task patterns, CSCL Components, and Collaborative Script) are related to each other.

In Chapter 4 we delve into the description and implementation of Beehive. It describes Beehive's architecture and its design elements to provide more concrete guidance to software developers, simplifying the development of computer applications that support synchronous collaborative learning. This chapter also explains in detail how various components are related within Beehive that enables teachers to reuse and implement pedagogical collaborative learning designs.

Then in Chapter 5, we evaluate Beehive based on three cases and by comparing Beehive with other CSCL applications, and finally in Chapter 6 we present our conclusion on this project.

2 Background

This chapter presents the research literature on the current theories, developments, and standards regarding collaborative E-Learning that represents the focus of our proposed pedagogical application framework. It provides the theoretical background on which Beehive is based. Since our Section 2.1 introduces the research literature related to collaborative learning. It starts by introducing E-Learning and how it is moving toward adopting collaborative learning. Our research aims at improving the impact of e-learning, so first we focus on current challenges in E-Learning, for instance, why most online teaching/training organisations suffer from high participants' dropout rates, and discusses how the collaborative learning can help in overcoming these challenges. It describes the general benefits of collaborative learning, and how it positively affects the various types of learning (distance learning, traditional learning, and life-long learning). Third, it points out the specific challenges associated with collaborative learning, of which we needed to be aware while designing Beehive, including design complexity, teachers' technical skills, resource limitations, etc. It also describes how instructors' roles are changing (from formal lecturing to facilitating, coaching, etc.) and how learners' attitudes and expectations are also changing. Finally, this section focuses on the synchronous collaborative learning that represents the scope of our pedagogical application framework. It presents in detail the benefits of synchronous collaboration over asynchronous and face-to-face collaboration.

Since the main objective of Beehive is to support collaborative learning, Section 2.2 introduces the literature on CSCL development and how this new field that has emerged, underlining the importance of collaborative learning and the current solutions (tools and projects) developed to support various collaborative learning styles. It states the different educational models that have influenced the development of such projects. It also presents the various CSCL challenges from which these projects suffer, and which have limited its popularity.

Frameworks are composed of Design Patterns and Components. Section 2.3 introduces the literature on Design Patterns and how they can be used to improve the reusability of successful collaborative learning designs. Then it presents some face-to-face collaboration techniques as candidates of these patterns. It describes the current collaborative learning models that are based on implementing these patterns, including their main challenges.

Section 2.5 presents the collaboration script, which is a core component in Beehive, and expresses its necessity in representing the solution part of the collaborative learning patterns. It explains why structuring is important to reach up to the complexity of the collaborative learning designs (roles and groups settings, tools, etc.). It describes the various attempts to define a collaboration script and it points to the various limitations found in these attempts. Finally, it presents some of the general collaboration scripting language characteristics that need to be addressed in Beehive.

Finally, Section 2.6 introduces the literature on standards, their benefits, and how they can represent the collaboration scripts defined in Beehive in a formal way to implement collaborative learning designs.

2.1 Collaborative E-Learning

2.1.1 E-Learning

In the last few decades, there has been a constant growth in E-Learning. It started from the discovery of radio, through to films, TV, VCRs, PCs, and the Internet [140]. All these developments have contributed to the idea of flexible delivery mode for learning materials [94]. Nevertheless, the growth of the Internet has had the greatest impact on E-Learning. A common definition of E-Learning is "the delivery of educational content via any electronic media" [122]. We prefer Goodyear's definition "E-Learning is a learning in which Information and Communication Technology is used to promote connections: between one learner and other learners; between learners and tutors; between a learning community and its learning resources" [54]. This definition is comprehensive and focuses on both learning resources and human communication.

Currently, E-Learning is seen by many researchers as a web-based learning that allows flexible and easy access to various learning resources [5]. As the Internet expands, the use of the Internet as a delivery mode is also increasing. The nature of the Internet as a non-proprietary delivery system enhances not only the delivery, but also the creation of learning contents, regardless of time and space barriers. Due to the rapid advances in web technology, E-Learning tools have moved from just delivering text-based content to rich media-based content (images, audio, video, animation, interactive multimedia, etc.) [87]. E-Learning systems have also evolved from content delivery to more advanced and sophisticated systems, such as Learning Management Systems (LMS), Content Management Systems (CMS), Virtual Classroom Systems, etc. LMS focuses primary on course administration and registration procedures [9]. LMS has been the most dominant and successful E-Learning system. One could hardly find a university around the world that does not use an LMS.

E-Learning systems have affected several learning platforms: distance learning, traditional learning, and life-long learning. Distance learning has been the main user of E- Learning systems. It has been the most viable solution to providing education to vast numbers of learners in recent years [97]. Distance learning has been available for many years in different formats. It has advanced from low-tech media, characterised by no interaction (radio, tape, textbooks, etc.) through wider broadcasting (TV or radio) with low interactivity (phone, or e-mail) to the current stage, characterised by web pages with online syllabus, chat sessions, forums, etc. In the last twenty years there has been a significant increase in the number of distance learning programs [19]. Traditional learning has also rapidly moved toward integration with E-Learning solutions for more flexibility. Instructors can use E-Learning tools to present their lecture materials on the net, they might use Q/A for posting questions after office hours, and include forums for announcements and discussions.

Life-long learning is based on a current trend in the field of education and training in which learning has become a life-long process [48]. There is a worldwide demand for continuous access to knowledge, assessment and accreditation related to employment. Furthermore, the constant appearance of new knowledge and products makes life-long learning more important. E-Learning provides the appropriate environment for flexible training through which employees can access training programs regardless of time and location [74]. Distance learning, blended learning, and life-long learning will grow even more because of their convenience and flexibility [97].

Even though the potential of E-Learning seems promising, there is still a growing feeling that focusing only on delivering learning content is leading E-Learning to be perceived as learners sitting behind their computers, viewing and turning content pages. This might lead learners to static, passive, low motivation, and isolation [64].

Several online teaching/training programs are reporting high dropout rates. The cause might lie in the current E-Learning settings which might emphasis memorisation and individual responsibilities [104]. Psychological and pedagogical theories agree strongly that viewing traditional lectures that serve only to transmit information from

instructor to learners is not very effective [14]. Less attention has been to improve the learning process in terms of design and community [32]. The business media have pointed out the main challenges in traditional E-Learning platforms. They estimate the learner dropout rate from online E-Learning programs is around 35% [120]. Other studies suggest even higher rates. According to one report, 70 per cent of corporate learners do not complete scheduled online training programs, whereas 42 per cent in distance learning do not complete their courses [96]. This suggests that the access to flexible courses is less important to learners than the quality of learning. One main component that has been missing in these platforms is the social component. The social factors in teaching and learning are still critically important in conducting a successful learning experience. Brogan has suggested in his study that the lack of social interaction is the main factor for the quick loss of interest in online learning programs. Forty per cent of surveyed learners said that they miss the face-to-face interactions with instructors, and 25% said that they miss the group dynamics [19]. In O'Connor's study, he integrated synchronous virtual solutions (chat, video conference) in one of his courses. It caused a higher completion rate (around 90 per cent). He concluded that E-Learning should become social again. Another study carried out by Millbank also showed that the retention rate was raised when mixing contents with real-time interactivity. The retention rate of the trainees was raised from about 20 per cent, using traditional online lessons, to about 75 per cent [91].

These studies indicate that learners appreciate the social component in E-Learning and prefer an interactive medium that allows them to interact and collaborate. Thus, integrating collaborative learning in online programs would successfully increase learners' satisfaction and learning outcomes. Therefore, the focus of E-Learning and E-Training solutions has started to change from passive reception to active knowledge construction through interactions and negotiations [25, 27].

Historically, education has been concerned with learners' social and intellectual development. The recent educational theories are pointing to the importance of human dialogue, interaction, negotiation, and collaboration to produce effective learning

outcomes [14]. For example, social constructivist theory calls for a focus on group collaboration not on objects (books, grades, instructors, etc.), since knowledge that is not constructed and shared tends to be forgotten quickly [19].

There is also an obvious shift in today's society interest from competition to collaboration. In the business literature, authors like Peter Drucker point out that society today is calling for workers who are skilled in problem-solving, collaboration and continuous learning [56] . Collaboration skills are considered to be one of the highly valued attributes of today's organisations environment. It is now common to see employers work together to solve complex problems and share their own knowledge and experiences with each other [6]. As a result, Education must prepare learners with social and intellectual skills for business and professional careers. Isolated learners may fail to develop and refine cognitive and interpersonal skills [14].

2.1.2 Collaborative Learning

Collaborative learning theory is based on social and intellectual interaction. It relies on sharing information, insights, and perspectives, which potentially creates new knowledge [78]. The commonly used, but unsatisfactory, definition of collaborative learning is that "it is a situation in which two or more people learn or attempt to learn something together" [36]. More accurate definition is that "collaborative learning is a methodology that enables learners to work together in a small group in a structured approach to accomplish a common goal" [63, 103].

Collaborative learning was introduced in the early 1980s and has evolved since [69]. Collaborative learning has been popular in schools and universities in the last decade [117]. Informal collaborative activities have grown into structured activities where learners are taking more ownership of their material and developing higher thinking skills [104]. Collaborative learning is based on various issues such as, small-group format, task assignment, role assignments, and internal activity structure. In a small group, everyone can contribute, share ideas, develop interpersonal skills, and learn conflict management more than in a large group [26]. The tasks for learners, working

together, must be clearly defined to achieve the desired outcomes. When groups are guided by clear task descriptions, learners tend to engage more effectively. Furthermore, learners participating in collaborative activities with assigned roles tend to be more task focused than learners without roles [132]. Finally, collaboration sessions should be properly organised and structured to achieve effective outcomes. Collaborative learning is, by its definition, a structured approach that involves a series of steps, requiring learners to create, analyse and apply concepts [103].

In collaborative learning, learners are individually accountable for their work, and, at the same time, the whole group's work. Members are encouraged to feel responsibility toward sharing their knowledge with others to reach the group's common goals. Productivity is influenced by the ability of groups to efficiently divide tasks, exchange findings, and construct common knowledge [26]. Collaborative learning stresses the need to train learners, how to work collaboratively to produce appropriate group outcomes [104].

Recently, E-Collaboration has been a new trend within E-Learning. E-Collaboration is defined as "any kind of group learning that takes place mainly in a virtual environment" [16]. E-Collaboration is becoming more accepted and expected not only in online learning but also in organisational training and meeting [14]. The practice of using small group settings in many training programs creates a natural candidate for online collaborative learning. Managers and employers can meet and share their work experiences with others seated away from them in an online environment [14]. Training and meeting in various geographical locations is costly, and many organisations might face decreasing budget and human resources [121].

Recent research has clearly pointed to various benefits found in collaborative learning, such as positive learning outcomes, positive social skills, higher thinking skills, higher motivation, and higher satisfaction [69]. Many positive learning outcomes can be realised when adopting collaborative learning, such as deeper understanding of content, increased overall grade achievements, active and constructive involvements in content creation, and higher retention rates [81]. Social skills, such as conflict management, oral

communication, teamwork skills, leadership, decision-making, trust-building, etc., are improved when engaging in collaborative learning [69]. For example, in a debate session, participants learn to resolve conflicts, while in a group nomination session (described in Section 2.3.3), they learn how to make decisions. Usually, learners do not acquire these skills naturally. The traditional educational platform focuses more on competition rather than collaboration [116]. The collaborative learning environment helps group members to identify what types of behaviours are necessary for them to work socially, and to realise the importance of healthy, positive, and helping interactions [51]. In collaborative learning, learners tend to develop higher thinking skills [39] such as critical thinking, creative thinking, and elaborative thinking. When learners work in a group, it is common to see one learner discussing the investigated questions while other learners are listening. This helps them to develop valuable problem-solving skills by formulating their ideas, discussing them, receiving immediate feedback and responding to questions and comments. Motivation in group-based learning is high. In the traditional classroom setting, many learners are hesitant to ask or offer opinions, but when learners work in small groups, learners are encouraged to participate more and produce solutions that come from the group rather than individuals [52]. Furthermore, when individuals work together toward a common goal, they tend to encourage each other. Finally, learners who are actively involved in the learning process are much more likely to become interested in learning and make more effort to perform their tasks [42]. Collaborative learning enhances learners' satisfaction with their learning experience. Usually, people find more satisfaction with activities that value their abilities and position them in the centre of the learning process [81].

2.1.3 Modelling Collaborative Learning

The collaborative learning model has the same essential components that are contained within any learning design, such as *objectives*, *context* and *forces*. Figure 1 shows our modelling of the collaborative learning general process. This process is described as an interactive flow. First, the process starts with the course *objectives* specified by the instructor or department. Boyley emphasis on the importance of course objectives. He defines a collaborative activity as a manner of accomplishing teaching *objectives* according to how the activity guides learner interaction with peers and learning resources [18]. Then the educational *context* is used to help instructors select a collaborative activity. For example, in the context of a joint problem-solving activity, negotiation can be viewed as attaining agreement on the assigned tasks (how to represent the problem, what sub-problem to consider, etc.) [38].

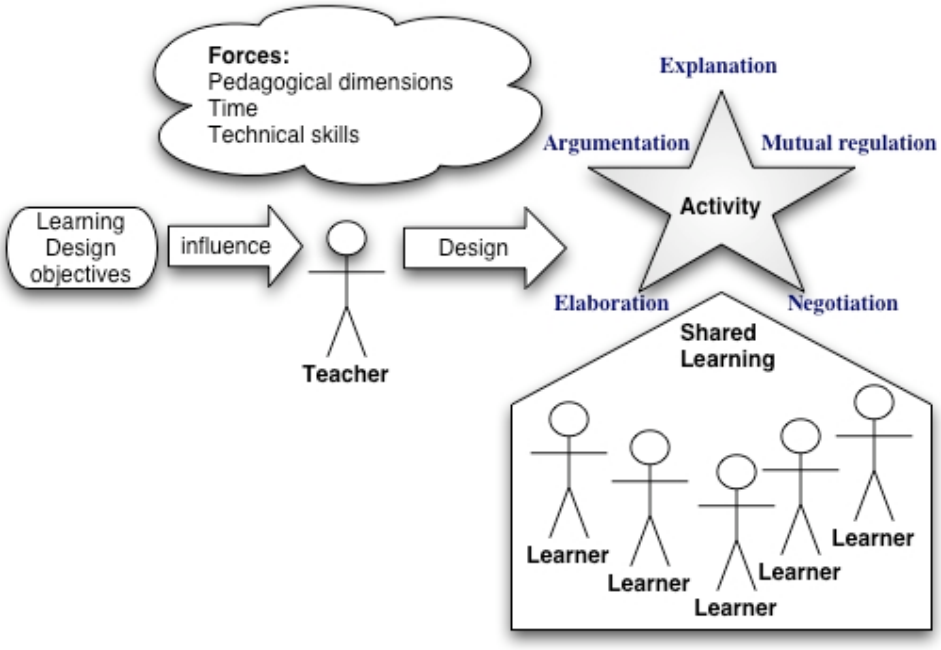


Figure 1: Modelling the collaboration process

The last component that affects the design of collaborative learning activities is the environmental *forces*. The environmental forces according to this model are divided into three types: class size, time and technology. The class size is an important factor in choosing which technique to follow. For example, a debate activity is difficult to implement in a class with 100 learners. The second type of force is time workload. Time workload plays a major role in design considerations, and it is often the most important factor for an instructor designing a learning activity. The instructor needs to know how much time and skill might be required in designing and facilitating the activity, and how much time learners would be required to spend on it. The third type of force is technological, which has two aspects, "tools", particularly which CSCL tools are available and "technical difficulty", which indicates how much time the instructor would need to learn the tool. The design process in this model starts with pedagogical objective not with tool selection [11].

Collaborative learning can be modularised in separate collaboration activities, through which each activity can be considered as a conceptual entity within a course. They could be integrated in a course as intended points of collaboration at specific positions [138].

2.1.4 How E-Collaboration is Changing Instructor and Learner Roles

Current research suggests a deeper look at the role of instructors in the collaborative learning environment. Their roles are evolving from just instructing learners, to many other responsibilities, such as facilitating, information generation, information-sharing, etc. [14]. Instructor roles can be categorised into four main categories: *pedagogical, managerial, technical*, and *social*. The *pedagogical* roles include providing a session's instructions, a session's information, guiding, overviewing, debriefing, etc. The *managerial* roles usually include coordinating, group-forming, and session design. The *technical* role's primary task is to support learners with technological issues, such as software usability, audio/video settings, etc. Finally, the *social* task might include

instructor empathy, interpersonal outreach, such as welcoming, ice-breaking, encouraging, etc. This shift in roles requires instructors to be properly trained to attain wider skills and attitudes [34].

Development in E-Collaboration does not only affect instructors but also learners. Learners need to move away from learner-instructor dependency. Their roles might include searching, reflecting, discussing, negotiating, elaborating, etc. This might demand more effort and responsibilities on how to work together, praising others, participating in turn, sharing decisions, etc. [102].

2.1.5 Challenges of E-Collaboration

After reviewing recent research on collaborative learning, we have identified three main challenges: resistance, control, and resources. In general, instructors and learners are reluctant to change. They are usually satisfied with the way that they are used to, and are unlikely to change their attitudes unless they have the appropriate support on how and when to collaborate [104]. The second challenge is losing control. Instructors are usually unsure of their ability to control collaborative sessions. They are afraid that a session will lose its track if learners do not perform properly [122]. Finally, designing a collaborative activity is considered to be more resource intensive for both facilitator, and learner; facilitators usually resist spending more time in planning alternative pedagogies, designing activities, managing technical skills, etc. Learners feel that the lecture method is easier because they are passive during the class, while in collaborative learning, they need to be more active and intensive [104].

Designing and adopting E-Collaboration is even more challenging than face-to-face. It requires significantly more time and effort from both instructors and learners [14]. In face-to-face settings, the instructor can facilitate more effectively. He can direct, manage, and facilitate the session progress in a direct way. In the online environment there is a virtual distance that usually causes the isolation feeling and the missing of non-verbal language, such as body movement, facial language, etc. The virtual distance might also lead to free-riding phenomena, whereby there are a few learners doing all the

work [122]. E-Collaboration might also require more effort from educational organisations to deal with a number of challenges, such as technical breakdowns, slow bandwidth, firewalls, technical support, etc.

2.1.6 Synchronous E-Collaboration

The way we build and use collaborative learning tools is often a sign of the way we understand what learning and teaching is. The very common approach of taking teaching as a way of transferring knowledge emphasises the use of technologies such as one-way video-conferencing. The socio-constructivist approach puts the emphasis on collaboration, and the intensive use of synchronous and asynchronous collaboration tools. Asynchronous collaboration can be performed at any place at any time [85]. It is usually performed by using asynchronous tools, such as, wiki, forum, e-mail, etc., while synchronous collaboration is usually performed by chat, video-conference, etc.

Asynchronous collaboration has been dominant in recent years. Its main advantage is providing flexibility in participation time. Learners can participate at times that are more convenient, for instance, after school, during the weekends, etc. This allows more time to search, think, and reflect. Nevertheless, there are many challenges to asynchronous collaboration which are listed below:

- Collaborative learning aligns with the social constructivist theory [101], through which social interactions become weaker in the asynchronous model due to the significant time delay between responses and the lack of the group's real-time sense [102].
- Most of the collaborative techniques have originated from face-to-face learning techniques (for example, brainstorming, debate, etc.), which are performed live at the same time, requiring spontaneous responses, and quick turn-around time rather than extended discussion [14].
- Asynchronous learning is used effectively in long-term projects and study cases, but not for short-term activities that are usually embedded inside a single lesson to clarify a certain concept.

- In asynchronous environments, learners tend to form relationships more slowly than in the synchronous environments [132].
- In asynchronous environments, due to delayed responses, participants find it more difficult to follow the session's structure [89].

Face-to-face collaborative learning is still preferable in traditional and blended learning, but the distinction between face-to-face and online collaboration is disappearing. Researchers predict that online collaboration will be dominant in the future [75, 113]. The literature related to synchronous collaboration identifies some advantages over face-to-face:

- The ability to record a session's interactions and the ability to present them in a session transcript. Sessions transcripts are important for assessments. They can help learners and instructors to review the process and support reflection, elaboration, and comments. Transcripts are also valuable for use in research to improve collaboration effectiveness [57].
- The ability to maximise interactions by minimising social pressure. The presence of virtual distance contributes to the democratic feeling of the session. The text-based format reduces cues regarding appearance, race, and gender. Some systems enable instructors to make learners' contributions partially anonymous which means other learners cannot see who contributing, while the instructor can [98]. This takes the focus off the learners and places it on their contributions [88], and at the same time, learners know that they are accountable for their contributions [98]. Many studies have found that learners contribute more in the online environment than in the face-to-face environment [71, 89, 111, 118].
- The ability to enhance quality interactions. Various studies have found that learners' responses were longer, deeper, and more subject-focused in the online environment [57].
- Freedom of space requirements. Synchronous collaboration can happen at any place with an online connection, while face-to-face collaboration sessions

usually require special space arrangements (tables, multi boards, etc.), which are rarely found in the traditional classroom setting.

- Time flexibility. The synchronous environment allows more flexibility in the ability to provide more than one session at different times to avoid schedule conflicts [89].

Even though the recent advancements in web applications and improved bandwidth have led to a greater use of synchronous tools, many challenges plague the effective uptake for informal synchronous collaborative learning [98]. First, interactions may become overwhelming and difficult to manage, especially with chat sessions. Second, coordinating a proper time with dispersed groups is usually difficult. Third, unbalanced participation may occur when participants with the fastest typing skills tend to contribute more [102], or some participants choose to just watch and free-ride while other learners are doing all the work [122]. This presents an additional challenge for designers to structure synchronous collaborative learning sessions in a formal way to obtain the desired learning objectives.

2.2 CSCL Systems

In the last decade, the development of the Computer Supported Collaborative Learning domain has been remarkable [59]. CSCL systems have highlighted the importance of social interactions the in E-Learning environment [39]. CSCL is an interdisciplinary domain (software engineering and education) whose main objective is to allow instructors to become directly involved in designing collaborative activities in the online environment [61].

2.2.1 CSCL Characteristics

Different practitioners in the collaborative learning area (instructional designers, instructional technologists, educational media specialists, educational psychologists, learning theorists, computer scientists, and human-computer interaction) were investigating various commercial products to identify available technological solutions that can facilitate collaborative learning [16]. This field extends the usage of technology as a communication tool for learning, to thinking about how it can also be used to facilitate learners' collaboration, interaction, and knowledge-building. CSCL tools are intended to facilitate various collaborative activities, such as idea-generating, idea-evaluation, decision-making, group dialogue-structuring, interaction-tracking, data-collection, and online mentoring and feedback [14].

In general, CSCL systems should contain certain characteristics, such as community-based, simplicity, customisation, mediation, and reusability. CSCL applications should permit learners to be situated in learning communities that are mediated by rules of participation and by division of labour [51, 130]. Simplicity is a main issue in adopting CSCL applications. Instructors need to be active players in applying technological solutions to their designs. These solutions must be as intuitive and close to their users as possible [33]. Also, instructors should be able to customise technological solutions to their particular needs in every educative scenario [60]. CSCL tools should be designed to provide not only communication but, more importantly,

computer-mediated collaboration [39, 51, 85]. Finally, reusability is the current trend in instructional design. CSCL applications should be able to be configured and tailored by educators to be reused in many different learning scenarios [15].

2.2.2 CSCL Tools

CSCL tools can be divided in two categories: asynchronous tools and synchronous tools. As mentioned earlier, asynchronous tools enable collaboration at "anytime, anyplace", providing more management freedom. These types of tools are appropriate for collaborations that require more time for reflection. Synchronous tools enable "same-time, anyplace" collaboration, providing immediacy and faster response, which enhances spontaneous thinking skills [14].

Although many systems have been developed to support collaborative learning in the online environment, there are common tools used in most of these systems. We start by listing some of the tools found in the asynchronous collaboration environment:

- **Forums**: This is a frequent used tool within the asynchronous environment. It is mainly used to facilitate topical discussions through which instructors can use it to create threads of topics to be discussed by learners.
- **Q&A**: This tool is usually used by learners to post their questions regarding various class issues. The answers can be reused by other learners who have the same questions.
- **E-mail**: It is usually used by facilitators and learners to send and receive notifications and communications.
- **Wiki**: Learners can use a wiki to create a set of documents that reflect shared knowledge within learning group. Wikis can be used to facilitate knowledge-broadcasting and idea-exchanging [8].
- **Weblogs**: The weblogs or blogs were introduced as a place in which learners are encouraged to initiate discussion, comments, notes, news, etc. in order to create and share personal knowledge [141].

- **Asynchronous File-sharing tools**: They allow participants to upload their resources to share them with others.
- **Asynchronous Polling tools:** They enable learners to vote on certain topics or issues.

The main tools found in the synchronous collaboration environment are listed below:

- **Text chat**: This is a text-based communication tool that instructors and learners can use to elaborate, brainstorm, argue, and discuss instantly. There might be various chatting settings to accommodate various learning interaction types, such as private, public, restricted, etc.
- **Audio/video conferencing tools**: These enable instructors to present their lectures in the online environment. They can be also used to facilitate groups' virtual meetings.
- **Application-sharing tools**: Application-sharing tools allow instructors and learners to share their programs and windows. These tools are effective in demonstrating to remote learners how to use certain software applications.
- **Whiteboard tools**: A whiteboard simulates the communication that occurs when instructors draw on the class blackboard.
- **Synchronous File-sharing tools**: They allow participants to upload their resources to share them with others instantly during a session.
- **Synchronous Polling tools:** They enable learners to vote on certain topics or issues live during a session.

Various technical limitations have caused asynchronous tools to be used more in recent years. Nevertheless, synchronous tools are starting to move forward, due to recent advances in both bandwidth and mobile technology. Low bandwidth produces various technical problems that usually limit the use of audio/video conferencing tools. As shown in Figure 2, we have found in an evaluation study (Appendix A) that audio/video quality is highly dependent on bandwidth size. Bandwidth size also has a positive correlation with user adoption of these tools. Users are reluctant to use advanced synchronous tools unless they acquire appropriate bandwidth. The advances and

widespread adoption of mobile systems provide users with more flexibility to access synchronous sessions.

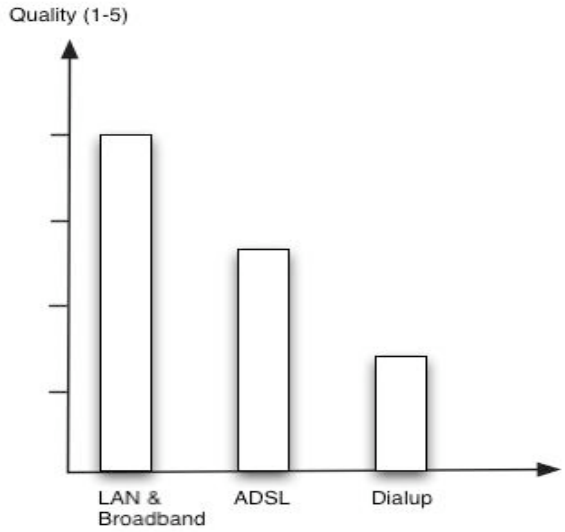


Figure 2: Evaluating audio quality for various bandwidths

2.2.3 CSCL Applications

In the past few years, a large number of applications have focused on providing CSCL [7]. Most of these applications started at universities and/or other research institutions. Their aim has been to solve the primary needs of supporting collaborative learning across different organisations and platforms. In this section we introduce some of these CSCL applications, which represent a wider range of similar applications, such as:

- **Breeze**: This is a commercial software, originally produced by Macromedia and now by Adobe [2], as a rich web communication system that provides online meeting, collaboration, real-time web conferencing, and live presentations. Breeze enables instructors to easily create engaging communications that include voice, video, and animations. Breeze has four main components:
    1- Breeze Meeting: It supports large-scale meetings and small collaborative group meetings. It provides audio/video communication,

application-sharing, and white-boarding.

2- Breeze Training: It provides online training. Instructors can use this product to deploy, track, and manage online courses.

3- Breeze Presenter: It allows designing of media-rich content by including voice annotation and insertion of polls and quizzes into PowerPoint slides.

4- Breeze Event: It manages learners' registration, qualifications, notification, automatic e-mail reminders, and tracking.

- **LAMS** [76]: This is an open-source software, developed by Macquarie University. The intention of LAMS is to enable instructors to design a sequence of learning activities for learners that includes content and collaborative tasks. LAMS provides instructors with a set of collaborative tools that can be easily dragged and dropped to design sequences of collaborative tasks. It allows instructors to runs the sequence of tasks for learners and allows them to monitor and track learner progress.
- **RELOAD** [110]: This is an open-source authoring environment, developed at the University of Bolton. Its main aim is to allow instructors to author learning designs based on activities. A single activity could be any form of learning activity, such as reading a learning material, collaborating with peers, visiting a museum, etc. The RELOAD Editor allows instructors to design a unit of study by sequencing learning activities in a simple format. In each step, instructors need to specify at least the task description. They can also include reading resources or communication service in each step. The RELOAD Player is used to run the design.
- **COLLAGE** [23]: This is a high-level specialised learning design authoring tool for collaborative learning, developed at the University of Valladolid, Spain. The COLLAGE authoring tool enables instructors to easily create potentially effective collaborative learning designs by particularising and

customising some of the best practices in collaborative learning, according to the requirements and conditions of a particular learning scenario.

- **dotLRN IMS LD package:** This is a package that is developed within the E-LANE project [43]. It is the first LMS application that fully supports IMS LD standards (Levels A and B). It is similar to RELOAD in enabling instructors to create and run learning activity designs, and at the same time it benefits from integrating with dotLRN [41]:
  - User management and permissions policies are achieved by the dotLRN platform.
  - The ability to interact with other packages of dotLRN (assessments, learning objects repository, etc.).
  - The ability to be used by communities, classes, groups, etc.

2.2.4 CSCL Developments

Many CSCL applications have been used since the origins of E-Learning. It has been subjected to a long process of evolution. We describe here a number of applications developed around different educational models. In the case of proprietary E-Learning products, it is not possible to determine what their educational models were at the time of design, so we base our description on how they are normally used.

Three main educational models are described and discussed in the following sections: resources-based model, learner-centred model, and activity-centred model. As shown in Figure 3, these models contain the same elements but their mass centre is different. These models cover issues such as resources, human support and logistics, individual activity, and group activity. Any successful learning design should encompass all these issues that might be beyond the model's main focus, but are still considered as an essential part in improving learners' overall learning experience.

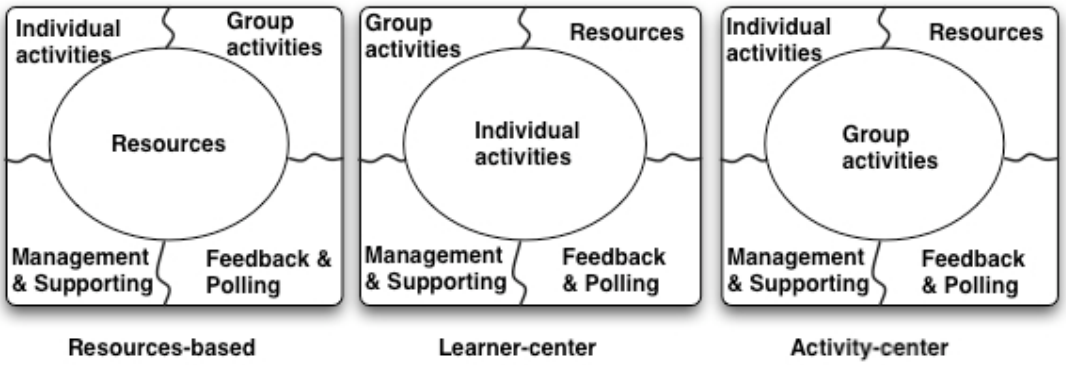


Figure 3: Educational models

Resource-based Model

Many instructors follow a model that puts the focus on teaching materials to be fixed in advance by either an instructor or a content expert [72]. This model is based on instructivist theory, through which learners observe and follow the instructor's instructions and explanations. The resources may include static content such as lecture materials, timetable information, and even multimedia animations. Software products that have been focusing on this model, such as WebCT [134] and Blackboard [12], are the most dominant.

Virtual classrooms (emulation of traditional classrooms) have enabled instructors to stream their real/recorded presentation and resources, through which learners can logon to a website and see/download the presentation remotely. Many of these applications have emerged recently, such as Centra [22], Breeze [84], Webex [135], ConferenceXp [24], etc. These applications are being increasingly used in universities around the world, but nevertheless, they are still based on the instructivist theory, in that they mainly facilitate transferring knowledge from the minds of instructors into the minds of their learners. This approach has been challenged by many researchers [51].

Resource-based applications have tried to practically enhance individual learning by including a communication environment which enables learners to communicate with each other [138]. They might include a forum to enable learners to discuss topics after

class hours or e-mail reminders to be sent to learners. These applications could be classified as learning environments enriched by communication facilities. Learners may benefit by having better access to these resources and communication tools, but collaboration is not a focus of these applications, and communication is seen in many cases as an add-on feature [39].

Learner-centered Model

Current psychological and pedagogical theories agree strongly on viewing teaching that only serves to transmit information to learners as not very effective in the long run [72]. Progress in cognitive and learning science has also developed new forms of supporting learner experiences by focusing on their individual cognitive capabilities and needs. They encourage learners to structure their learning process in a very flexible way [90]. This model assigns a better diverse role to documents used. Learners generally select for themselves the documents they need from a larger choice (which includes the whole Internet). More importantly, they actively participate in the production and annotation of contents, some of which can be generated during the running of the course and can be reused by the same or other learners [72]. This form of learning takes into account individual needs, interests and styles, and prepares learners to become better general problem-solvers. Learners in this model have more flexibility in choosing their subjects, their working strategies, and their own goals. Instructors prepare a certain number of learning paths to select from, and give advice according to their learners' needs [72].

A number of applications have been tested in teaching scenarios, such as PRISE [114], PCeL [29], etc. The supporting educational model for this type of software is based on the constructivist theory.

Activity-centred Model

Learning, which has historically been concerned with social and intellectual development [14], is currently introducing and encouraging activity-based learning [31]. Many researchers are predicting that learning will be seen as more socially shared, active, and interactive than in the past [14]. This recent interest has originated from

various socio-constructivist schools where its goal is creating deeper, better integrated, and applicable knowledge. In this model learners are expected to collaborate, which makes learning more interesting and more fun [113], and effective learning is understood to happen when learners interact with their learning environment [74]. The learning environment may contain a collection of learning objects and services. Learning objects are considered to be any reproducible and addressable digital or non-digital resource. Service facilities are used during the teaching and learning to facilitate group activities, for instance, a discussion forum or some other communication facility. Very few applications have been developed based on this model, whereby they are originally composed of a series of tools, organised in a sequence that breaks down the whole activity to mini-stages, in which each stage is performed using one tool. Several learning design editors are available that enable designers to produce activity sequences. LAMS, for example, allows instructors to select, arrange, and sequence activity tools such as chat, Q/A, forums, etc., in a very user-friendly manner.

2.2.5 CSCL Challenges

Researchers in this field have demonstrated the complexity inherent in the design of CSCL. Conceptual tools are needed to refine and combine research findings and current practices in order to make them applicable for both CSCL developers and instructors [10]. In the last decade, CSCL developers exerted significant efforts to make genuine collaborative learning a part of E-Learning courses; yet there is little evidence in adopting these collaborative learning styles. Many challenges that arise are actually not trivial. Designing collaborative learning activities is complex and requires a lot of training. Users of CSCL applications are not usually technical experts [10]. However, they have to properly select, configure, and arrange these applications to support their particular learning situations [130]. Moreover, since CSCL applications are developed by technologists, they need collaborative learning domain knowledge from a technological viewpoint. In addition, the requirements posed by educators are highly dynamic [32].

Four main challenges are found in implementing CSCL applications:

- **Lack of experience**: Regardless of the enthusiasm for using CSCL, instructors still face various challenges regarding their pedagogical and technical experience. In general, instructors do not have enough background in the various types of successful face-to-face collaborative techniques found in education literacy, and even if they have this, they do not know how to implement them in the online environment. Instructors lack appropriate experience in how to design collaborative activities that can benefit their teaching objectives. They have to think carefully about the activity structures and instructional events that they want to include [14]. Finally, they have a further lack of experience in how to use, select, and embed CSCL tools in their courses. This uncertain feeling stands as a significant obstacle in adopting CSCL. Thus a new requirement is posed to the CSCL developers: how to provide technological solutions for collaborative learning capable of being adopted by instructors lacking various types of experience [60].
- **Lack of technical skills**: The implementing of CSCL designs is tricky. Instructors not only need to evaluate tools but also to select, configure, and support. Usually instructors have little time and interest in learning new technologies that are constantly emerging at a high rate. There is a constant need to test pedagogical inventions that can foster online collaboration [14]. This methodology requires much more detailed preparation than traditional teaching [85]. Therefore, collaborative technology should be very "user-friendly" and even instructors who do not have much experience with computers can easily learn to manage them [85]. These systems should be simple enough to place the control of design with instructors, rather than software designers [77]. However, such tools are crude or nonexistent in most higher education and training courseware [14].
- **Facilitating limitation**: It is not sufficient to provide learners with collaboration tools. Facilitators should monitor learners' process in order to guide participants

during the collaborative session. However, it is difficult for a facilitator to support a collaborative session when many teams have to be monitored at the same time [6]. CSCL systems should be designed to help instructors in facilitating various groups.

- **Resource limitation**: Designing and implementing collaborative learning is time and resource demanding. An instructor first starts by careful designing of a collaborative session that is appropriate for his pedagogical objectives. Then he needs to evaluate the best technological solution for his design. Finally he needs to manage, support, and facilitate the designed session [113]. In a study conducted by Palloff, she found that instructors in the online environment need to spend twice or three times more to deliver a course than in a face-to-face situation [102]. Even though collaborative learning requires significantly more time and effort, it can generate new knowledge, attitudes, and behaviours [14].

2.3 Design Patterns

2.3.1 Patterns

The interest in capturing and recording successful designs that can be reused by less-experienced practitioners has surged in E-Learning [86]. Christopher Alexander was a pioneer employing patterns to communicate good practices in the construction of towns and buildings, using architectural design and arrangement techniques [4]. His theory states that each design problem is the result of certain forces in a specific context and patterns should describe a way to resolve these forces. He stated that "each pattern describes 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 without ever doing it the same way twice" [34]. A pattern seeks to organise information regarding a contextualised common problem and its broadly accepted solution couple in a specific context, and presents them in a design that can be reused by non-expert practitioners within similar settings [10]. A pattern does not provide a complete solution but rather provides enough guidance to allow users customisation and intervention in each reuse [86].

There are two types of patterns: design patterns and construction patterns. Alexander's design patterns refer to understanding the geometry of a building and the relationships between its parts, while construction patterns examine the materials and processes needed in order to put the designs into practice. Pattern language tries to puts all patterns together. It is defined by Alexander as "a collection of interconnected (related) patterns that enables the generation of a coherent whole (for example, a town)".

Despite the fact that the word "pattern" has been widely used in different disciplines, its use is better known in the fields of architecture and software engineering [50]. Recently, other domains (for example, E-Learning) [129] are moving toward using patterns. E-Learning patterns are different from software patterns. E-Learning patterns are closer to Alexander's notion of design patterns than to the way they have been used

in the field of software [10]. E-Learning patterns take into account both material and social issues. E-Learning patterns in the CSCL field are more complex. Their primary objective is to conceptually bridge the collaborative learning patterns and software patterns [7] and to enhance teachers'/designers' perception in the meaning of design reuse.

2.3.2 Instructional Design Patterns

Designing successful courses, lectures, and training programs is a challenging task. Instructional design is a systematic approach to course development which ensures that specific learning objectives are accomplished [115]. It focuses on structuring learning process in an appropriate way within certain pedagogy that makes learning more efficient. For example, reusing the same courseware content of a highly prestigious university (for example, MIT open courseware) does not guarantee achieving the same results. The instructional design field has evolved from just a simple course structure that is usually presented in a course page with some links to course materials (Power Point slides, reading materials, animation, etc.) to more complex designs that allow sequencing, testing, activities, multiple paths, etc. The emphasis on the learning process has led to a wider investigation by E-Learning researchers and practitioners on how the pattern approach can encourage reusing successful learning designs [31]. The current design pattern approach in the E-Learning environment is employing patterns that describe and capture courses' and course modules' abstract structure to allow their reusability in other learning designs [32]. Various projects have been developed to support E-Learning patterns, such as E-LEN [44], Pedagogical Patterns Project [107], etc.

2.3.3 Collaborative Learning Patterns

The pattern approach currently draws more attention for identifying collaborative patterns within traditional course settings [29]. Collaborative techniques that are usually conducted in face-to-face learning are an ideal candidate for these patterns.

A literature review on the various collaborative techniques that have been used in classrooms shows that there are more than 100 techniques [49, 103, 106]. We have chosen to present and describe thirteen techniques that are popular within the teaching community and at the same time can be applied in the online environment. They are also chosen to represent a wider range of techniques that tend to develop various types of thinking skills. We will refer to these techniques throughout this thesis.

These techniques are: **Informal Group Discussion**, **Round Table Discussion, Brainstorming**, **Group Nomination**, **Debate**, **Jigsaw**, **Pro/Contra**, **Think Pair Share**, **Pyramid**, **Buzz Group**, **Role Play**, **Case Study**, and **Team Pair Solo**. These techniques encourage the development of various types of thinking skills, such as creative thinking, critical thinking, analytical thinking, reflective thinking, etc. For instance, Debate techniques encourage the development of critical thinking skills, while Brainstorming techniques encourage creative thinking skills. Nevertheless, there is no agreement on common definitions or descriptions of these techniques. The same technique can have different formats. For instance, the Debate technique could be conducted in formal or informal formats. Furthermore, the formal format may have various types (for example, two-person format, three-person format, Oxford format, Cambridge format, etc.).

The collaborative learning techniques are categorised in this section according to which thinking skills they develop. Nevertheless, some techniques can develop more than one type of skill, and furthermore, these skills sometimes do not rely solely on the process, but also on the contexts and contents. We start by listing the three techniques (**Debate**, **Pro/Contra**, and **Role Play**) that basically promote the development of critical thinking skills [14]:

- **Debate** technique: This technique is ideal for clarifying a controversial issue by presenting the best argument for or against the proposal. This interaction strategy is used to help learners developing their critical thinking and listening skills.
  Procedure:
  1. The proposer proposes the motion.
  2. The second proposer seconds the proposer.

3. The opposer opposes the motion.
4. The second opposer seconds the opposer.
5. Speakers participate from the floor.
6. The proposer sums up for the motion.
7. The opposer sums up for the opposition.
8. All participants engage in voting on a particular side.

- **Pro/Contra** technique: This is usually used to encourage learners to present different sides' views.

  Procedure:

  1. The instructor provides a specific period of time to participants to study a particular side of a controversial issue.
  2. Each side starts its own discussion.
  3. When the discussion is completed, both sides join and start a common discussion.

- **Role-play** technique: This is usually used to train or demonstrate different aspects of a topic.

  Procedure:

  1. The facilitator provides guidelines for each role.
  2. Each participant with a role starts acting.
  3. At the end, the facilitator and participants debrief the situation and various perspectives of the individual roles.

Second, we present the two techniques (**Brainstorming** and **Group Nomination**) that promote the development of creative thinking skills [14]:

- **Brainstorming** technique: This technique is usually used to generate many ideas or solutions in a short time. It motivates groups to generate many creative new ideas or solutions by not allowing criticism or elaboration.

  Procedure:

  1. The facilitator specifies the problem.
  2. Participants start posting their solutions.

- **Group Nomination** technique: This technique is an evolved version of the Brainstorming technique. It is ideal for decision-making for a certain topic or problem without a specific solution whose resolution implies not only creating ideas or solutions, but also choosing the best idea.

  Procedure:

  1. Posting ideas, no criticism or elaboration is allowed in this step.
  2. Discussing posted ideas to obtain clarification and evaluation.
  3. Idea-prioritising, each participant is asked to assign a mark for each idea.
  4. Idea-reporting, reporting the highest idea to other groups (done by chairperson).

Third, we present the four techniques (**Jigsaw**, **Think-Pair-Share**, **Pyramid,** and **Buzz Group**) that encourage the development of analytical and critical thinking skills:

- **Jigsaw** technique: This is usually used when a small group faces the study of a large amount of information regarding a complex problem that needs resolution. It requires handling information that can be divided and used for the resolution of independent sub-problems.

  Procedure:

  1. The instructor allocates a specific period of time for each participant in the group (jigsaw group) to study a particular sub-problem.
  2. The participants of different groups that study the same problem meet in expert groups for exchanging ideas.
  3. The jigsaw group participants meet to solve the whole problem.

- **Think-Pair-Share** technique: This is usually used with content that requires individual reflection, peer discussion and group solution-sharing.

Procedure:

1. Each participant thinks silently about the question posed by the instructors.
2. Individuals pair up during the second step and exchange ideas.
3. Each pair shares its responses with the entire class.

- **Pyramid** technique: This is usually used with content that requires individual reflection and then multiple levels of group discussion and explanation.

  Procedure:

  1. Each participant thinks silently about the question posed.
  2. Individuals exchange thoughts with their small group.
  3. They share their responses with larger groups.

- **Buzz Group** technique: This is usually used to gather potential solutions for a problem using short-time group discussion.

  Procedure:

  1. The instructor posts a problem to the class.
  2. Participants exchange thoughts on how to solve a problem within their group.
  3. Each group submits its resolution to the entire class.

Finally, the rest of the techniques (**Informal Group Discussion**, **Round Table Discussion**, **Case-study**, and **Team-Pair-Solo**) are highly dependent on their context:

- **Informal Group Discussion** technique: This technique establishes and encourages group knowledge-sharing among learners through discussions.

  The procedure is quite simple:

  1. The facilitator specifies the discussion topic.
  2. All learners start the discussion according to their experience, and build upon others' contributions.

- **Round Table Discussion** technique: This technique encourages balanced participation. It motivates the generation of new abstract understanding by enabling balanced knowledge-sharing.

  Procedure:

  1. The facilitator specifies the discussion topic.
  2. The first learner at the table starts talking.
  3. When the first learner finishes, the next learner starts.
  4. This process continues until all learners finish.

- **Case-study** technique: This is usually used to help participants to develop skills in identifying concerns, analysing problems, and considering solutions.

  Procedure:

  1. The facilitator provides a story or situation material to learners.
  2. Each participant reads or watches the related material.
  3. Each participant submits his own resolution of proposed situation.

- **Team-Pair-Solo** technique: This is used with a content that requires group discussion, peer explanation, and finally individual reflection.

  Production:

  1. The facilitator posts a discussion issue.
  2. The entire session discusses that issue.
  3. Individuals pair up and exchange thoughts.
  4. Each participant reflects on his resolution for that issue.

All these techniques can be presented in a pattern format. The Alexander patterns [4] have this structure:

i) A picture (showing an archetypal example of the pattern)

ii) An introductory paragraph setting the **context** for the patterns.

iii) **Problem headline**, to give the essence of the problem in one or two sentences.

iv) The body of the problem (its empirical background, evidence for its validity, **examples** of different ways the pattern can be manifested).

v) The **solution**, stated as an instruction, so that you know what to do to build the pattern.

vi) A paragraph linking the pattern to the smaller patterns which are needed to complete and embellish it.

We have adopted the Goodyear's notation [54], which varies from Alexander's notation but still contains the fundamental principles, to represent collaborative learning techniques. For example, the Jigsaw pattern is presented in Table 1.

Name	Jigsaw
Problem	A complex problem whose resolution requires handling information that can be divided into disjoint sets and used for the resolution of independent sub-problems.
Context	Several small groups facing the study of a large amount of information for the resolution of the same problem.
Session's Agenda (flow of session's step)	1- Instructor gives a specific period of time for each participant in the group (jigsaw group) to study a particular sub-problem. 2- The participants in different groups that study the same problem meet in expert groups for exchanging ideas. 3- Finally, jigsaw group participants meet to solve the whole problem.

Table 1: Jigsaw pattern

2.3.4 Implementation of Pedagogical Models

After reviewing various approaches and systems that can be used to facilitate collaborative learning patterns, we have realised that there are three basic approaches that can be modelled in a 2-layer framework. These approaches define how collaborative learning patterns can be implemented by using CSCL tools and components. The top layer represents the collaborative techniques, while the lower represents CSCL components. The following sections describe these three models (general-based, specialised-based, and component-based). We describe the models by the way the tools implement the patterns and the number of patterns implemented by a tool, and the number of tools required to implement a pattern.

2.3.4.1 General-based Model (m-1)

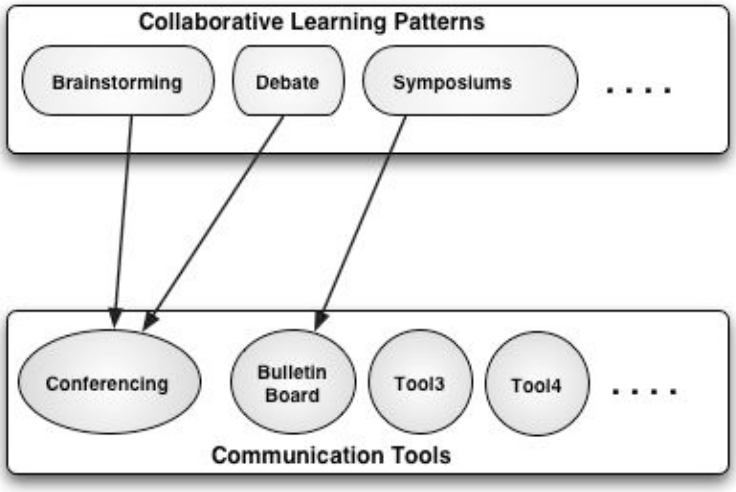


Figure 4: The m-1 model

This model mainly relates more than one pattern (**m** patterns) to the same communication tool, as presented in Figure 4. The first layer is the most general layer, as it describes the different collaborative learning patterns. Paulsen [106] was a pioneer in defining a framework for pedagogical CMC (Computer Mediated Communication) models. His contribution was to divide the existing technique into four groups according to four communication tools categorised as illustrated in Figure 5. The first category contains all techniques that can be conducted via one-alone tools, such as retrieving information from online resources without communication with the instructor or other learners. The second category relates to the one-to-one tools (e-mail applications). The third category relates the one-to-many tools (bulletin boards). The final is the many-to-many tools (computer-conferencing systems or bulletin-board systems).

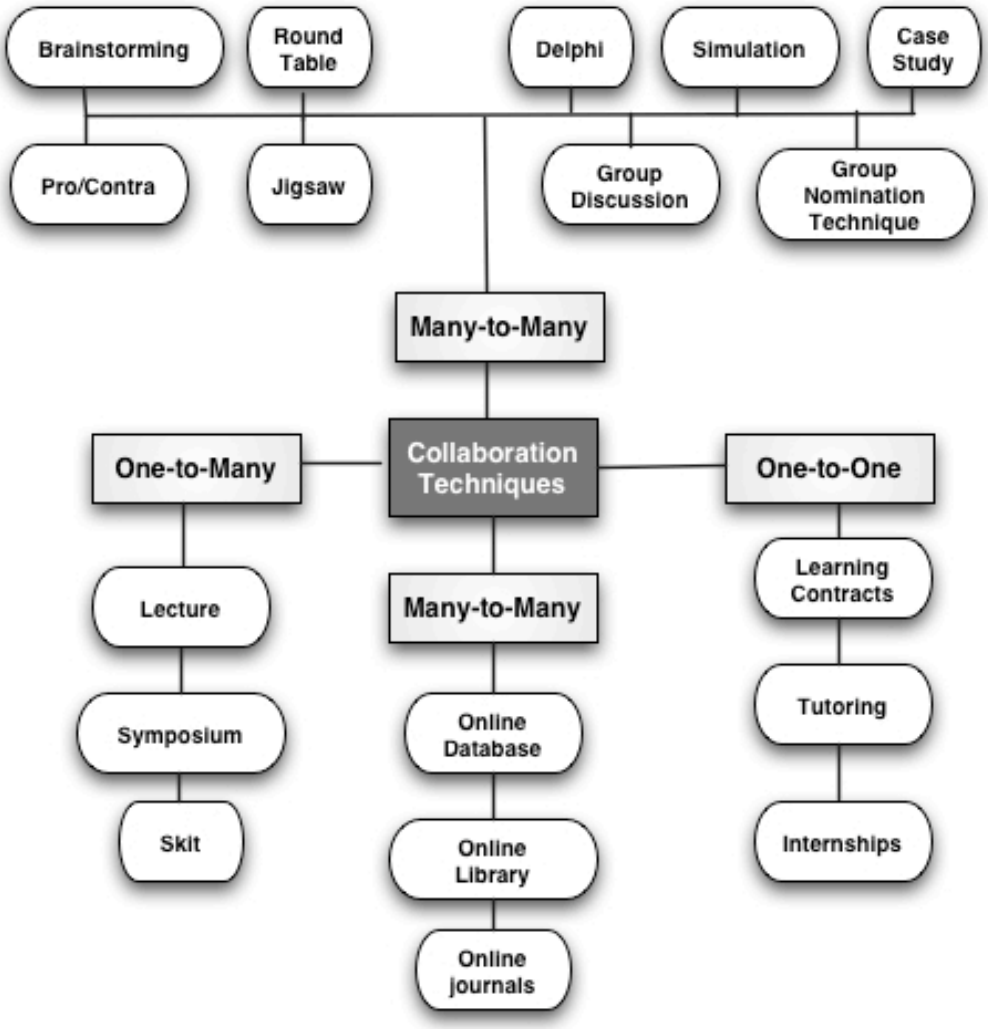


Figure 5: Collaboration technique categories

This approach has not been adopted widely, due to technical difficulties in applying different techniques using the same tool. Defining various collaborative learning patterns around fixed communication tools makes them hard to adopt or use in other contexts. Those generic tools are also not always appropriate or sufficient to build a meaningful learning experience online. In fact, for most collaborative patterns, these individual tools are insufficient. For example, an instructor, who wants his learners to participate in a brainstorming activity, might be forced to restrict his selection to a bulletin board and a text chat, but an online brainstorming activity would probably

require more functionality in order to be effective. It might also require, for example, an idea chart to keep track of the ideas that arise during the session, a stopwatch tool to keep track of the time, a chat area for discussing the ideas as they arise, a voting tool for deciding on the best idea, and a tool that enables a group chairperson to post the winning idea to other groups. Moreover, there are instances when only certain participants should be able to access certain tools at certain times. Clearly, it is common for more than one simple tool to be required and integrated, in order to carry out one type of collaborative pattern effectively [15].

2.3.4.2 Specialised-based Model (1-1)

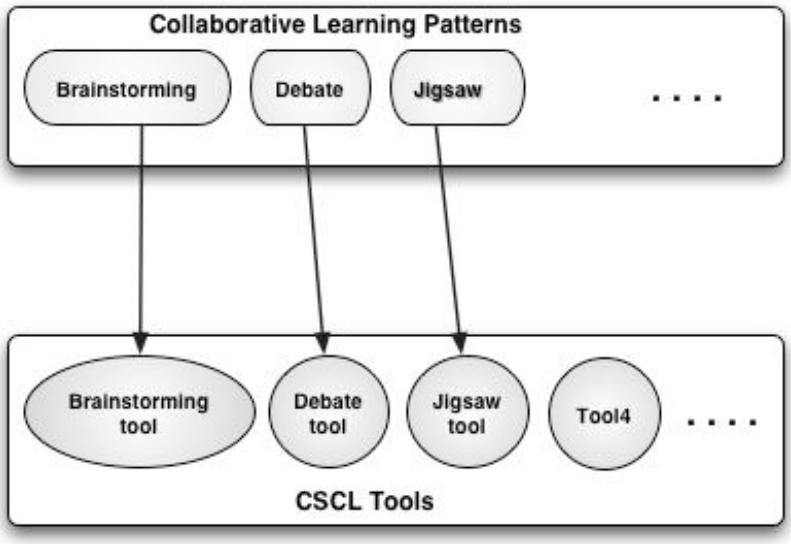


Figure 6: The 1-1 model

The second approach is building software that supports collaborative learning which involves providing instructors with a specialised tool for each collaborative learning pattern as presented in Figure 6.

In current E-Learning systems, there have been many attempts to develop specific tools for specific pedagogical techniques. The L3 [138] project has developed a series of PoCs (Point of Collaboration) tools (such as the Brainstorming tool, Pro/Contra

tool, and Group Discussion tool). Other projects have also developed specific pedagogical tools, such as the Brainstorming Toolbox [82], FACILITATE [46], and Fablusi [45]. This approach to build specific pedagogical tools is difficult to implement and reuse, due to four main reasons:

1- The large number of possible collaborative techniques (over 100 techniques [11]).
2- The same collaborative technique might be performed in different ways by different instructors.
3- Instructors should be able to design new and customised collaborative learning activities and techniques.
4- The low reusability, whereby tools can hardly be used in other techniques [40].

2.3.4.3 Component-based Model (1-m)

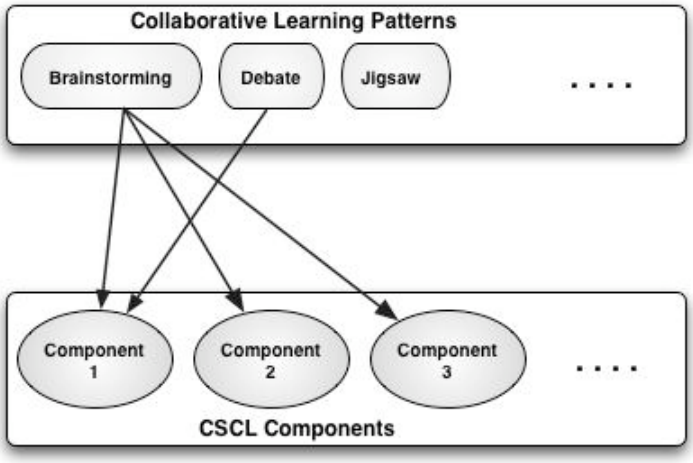


Figure 7: The 1-m model

To encourage maximum reusability, patterns have been introduced in both levels: in the collaborative learning level and in the software components level, as shown in Figure 7. During recent years, the technologies associated to the Component-Based Software Engineering have emerged as a promising solution for the achievement of software reusability and flexibility, in the development of complex software systems [7]. This approach helps software developers to understand the requirements of specific types of

CSCL and make it easer to identify common software components. There have been two directions in this approach. The first one is to provide instructors with a set of components to select and sequence to support their collaborative activity designs (for example, LAMS). The challenge to this approach is the complexity of collaborative learning designs. The design should support the interactions between learners, their instructors, the tasks they undertake, and the result they deliver according to pedagogical structure [51]. In the second direction, instructors start by selecting a collaborative technique, not a tool. Systems that follow this approach assemble tools from smaller components to support a specific technique [79] (for example, COLLAGE [61]). Nevertheless, building a component-based framework is challenging. The software developer must face a main challenge related to the particularities of pedagogical techniques in order to build reusable CSCL components [47].

One specific approach in this direction was suggested by Schneider [113] that is similar to Beehive's approach, in which task patterns can compose various collaborative patterns. He stated that various collaborative learning patterns could be formed from smaller mini-activity patterns, where specific CSCL components could be identified to support these mini-activities. He identified the following six mini-activities:

- **Do** (production)
- **Deposit** (sharing)
- **Read** (consumption)
- **Look** (discovery)
- **Discuss** (interaction)
- **Feedback** (report of results)

These activity patterns cover the interaction between learners, their instructors, and learning environment (learning resources and tools). They also trigger certain learning mechanisms, for instance, the **Read** triggers some learning mechanisms (induction, deduction, compilation, etc.), while the **Discuss** triggers other learning mechanisms (explanation, disagreement, mutual regulation) [36].

For example, in the Jigsaw pattern: each group member starts by accessing a specific subset of the information necessary to solve the sub-problem. Therefore, the activity pattern in this step is number 3 (**Read**). Then participants of different groups that are studying the same sub-problem meet in expert groups for exchanging ideas. This step is related to activity pattern number 5 (**Discuss**). The last step, in which jigsaw group participants meet to share their findings to solve the whole problem, is related to activity pattern number 2 (**Share**). So for this technique the activity pattern flow is: **Read-Discuss-Share**. Other techniques may contain some/same patterns but in the reverse order. For example, the Team-Pair-Solo patterns flow is: **Share-Discuss-Feedback**.

Examples of CSCL components that can support these activity patterns are presented in Figure 8. For the Jigsaw technique, three suggested components might be used respectively: Information Viewer, Conferencing, and Forum.

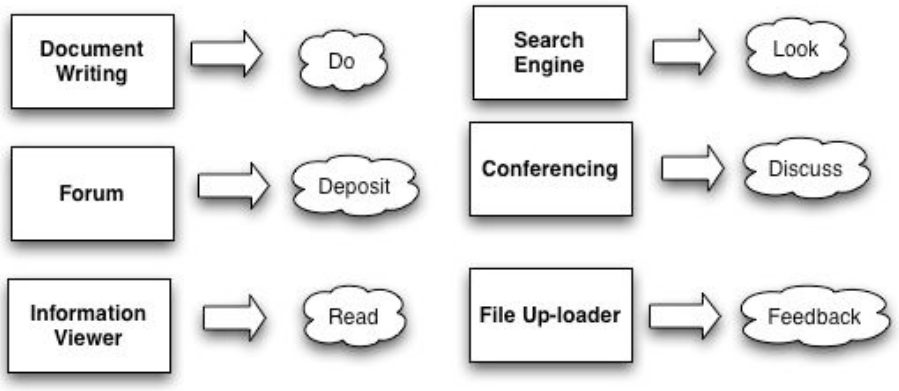


Figure 8: Sample CSCL components for the mini-activity patterns

2.4 Application Framework

The concept of application frameworks was introduced to benefit from reusing software's components and design [47]. A common definition of an application framework " framework is a reusable design of all or part of a system" [70]. The application frameworks consist not only of software components but also of design patterns. Components reusability idea calls for:

- Components that can be easily reused by a developer who does not have to know how the component is implemented, but only needs to learn how to use it.
- Components that can be easily connected to make new systems, where these systems are customizable and efficient.

Frameworks also provide a form of design reuse. They usually present schemas that represent the system's high-level design. Frameworks are similar to specific-domain open architecture in supporting design reuse [70].

The primary benefits of application frameworks are modularity, reusability, and extensibility, as described below:

- Framework modularity helps improving software quality by localizing the impact of design and implementation changes to reduce the effort required to understand and maintain the software.
- Frameworks enhance reusability by defining generic components that can be reapplied to create new applications. Also, the framework reusability aims to avoid re-creating common design solutions to recurring application requirements and challenges.
- Framework extensibility is one of the most important attributes of a framework. Extensibility is essential to ensure scalability, in which new application services and features can be added easily.

2.5 Collaborative Script

Structuring the collaborative learning session is necessary to achieve proper learning outcomes, since informal collaboration does not necessarily produce effective learning [37]. Providing learners with communication tools included at the course home page and expecting them to collaborate by using these tools might lead to many disappointments. Collaboration should be built around specific collaborative mini-activities that are arranged and integrated properly in the course design to encourage learner participation. The learners should understand clearly what they are expected to do, who will do what, and what the objectives are behind doing it.

Even though in some cases informal collaboration sessions may produce many interactions, it might be misleading for researchers to just rely on counting interactions as the only way of evaluation. Many of the interactions may not relate to the activity tasks, or they might be general reactive statements referring to just a single message [109]. A way to deliberately promote productive interactions is by the use of collaboration scripts [129]. The collaboration script is defined as a way of describing a mini-activity flow in a single collaborative learning situation in a formal way [131]. A session script should also describe roles and their association with these mini-activities [75].

2.5.1 Modelling the Scripting Process

The design of collaboration scripts is a new focus of research within the CSCL environments [90]. Each face-to-face collaborative technique contains a scenario. These scenarios are the essential part of the solution section inside pedagogical patterns [31]. The collaboration scripts are intended to capture the pedagogical patterns' solutions that can be used repetitively by many instructors [61]. The script usually defines different phases (mini-activities), where each phase contains at least an elementary activity, which in turn can be supported by a specific tool in the run-time environment [113], as shown in Figure 9.

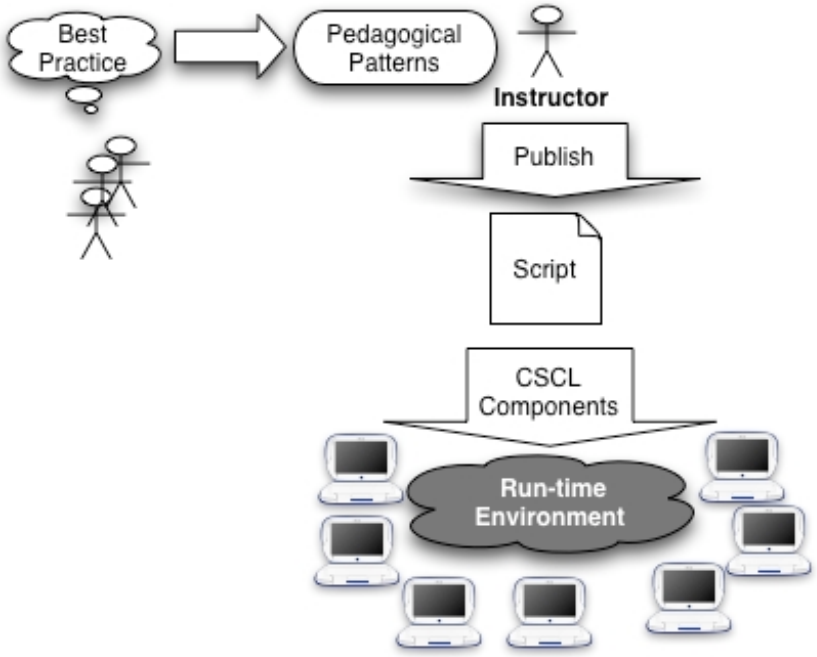


Figure 9: Modelling the scripting process

The run-time environment should contain all the resources and tools needed to carry out the activities [75]. It should interpret the script to initiate the proper CSCL components automatically. CSCL components in turn must support learners by directing them to interact according to a collaboration script.

2.5.2 Scripting Benefits

In general, many learning benefits can be accomplished when instructors structure their collaboration sessions before their learners engage in these sessions, such as:

- **Positive pedagogical outcomes**: Scripts encourage instructors to think deeply about the pedagogical objectives of a session and its pedagogical values [37].
- **Effectiveness**: Activity effectiveness is not guaranteed if the instructor simply asks learners to participate in informal group activities to accomplish certain tasks. There is usually the risk that learners cannot start, get lost, or cannot reach common goals [113]. In synchronous collaboration, it is even easier for

participants to become confused about what exactly they need to do, due to the time limitation [102].

- **Tasks focus**: The problem of controlling collaborative sessions refers to the facilitator's ability to keep learners focusing on the tasks [13]. What usually happens in informal sessions is that discussion over time extends to some unrelated issue that someone has mentioned. This tends to deviate learners from the original intended discussion topic [111]. For example, Olanirain realised that when a facilitator provides learners with directions in the Debate session, learners are likely to be more focused on the debating topic and they use more justification for their positions, rather than just trading their opinions [98].
- **Facilitating**: Reading the collaborative script makes it easier for the facilitator to guide knowledge exploration and communication among learning participants [17]. It is even easier to facilitate if learners themselves understand the script and their part in that script before they engage in the session.
- **Reusability**: Scripts can be reused in the same or different learning context to save time and effort.
- **Bridging**: Scripts can help in bridging the gab between CSCL developers and educational practitioners. Collaborative script notation, written by instructors, can help CSCL developers to identify and develop appropriate CSCL tools.

There are even more benefits to using automated scripts that can be interpreted by CSCL systems, such as tools selection, decreasing cognitive load, and systematic facilitation. For the tools selection, automated scripts can be used by the system to select and configure the appropriate CSCL tools needed in the collaboration session [108], minimising the effort needed by the instructor (hiding implementation complexity) [28]. Regarding decreasing cognitive load, scripts may interfere with the main learning process. The load is increased by the necessity to understand, memorise and execute the script [37]. Therefore, systems can use the automated script to display messages to direct learners in performing their tasks, for instance, "start posting your ideas", "it is time now for you to vote", etc. Finally, automated scripts can be used for systematic facilitation. In

the case of a large number of groups, it becomes more difficult for a single facilitator (for example, instructor) to guide the collaboration process. Automated scripts allow CSCL systems to mediate the communication and collaboration process for all groups at the same time in an automatic way.

2.5.3 Attributes of Scripting Language

The intention of a scripting language is to formalise a collaboration script to specify exactly how learners, within their groups, play certain roles in collaborating towards certain outcomes within environments that contain related tools and content [90]. Some projects have proposed scripting languages to facilitate the modeling of collaborative learning processes. Their aims have been mainly to enable instructors to use a scripting language to publish CSCL scripts that are suitable for being computationally represented and interpreted [90, 129]. Dillenbourg, a pioneer in collaboration scripts, pointed to the similarity between scripts and languages. Similar to how languages are made, words and grammatical rules that construct an infinite number of sentences, collaboration scripts are based on a limited number of components but still can form a large number of scripts [86]. Most collaboration scripts are presented in a sequential format whereby learners are engaged in a series of activity phases:

Script = [phase1 phase2 phase 3 ...]

Each phase of the script specifies components that describe exactly how learners should collaborate within that phase:

Phase = [Component1 Component2 Component3 …]

These components usually specify the tool, time, role, etc. [37]. The audience of the scripting language consists of instructors and learning designers [129].

In order to successfully apply the scripting language, it should maintain several characteristics, such as formalisation, linearity, timing, and role based. The scripting language should first be formalised to enable script players to interpret and manage the flow of mini-activities needing to be performed during a collaboration session [131].

Linearity is another important aspect. Looking at the various scenarios found in

face-to-face collaborative techniques, it is obvious that they follow linear structures [90]. For example, in the Group Nomination Technique, learners start posting ideas and when they have finished, they start discussing posted ideas, and so on. Also in synchronous sessions, groups or individuals need to synchronise their progress. They might need to synchronise separation and joining back at certain phases in a linear process. Finally, it would be very difficult for instructors to design non-linear structures by using conditions and notifications. This type of specification might also be very complicated and confusing learners to follow [37].

The third characteristic is time. The time component should be explicitly specified in the language. The activation and the process of a script are considered to be mostly automatic when usually transitions from state to state are triggered by specific timeframes.

Finally with roles, activity phases tend to be more specific [137]. Specifying roles for a certain activity allows systems to control who has the right to speak or perform during that phase.

2.5.4 Scripting Language Challenges

The main challenge is how to design a scripting language that is easy enough to be used by instructors and in the same time can reach the complexity of online collaborative learning designs. Modelling various collaborative scenarios and creating technical specification is complex and challenging [61]. Three main obstacles are involved: identification, representation, and adoption.

Identifying a collaboration script language and its computational representation is not trivial. As in each language, rules should be defined and followed to create proper outcomes and it should be kept as simple as possible to enable smooth adoption.

The scripting language needs to describe not only the appropriate CSCL tools needed in the activities, but also the resources, organisational forms, roles, timing, etc. [129]. There are various types of resources that can be associated with collaboration. The scripting language needs to describe these resources in a formal way. Input

resources can represent the resources that need to be conducted before the activity, such as reading materials, a list of questions, etc., while output resources represent the outcome of that activity, such as reports, ideas, etc. Also, the language should specify the different types of organisational forms found in collaborative learning settings, such as peer-to-peer, small groups, large groups, etc. There might also be different roles that have different access rights (for example, only the chairperson can submit the group report).

The representation of scripts can be accomplished by using XML, due to its system independence and interpretability by machines and partially by humans. The problem is that educators are not usually familiar with reading XML documents. Therefore, authoring tools are recommended to facilitate the elaboration of collaboration script implementation.

From the usability perspective, adopting effective CSCL scripts is challenging [90]. The scripting language should be kept as simple as possible to enable quick adoption by users with basic technical backgrounds (for example, instructors) [37]. It should also be as similar as possible to the face-to-face scenario representation and abstractions to enable instructors to relate more to their experiences [11]. Finally instructors should be aware of over-scripting, whereby they may tend to control even the smallest details in the session, restricting learners' creativity and moving toward the instructor-centred model [35].

2.6 Standards

In general, the purpose of E-Learning standards is to provide interoperability and reusability of data, data structures and communication protocols between organisations [80]. E-Learning standards offer a common language for sharing resources and adopting them without restricting the customisation [64]. The process for creation of standards began when cooperating organisations worked together to develop initial specifications that they hoped to be used by a large community. Some of the main contributors to interoperability standards are:

- IEEE's Learning Technology Standardisation Committee (LTSC) [83].
- IMS Global Learning Consortium [65].
- The Aviation Industry CBT Committee (AICC) [3].
- The USA Department of Defense's Advanced Distributed Learning (ADL) [1].

Many interoperability specifications are included in the standards, such as the learning object meta-data specification, content packaging, learning design specifications, etc. The learning object meta-data specification aims to provide a definition of how various learning resources are described and tagged. The content packaging specification addresses the description, structure and location of online resources. The learning design specification supports the use of a wide range of pedagogies in E-Learning.

2.6.1 Learning Objects

A learning object is usually considered to be the lowest level of granularity of knowledge, which can represent the smallest indivisible element in a course [5]. Learning objects can take any form, digitally or non-digitally, such as text, image, audio, etc. Several proposals have been produced to specify learning object meta-data to support the delivery, reusability, and searchability of learning resources [5]. They specify the format, syntax, and semantics of data. These learning objects are intended to be assembled into higher-level units, so-called blocks (for example, course).

The IEEE LOM Draft Standard defines a set of meta-data elements that can be used to describe learning resources. This includes the element names, definitions, data-types and field lengths. The specification also defines a conceptual structure for the meta-data [65].

2.6.2 Content Package

The content packaging specification provides a description of packaging learning materials, such as course or a collection of courses, into interoperable and distributable packages. It provides a definition on how the content has to be packed, described and exchanged. The IMS Content Packaging Information Model describes data structures that are used to provide E-Content wrapping and packaging to exchange learning contents across platforms and applications, such as content creation tools, learning management systems, run-time environments, etc. [65].

The Sharable Content Object Reference Model (SCORM) [1] defines a conceptual model of how to handle, package and deliver learning contents. It includes a variety of learning standards (AICC, IEEE, IMS, etc.). The SCORM specification provides support for adaptive course strategies, whereby sequencing course contents could be controlled by scriptable prerequisites [5].

2.6.3 Learning Design

Learning Design emphasises process over content. A good learning process design is at least as important as content design [31]. It provides a generic and flexible standard to enable various pedagogies to be expressed. It tends to ensure effective results whereby a good pedagogical design includes somewhat structured pedagogical scenarios [113]. There is a great value in identifying effective learning design models and describing them in a standard form. This would embed pedagogical requirements in the norms being developed to help in developing good teaching practices [77]. Learning scenario structures are detached from the learning resources to enable learning design reusability [62, 74].

The IMS Learning Design (IMS LD) specification was developed to formalise learning designs that contain content and activities. The IMS LD [65] is an improvement on the educational Modelling Language (EML) designed by the Open University in the Netherlands in the late 1990s, which is focused on the performance of individual and group-learning activities [123]. IMS LD defines a structured XML-based language to describe and model a learning design in a formal way. It provides a counter to the trend towards designing for lone-learners reading from screens. It guides staff and educational developers to start not with content, but with learning activities, and recognises that there are important opportunities to learn when learners cooperate to solve problems in social and work situations [73]. IMS LD's main objectives are:

- To describe learning design in a formal way that enables the learner to attain particular objectives by performing learning activities in a certain learning environment.
- To enable the description of the learning design based on different pedagogical models (for example, Competency Based Learning, Problem Based Learning).
- To classify the learning components according to pedagogical meta-data (PBL Identify, PBL Define, PBL Analysis, etc.).
- To encourage reusability of the learning designs. The design skeleton can be reused across different domains and portable to any IMS-LD run-time environment.

Figure 10 shows the way the major elements of the learning design specification are hierarchically ordered (an asterisk * means that an element may occur more than once).

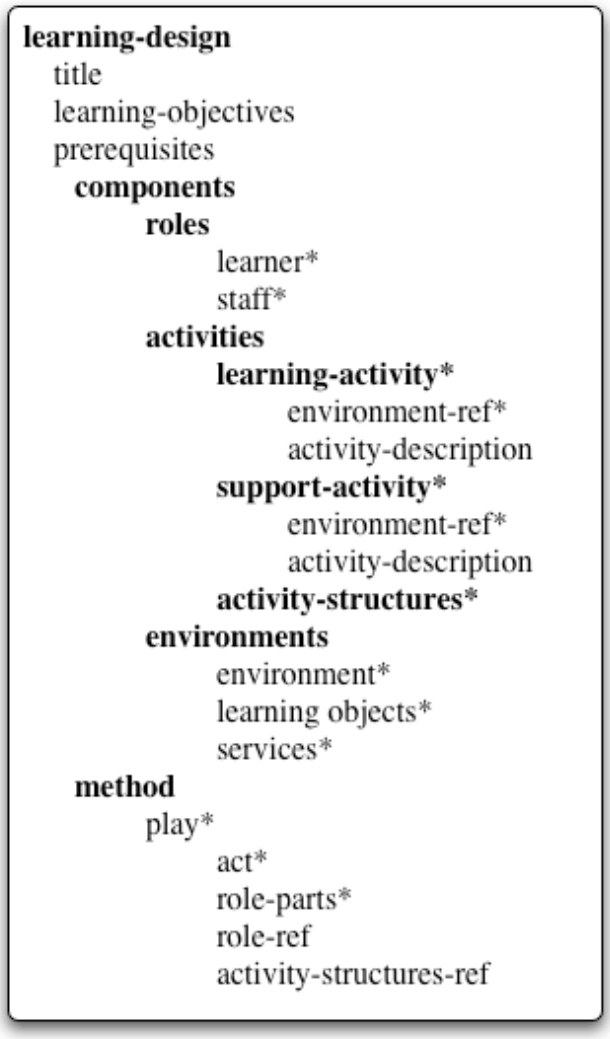


Figure 10: The basic IMS LD schema [65]

The components section provides the "building-blocks" for the method section of the learning design. These building-blocks are: roles, activities, and environments. Activities are one of the core structural elements of the learning design. They form the link between the roles and the learning objects and services in the learning environment. They describe the activities that a role has to undertake within a specified environment. There are two basic types of activities: learning activities and support activities. The learning activity consists of a single activity-description, while the support activity is meant to facilitate a role performing those learning activities. Learning activities can be

arranged and structured in the activity-structure section. The environment is the section that contains the collection of learning objects and services. Learning objects are defined as any reproducible and addressable digital or non-digital resource. Service facilities are used during the teaching and learning, for instance, a discussion forum or some other communication facility. The method section contains another core part of the learning design specification, which is the play section. A play specifies the actual learning design, the teaching-learning process, referring to the components declared in the component section. In the play, it is specified which roles perform what activities in what order. A play is modelled according to a theatrical play with acts and role-parts. In general: a play consists of a sequence of acts. In each act, different activities are set for different roles and are performed in parallel. When an act is completed, the next act runs until its completion [66].

3 Beehive's Conceptual Model

This chapter describes in detail the conceptual model for our pedagogical application framework. This framework is an extension of the application framework concept that is used in the software engineer field. We called it "pedagogical application framework" since it takes into account the reusability of both pedagogical designs and software components. The theoretical ground provided in the Background Chapter provides the appropriate base to describe such framework. The main objective of this framework is to integrate the domain-specific knowledge from educational designers and use it as the basis for designing a set of software components needed to assemble a wide range of synchronous collaborative learning tools. Section 3.1 describes the early version of Beehive's conceptual model and presents its limitations. We next present the current 4-layered conceptual model and discuss its potential advantages. Each layer in this conceptual model is explained in detail in the following sections. Section 3.2 introduces the Task Patterns layer and how this layer is related to the Collaborative Learning Patterns layer. It explains in detail how we have identified these common tasks and how they can be reused in assembling any collaborative learning design. Section 3.3 explains in detail the CSCL Components layer and how these components are mapped to the corresponding task patterns in the second layer in a systematic way to hide the implementation complexity. Section 3.4 discusses how groups of learners can be structured by various roles, and the various grouping methods that can be applied within the collaboration sessions. Section 3.5 defines and describes the proposed collaboration

script. It explains in detail how scripts bring and organise all pieces together (Tasks, CSCL Components, Roles, Groups) to support collaborative learning sessions.

3.1 Framework Description

A key design requirement for such a framework is that instructors should be able to manage, customise and reuse ideas in the whole design process. In order to support the complex and interdisciplinary development of the pedagogical collaboration tools needed for such designs, a multi-layered architecture is suggested to particularise different domains' concepts. Each layer has been designed for a tight integration with other layers, but still described independently within its own discipline. For example, if a user wants to use the collaborative learning platform in a business application, the top layer can be adapted to the new desired situation by defining new training design patterns in that discipline, for instance, eliciting requirements pattern.

We start this section by describing Beehive's evolution to present the rationale of the current Beehive's conceptual model. Our first attempt was designing a 2-layered conceptual model with its main objective being to provide instructors with a specialised tool for each collaborative learning technique (similar to the model described in Section 2.3.4.2). This approach had suffered from various limitations, such as the large number of existing collaborative techniques; the same technique might be performed in different ways, and instructors are limited to select form certain a list of collaborative learning techniques.

The next attempt was designing a 3-layered conceptual model. The benefit of this model is that its abstraction is driven by the educational field, while its implementation is driven by the software field. It basically presents a set of task patterns as an intermediate layer necessary to support the flexible implementation of various collaborative learning techniques. The 3-layered conceptual model is shown in Figure 11. The design process in this framework is relatively simple. First, an instructor starts by selecting a collaborative learning pattern from a list of patterns according to certain

environmental forces and the kind of problems learners need to solve (learning objectives) and its context (as explained in Section 2.1.3). Second, based on design reuse, the system automatically presents a suggested list of tasks to be performed by the instructor and learners. At this point, the instructor's participation in the design process terminates. Third, the system maps each task to a certain CSCL component and then assembles all mapped components in a single CSCL tool. The advantage of this approach is that it takes instructors' learning experience into account and at the same time hides technical difficulties [130].

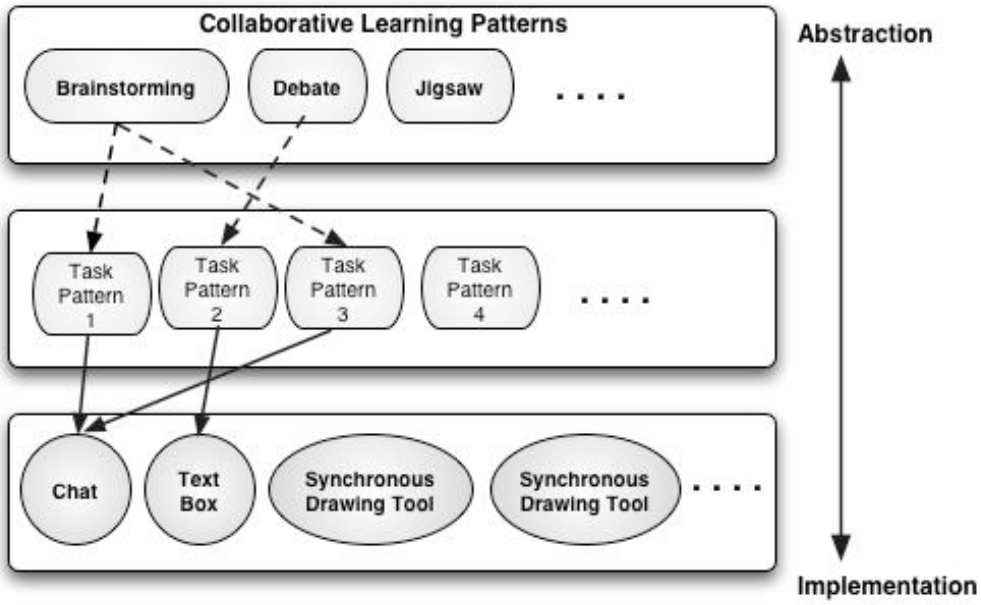


Figure 11: Beehive's 3-layered conceptual model

Nevertheless, there were two main challenges in this framework. The first challenge was that the resultant tools were usually crowded with components especially for techniques that require many tasks. Figure 12 shows the first version of Beehive's run-time application, implementing the Group Nomination technique. Learners found it confusing to deal with a tool that requires their concentration on many components at the same time. Usually one component is used in each step. Learners needed to watch all

components to realise which component is active. The web page size limitation also made it difficult to show all components of the session at the same time. Participants needed to scroll down and up to engage in the appropriate component. The second challenge was the increasing cognitive load on participants; they needed to understand, memorise and synchronise their activity process according to the session's scenario.

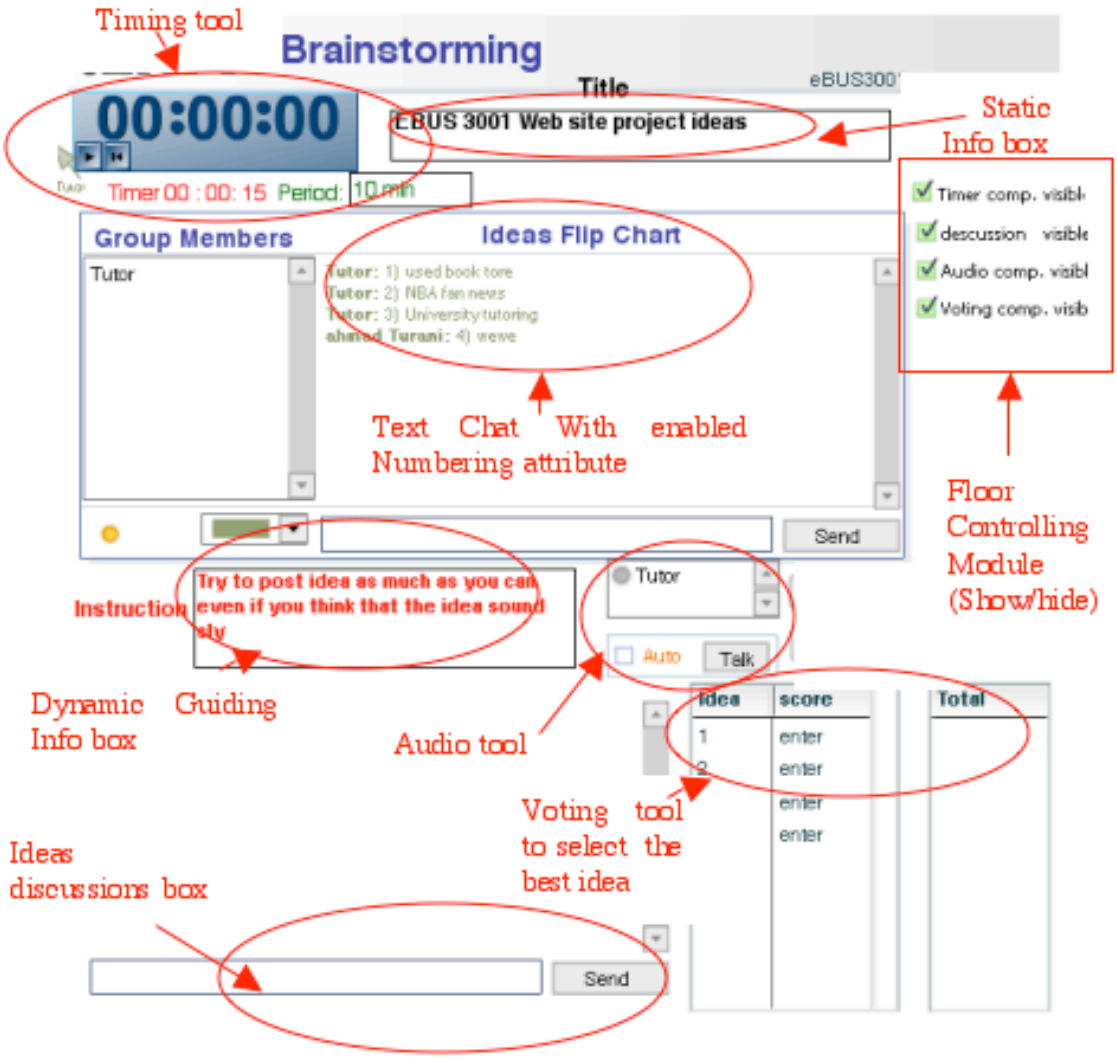


Figure 12: Screenshot of the first version of Beehive's run-time application

As shown in Figure 13, we have added one extra layer that contains two components to the Beehive's conceptual model: Collaborative Script component and Group Structure component.

As explained in Section 2.3.3, collaborative learning patterns have internal structures. The structure represents a set of steps for a group to follow. They also specify who will do what in each step. We have used this concept to enhance this framework. The Collaborative Script component is used to formalise the structures of collaborative techniques in a systematic way to present CSCL components in a sequential manner. It enables CSCL systems to present only the specified CSCL components related to the step that is reached in the session. It presents not only the specified CSCL components, but also the associated group setting, roles, and time limit of that step. It carries out in a sequential manner the pre-designed sequence of activity steps to facilitate learners, within their groups, to play certain roles in collaborating towards certain outcomes according to a specific pattern's scenario.

The Group Structure component specifies the internal structure of the groups (group size, roles, etc.). The Group Structure component is a dynamic component. There might be more than one group setting during the session. For example, in the Pyramid technique, participants start discussing in a small group and then move to larger groups until they all join in a single large group.

Finally, even though this framework encourages instructors to start their design by reusing existing collaborative learning patterns (starting from the first layer), it still allows instructors to design new techniques by starting their design from the second layer.

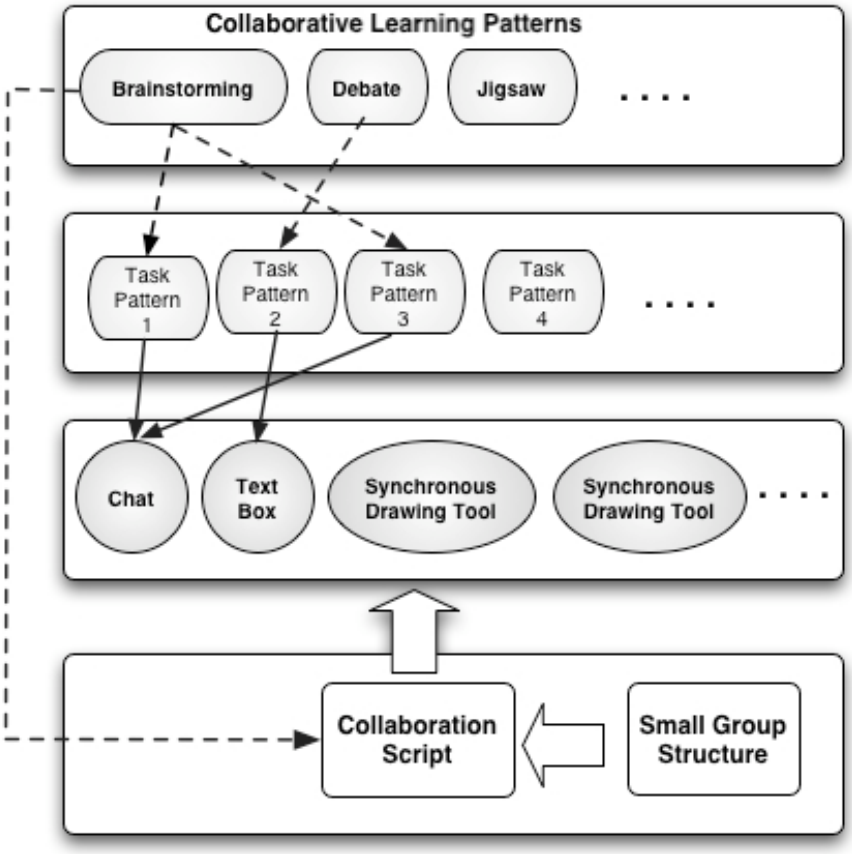


Figure 13: Beehive's 4-layered conceptual model

3.2 Task patterns Layer

By analysing the essential characteristics of face-to-face collaborative learning patterns, it is clear that these patterns are mainly based on a set of tasks for participants to perform [40, 77]. These tasks are usually arranged as a list of sequential steps needed to be performed by instructors and learners. For example, in the Group Nomination Pattern, learners need to perform several tasks, such as: to brainstorm ideas, to discuss these ideas, to prioritise these ideas, and finally to submit the chosen idea. The instructor's tasks are to provide the topic and to include some appropriate background.

We have realised that collaborative learning patterns have many task commonalities between them. This important characteristic can help in dimensioning tasks, after decoupling them from their resources, which can be reused to assemble a wider range of collaborative learning patterns. There are two main benefits from undertaking this. First, it will help software developers to identify and built the appropriate CSCL components that can be reused in facilitating new designed techniques. Second, it will be easier for instructors with little technical background to identify and select tasks rather than tools.

When we decomposed some of these patterns and decoupled them from their explicit resources, we were able to identify some of these recurring tasks, such as a group discussion task, information provision task, reflecting task, etc. A finalised list of tasks has been identified after decomposing the 10 collaborative techniques according to their procedures described in Section 2.3.3. The list contains these 14 common tasks, as shown in Table 2.

*	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Brainstorming			√							√			√	
Buzz group	√	√				√	√				√			
Debate	√									√		√		√
Group discussion	√				√					√			√	
Group nomination	√		√	√			√							
Jigsaw	√					√				√		√		
Pyramid	√	√								√				
Role-playing	√								√	√		√		
Team-pair-solo	√	√								√			√	
Think-pair-share	√	√					√			√			√	

Table 2: The identified common tasks in the 14 collaborative learning patterns

* 1- Discussing 2- Reflecting 3- Idea Generating 4- Idea Selection 5- Ice-breaking 6- Resource Searching 7- Reporting 8- Reflecting 9- Debriefing 10- Session Info reading 11- Group Reading 12 – Role Reading 13- Resource Reading 14- Voting

Even though Beehive describes and models collaborative learning, tasks are not limited only to collaborative types. Some tasks need to be carried out individually as shown above, such as Searching, Private Reflecting, etc. There are also tasks that cover

other issues such as resource provision, human support and logistics. Any successful learning design should encompass all these issues that might be beyond this framework's main focus, but are still considered as an essential part of the success of learners' overall learning experience.

We have chosen to categorise these tasks according to the Activity-Centred Model, described in Figure 3, since collaborative learning is based mainly on group-based activities. In that model there are five main categories: *Collaborative Tasks, Individual Tasks, Feedback & Assessment Tasks, Resources Provision tasks*, and finally *Supporting Tasks*. The above 14 identified tasks can be listed in four of these categories (*Collaborative, Individual, Resources*, and *Feedback*).

First we start by listing *Collaborative Tasks*. These tasks are the tasks that are need to be done collaboratively within each group:

- **Group Discussion Task**: This task enables participants to engage in a text/audio discussion chat.
- **Idea Generation Task**: This allows learners to post their ideas on a posting chart
- **Idea Selection Task**: This enables learners to prioritise ideas in order to select the best idea among them.
- **Collaborative Reflecting Task:** This enables participants to write a common text, such as a group report, annotate a shared text, or fill in the shared text's blanks.
- **Icebreaking Task**: This asks participants to introduce themselves to others at the beginning of a session.

A few other tasks can be added to provide more online collaboration flexibility, as illustrated in Figure 14. The video-conferencing can be an alternative for text-based group discussion to enable faster communication. The Collaborative Image Drawing/Annotating is an alternative for the solely text-based collaboration. The Collaborative Text Discussion Task is more appropriate for texts that need to be discussed part-by-part. The Slide Presentation Task enables the delivering of online

presentations. Finally, the File-sharing Task is added to enable learners to share their resources with others. The extended *Collaborative Tasks* are:

- **Conferencing Task**
- **Collaborative Image Drawing/Annotating Task**
- **Collaborative Text Discussion Task**
- **Slide Presentation and Discussion Task**
- **File-sharing Task**

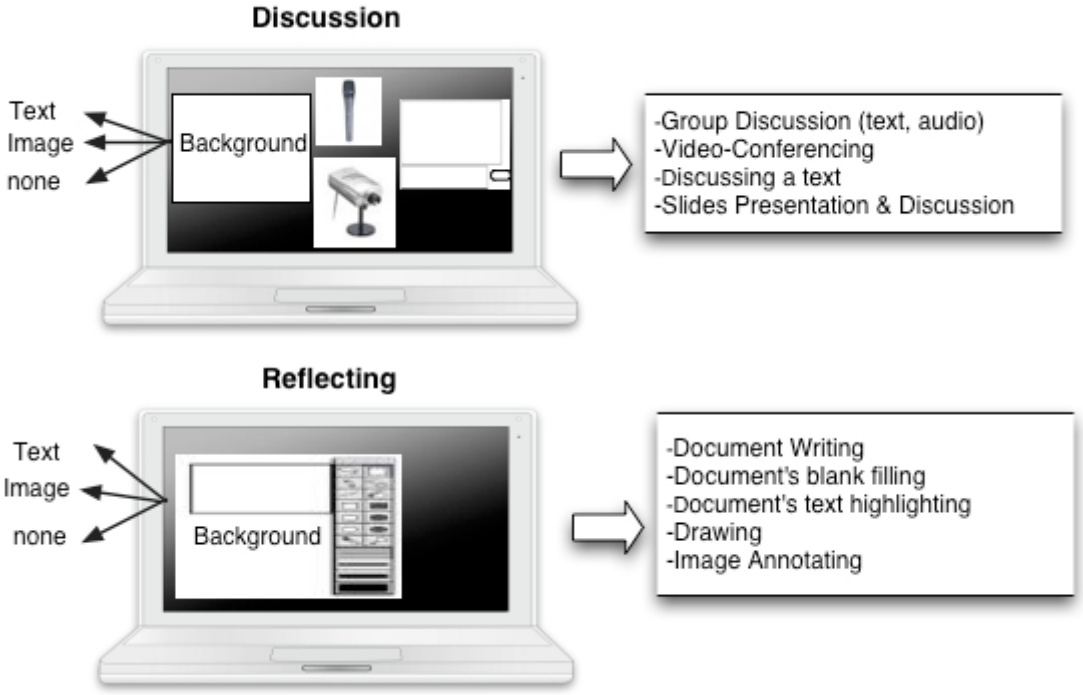


Figure 14: Presenting suggested types of online discussion and reflecting facilities

The second category is the *Individual Tasks*. These are the tasks that need to be carried out individually by all participants:

- **Private Reflecting Task**: This requires learners to reflect briefly on an issue raised by the instructors individually.
- **Private Thinking Task**: This enables learners to think privately about a posted issue before engaging in a group activity.

- **Searching Task**: This asks learners to search the net or other means for information needed in the session.
- **Outside Task**: As shown in Figure 15, this is a general task that asks learners to perform outside the session's task (for example, visiting a museum, posting in a forum, etc.). This task is important in providing more flexibility to the online session to enable blended learning.

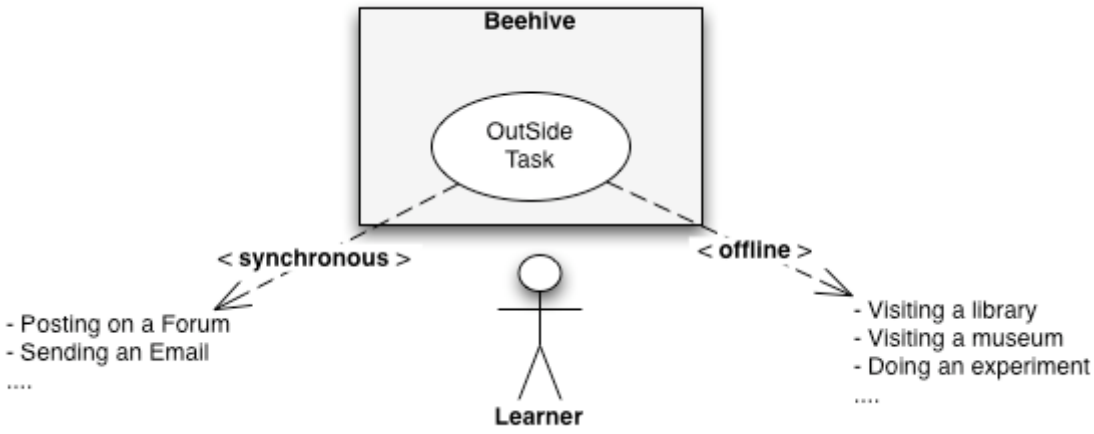


Figure 15: The Outside Task Pattern representation

The third category is the *Feedback and Assessment Task* category. This category groups tasks that are related to participants' (individual, group, and instructors) feedback. It includes:

- **Resolution Reporting Task**: This asks learners to submit the summary of their results as a group or individuals to the instructor.
- **Presenting Questions Task**: This enables instructors to present questions of different types (short answers, true or false, multiple answer, survey, etc.) to learners at a certain stage of the session.
- **Results Evaluation Task**: This provides instructors with a facility to evaluate participants' works.
- **Debriefing Task**: This allows facilitators to debrief the outcomes at the end of a session.

The fourth category is the *Resources Provision Tasks* category. It includes tasks that are related to providing the collaboration session with its resources. These tasks are:

- **Providing Objective Task**: This asks instructors to write down the objective, title, or problem statement of the session in a clear way.
- **Providing Info Task**: This allows instructors to provide information that is used in the session (to all participants, certain roles, or certain groups).
- **Uploading Resources Task**: This enables the instructor to upload resources files to the session.
- **Providing Session's Scenario Task**: This allows learners to review a session's plan before engaging in the session.

The final category is the *Management & Support Tasks* category. For more flexibility, some tasks are necessary in managing and supporting online sessions, such as:

- **Group-forming Task**: This enables facilitators to specify the method of selecting the group (random by the system, by learner, by instructor, etc.).
- **Timing Control Task**: This allows facilitators to control the time during a session in order to extend or shorten the times for certain steps.
- **Session Overview Task**: This asks facilitators to provide an overview at the beginning of the session of what learners are expected to do in the session.
- **Side-note Taking Task**: This allows learners to prepare and organise their thoughts during the session regardless of the steps they reached.
- **Monitoring & Guiding Task**: This allows facilitators to monitor and guide learners in different groups.
- **Interaction Tracking Task**: This provides participants with a facility to track their interaction levels, for more self-encouragement.
- **Online Helping Task**: This allows learners to ask for the facilitator's help to clarify the session's issues.

As mentioned before, these 29 common tasks not only allow the construction of the above 10 collaborative techniques, but they are flexible enough to form many new collaborative techniques.

The tasks repository, shown in Figure 16, includes all 29 common tasks and presents them according to the five categories. This enables the instructor to find the desired tasks needed in his design.

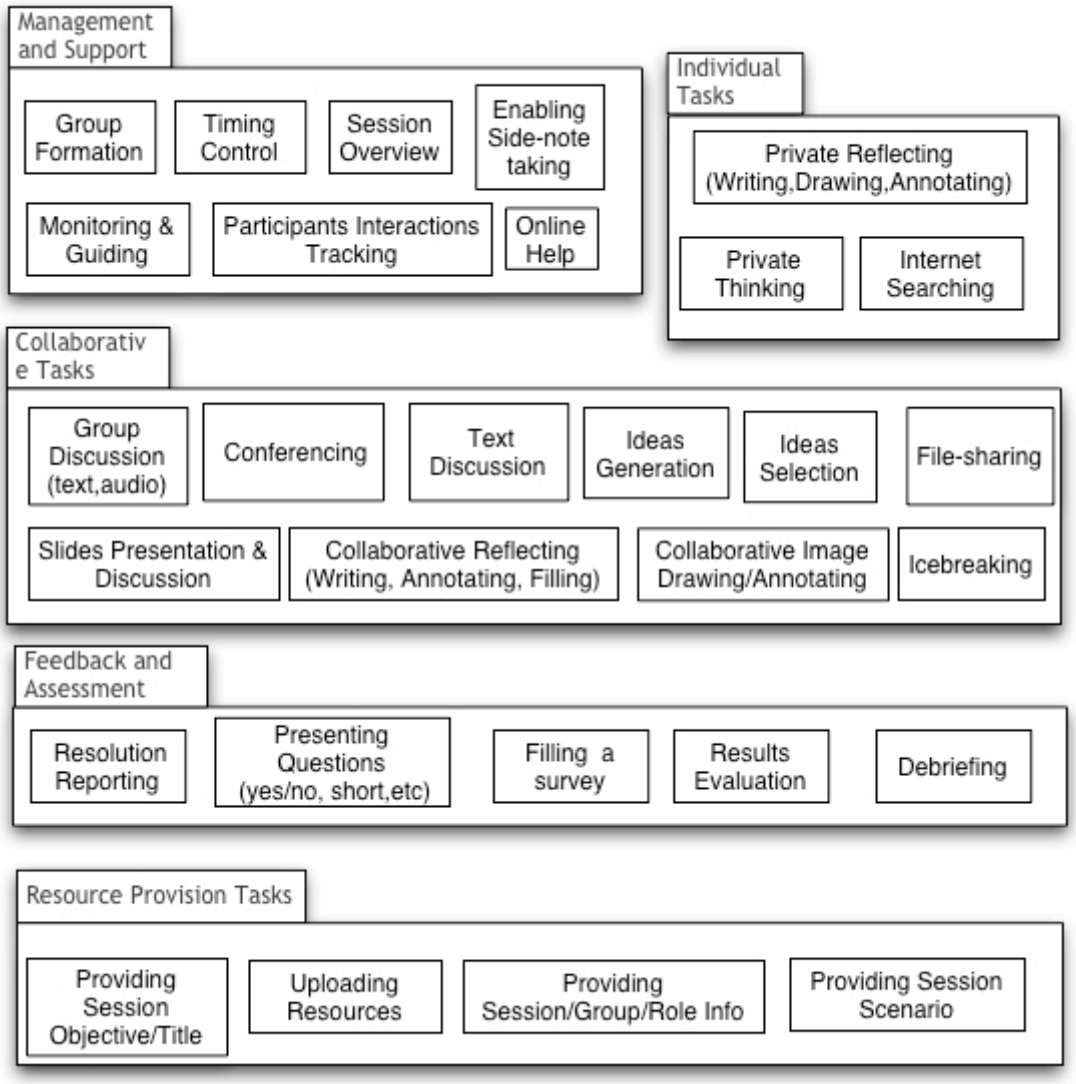


Figure 16: Task repository

To minimise the gap between the learning process model and the software model, these common tasks are presented in pattern format (described in Section 2.3.1) [30]. These task patterns can be easily identified by instructors and at the same time provide some guidance for software developers to develop appropriate CSCL tools [35]. For example, the Timing Control Task pattern is presented as follows:

i. **Problem**: How can a facilitator control the time of an activity?
ii. **Forces**: Some activities require timing. For more synchronisation, the time should be displayed on participants' machines. The timer should be controlled during the session so facilitators can extend/pause the time of a certain stage if needed.
iii. **Solution**: Providing a tool that displays the duration on the top and shows the time passed on the bottom. It must contain three buttons to control the timing, start, pause, and reset, which are only visible to the facilitator.
iv. **Related Pedagogical Techniques**: Mandatory to all pedagogical techniques
v. **Tool Example:**

As shown above, it is possible for instructors to learn more about the Timing Control Task pattern, and it helps software developers to build such a tool.

We also have checked that these task patterns cover all the six activity patterns described in Section 2.3.4.3. The "Look" activity pattern relates to learners' content consumption tasks, such as reading session info, session scenario, group info, role info, and session resources. The "Discuss" activity pattern relates to some collaborative task

patterns, such as group discussion, conferencing, document discussion, and slide discussion. The "Deposit" activity pattern relates to tasks that focus on sharing knowledge with others, such as idea-listing, file-sharing, collaborative text-writing/annotating/filling, and collaborative image-drawing/annotating. The "Feedback" activity pattern relates to questioning, reporting, evaluating, and debriefing tasks. The last activity pattern is the "Do" pattern, which relates mostly to the individual tasks, such as private reflecting, drawing, annotating, and searching.

3.3 CSCL Components Layer

This layer represents a collection of software components (bricks) that can be mapped automatically to tasks chosen by instructors. Each component facilitates a specific task. Software developers can use the information presented in the task's pattern to build the appropriate component. For instance, a chat component can be used to facilitate the Group Discussion task pattern. In this approach it is essential to identify and dimension the CSCL components needed to support various collaborative learning activities. Since there are 29 identified task patterns in Beehive, there is a need to identify 29 CSCL components. Other approaches have relied on instructors' experience to select the most appropriate tool to support their particular collaborative learning situations. For example, in service-based CSCL systems [130], instructors need to spend substantial effort in searching and learning how to use appropriate CSCL components.

Fortunately, there is another level of commonality found in these components, from the technical point of view. More than one task can be related to the same component but with some customisation. For instance, Group Discussion and Idea Generating tasks can be related to the same component (for example, Chat component). To promote software reusability, component-based software engineering introduced the concept of reusing the same components to construct various applications by parameterising the component's meta-model [40, 74]. For example, the Ideas Chart and

Ideas Discussion components are subclasses of the Chat component but with different parameter values.

We have applied the same concept to map more than one task pattern to the same components, as presented in Figure 17. We have identified 12 software components that are needed to furnish all 29 tasks: *Static Info Viewer, Dynamic Info Viewer, Text Chat, Whiteboard, Shared Pointer, Timer, Voting, Audio, Video, File Uploader, and Slide Presenter*.

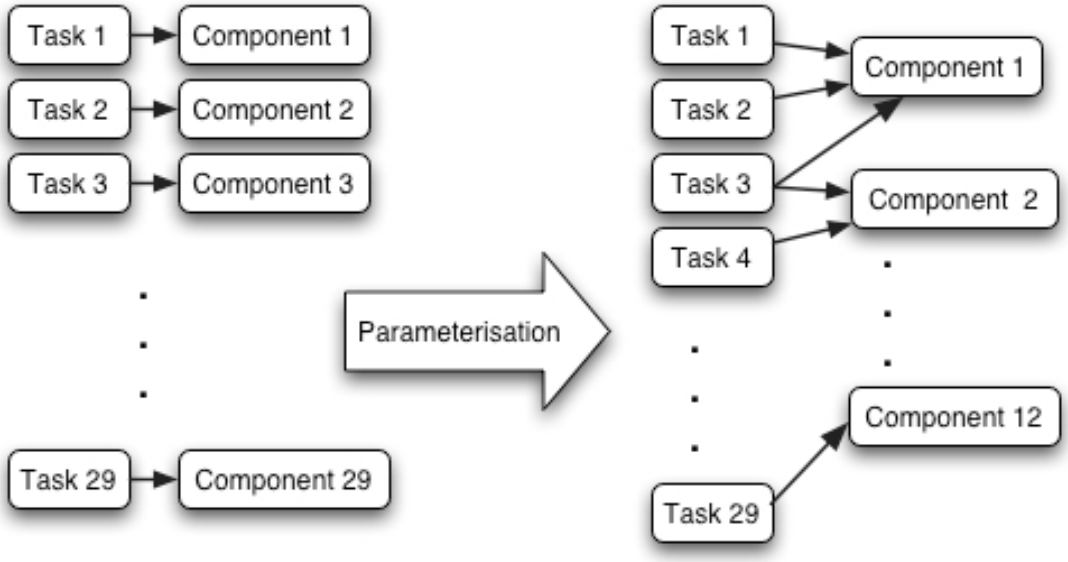


Figure 17: Components' reusability

Some of these components (*Text Chat, Whiteboard, Audio, Video, Voting, and File Uploader*) are similar to the synchronous tools discussed in Section 2.2.2. Regarding the other components: the *Static Info Viewer* component is used to hold information that does not change during the session, while the *Dynamic Info Viewer* component holds changing information. The *Timer* component is used to enable instructors to control the session process by providing more time or by speeding up the process. The *Shared Pointer* can be used in discussions to enable participants to draw

other participants' attention to a certain part of a text/image during the discussion. Finally, the *Slide Presenter* component is used to facilitate presentation.

The following section discusses in detail how the 12 software components can be mapped to the 29 tasks:

1. The Text Chat component can be used to support the following tasks: Group Discussion, Resolution Reporting, Presenting Questions, Idea Generating, Icebreaking, Debriefing, and Online Help task. Four parameters can be used to differentiate some of these tasks: *Insert, Delete, Contribution Type, and Contribution Visibility*. The *Insert* parameter allows the system to insert certain default text before the actual contributed text; such as inserting the group number to which the contributor belongs, the sequential contribution number (for ideas listing), etc. The *Delete* parameter allows for deleting certain text from the contributions, such as the contributor's name, for anonymity. *Contribution Type* could be "Single" or "Multiple"; "Single" indicates allowing only one contribution per participant (for example, answering a posted question, reporting a result, etc.), while "Multiple" indicates that participants can send as many contributions as they desire (for example, group discussion). *Contribution Visibility* is either "Always", whereby contributions are always shown to participants or "Controllable", whereby facilitators can control the visibility during the session (for example, when all learners answer a posted question, the instructor may decide to show all answers to them). Table 3 summarises the tasks that are related to the Chat component.

Tasks	***Insert***	***Delete***	***Contribution Type***	***Contribution Visibility***
Group Discussion Icebreaking Debriefing Online Help	None	None	Multiple	Always
Anonymous Discussion	None	Name	Multiple	Always
Resolution Reporting	None	None	Single	Always
Idea Generating	Num.	Name	Multiple	Always
Presenting Questions	None	None	Single	Controlled

Table 3: The Chat component parameters

2. The Static Info Viewer component is used to hold information that does not change during the session. This component is related to the following tasks: Providing Objects/Title, Providing Session's Information, Providing Group Information, Providing Roles Information, Providing Session's Scenario, Text Discussion, Idea Selection, and Filling in a Survey/Poll task. Only one parameter differentiates the Text Discussion task from other tasks: the *Annotation* parameter. It defines whether text annotation is permitted. In Text Discussion, participants might need to annotate certain parts of the text during discussion.

3. The Dynamic Info Viewer component is used to hold information that changes during the session and is related to the following tasks: Guiding, Collaborative Reflecting (Writing, Annotating, Filling), Private Reflecting (Writing, Annotating, Filling), Side-note Taking, and Results Evaluation. Three parameters are used to differentiate some of these tasks: *Update*, *Default Text*, and *Annotate*. The *Update* parameter defines the event that updates the text. For instance, when an instructor sends guiding text to participants, he needs to click the send button. While in the Collaborative Text Writing task, as soon as text has changed, it will appear to other participants. The *Default Text* contains the default text presented to learners. For example, in the filling in the blanks task, the default text will be the text that contains blanks needing to be filled in by participants. The *Annotate* parameter defines whether text annotation is permitted.

Task	***Update***	***Default Text***	***Annotate***
Monitoring and Guiding Results Evaluation	Send-Clicked	None	None
Collaborative Writing Private Writing	Text-Changed	None	None
Collaborative Annotating Private Annotating	Text-Annotated	True	True
Collaborative Filling Private Filling	Text-Changed	True	None
Side-note Taking	Text-Changed	None	None

Table 4: The Dynamic Info Viewer parameters

4. The Voting component is related to the following tasks: Filling in a Survey/Poll, Yes/No Voting, and Idea Choosing/Marking task. Two parameters can be used to differentiate between these tasks: *Idea Source* and *Type*. The *Idea Source* parameter defines the source of ideas. The instructor can be the source of providing ideas for voting (for example, Survey), while the other source can be learners themselves (for example, Idea Marking). Finally, for the *Type* parameter, there are different types of voting, such as "Yes/No", "Multiple", etc.

Task	***Idea Source***	***Type***
Filling in a Survey/Poll	Instructor	Multiple
Yes/No Voting	Instructor	Yes/No
Idea Choosing	Learners	Multiple

Table 5: The Voting component parameters

5. The Whiteboard component is related to the following tasks: Collaborative Drawing, Individual Drawing, Collaborative Image Annotating, and Individual Image Annotating tasks. Only one parameter differentiates drawing from annotating; this is the *Background Image* parameter. Drawing is usually done on a blank background, while annotating is done over a background image.
6. The Shared Pointer has no parameters and relates to the following tasks to allow participants to point to a certain part while talking: Collaborative Drawing: Collaborative Image Drawing, Overview, and Slide Presenting tasks.
7. The Audio also has no parameters and relates to tasks that require vocal communication: Text Discussion, Slide Presentation, Overview, and Icebreaking tasks.

8. The Video component relates to the following tasks: Conferencing and Icebreaking.
9. The Timing component relates to the Timing Control task.
10. The File Uploader relates to the following tasks: File-sharing and Session's resources tasks. Only one parameter differentiates these two tasks, the Source parameter. For the File-sharing task, the Source parameter consists of learners.
11. Slide Presenter relates to the Slide Presentation task.

Some task patterns map to no components, others to only one component or more than one component. For instance, the Outside task pattern is mapped to no components while the Providing Session Info task pattern is mapped to only one component, the Static Info Viewer component, and the Text Discussion task pattern is mapped to two components: the Static Info Viewer component to present the text and the Chat/Audio component to facilitate the discussion.

Table 6 presents all tasks and the components to which they are mapped.

Task	**Software Components**
1- Group Discussions	Chat, Audio
2- Idea Generation	Chat
3- Idea Selection	Voting, Dynamic Info Viewer
4- Collaborative Reflecting	Dynamic Info Viewer
5- Icebreaking	Chat, Audio, Video
6- Conferencing	Audio, Video

7- File-sharing	File Uploader
8- Collaborative Image Drawing/Annotating	Whiteboard, Shared Pointer
9- Collaborative Text Discussion	Static Info Viewer, Audio, Chat
10- Slide Presentation and Discussion	Slide Presenter, Audio, Chat
11- Private Reflecting	Dynamic Info Viewer
12- Private Thinking	None
13- Internet Searching	None
14- Resolution Reporting	Chat
15- Presenting Questions (short answer)	Chat
16-Presenting Questions (true/false and multiple answers)	Voting
17- Results Evaluation	Dynamic Info Viewer
18- Debriefing	Chat, Audio
19- Providing Objective/Title	Static Info Viewer
20- Providing Info (all, role, group)	Static Info Viewer
21- Uploading Resources	File Uploader

22- Providing Session's Scenario	Static Info Viewer
23- Group-forming	Group-forming
24- Timing Control	Timer
25- Session Overview	Static Info Viewer, Audio, Chat
26- Side-note Taking	Dynamic Info Viewer
27- Monitoring & Guiding	Dynamic Info Viewer
28- Participants' Interactions Tracking	Tracker
29- Online Helping	Chat

Table 6: Mapping tasks to software components

This list of task patterns is only a primary list. It can still be extended to reach new collaborative scenario tasks. Identifying new task patterns is flexible in this framework. There is no limit to the possible combinations of components that an instructional designer can select and parameterise to furnish new tasks. The process starts by first identifying the new task needs to be added to the list. Then the designer needs to fill in all information related to the task in a pattern format (Problem, Solution, etc). After that he needs to select which components are needed to facilitate that task. Finally, he might need to configure some of these components' attributes to properly accommodate the new task. The new task will also be added to the task list for reusability. However, software developers might need to develop new components to furnish a specific learning situation's task.

For each component, there are two user interfaces, the **Actor Interface** and the **Audience Interface**. The **Actor Interface** enables users to send their contributions

through these components (for example, sending contribution button, pushing to talk button, dragging the mouse to draw, etc.) and also to control the component (forwarding the slides, forwarding the timer, clearing the chat history, etc.) while the **Audience Interface** only allows the audience to view the actor's contributions (for example, viewing chat history box, listening to audio, watching the video screen, watching the drawing board, etc.).

3.4 Group Structure

A small group setting is essential in conducting collaborative learning [136]. Group formation is a complex task because it involves various issues, such as learners' roles and organisational factors [55]. The first step in the group formation is not the size but what type of roles these groups contain. In collaborative learning, assigning roles indicates which role has the right to speak in each step during the process [138]. Group size is also crucial to the collaboration successes. It might vary according to which technique is used. For example, in the Group Nomination Technique, the group size is usually around five (four general participants and one Chairperson) while in the Three Person Debate Technique, the group size is larger (about six debaters and six speakers). The size may vary from n=1 (individually) to n=2 (peer-to-peer) to $2 < n < 10$ (small group) up to $n>10$ for a large group (for example, the entire session).

The group structure could be dynamic. The group size in some techniques changes from stage to stage, while in other techniques, the group's participants changes. For instance, in the Jigsaw technique, participants join in the expert group then they move back to their jigsaw group.

The various grouping methods should be also taken into consideration. In Beehive there are four proposed methods: Learners, Instructors, Systematic, and Random. The Learner method allows learners to select their roles and groups themselves. The second method is to allow instructors to assign learners to various roles and groups. In this method, instructors are usually provided with a tool that lists all

learners in the class. Instructors usually form groups according to some constraints (for example, group size, available groups, available roles, etc.). This method provides instructors with more control over the session design but it may require a lot of his resources (time and effort), especially with large classes. He also needs to be familiar with his learners' capabilities. The third method allows the system to assign learners according to their background and expertise. Usually, learners with diverse backgrounds tend to reach common goals and solutions more successfully [14, 37]. The last method allows the system to automatically assign learners to roles and groups on the basis "who comes first will be assigned first".

The challenge in the first three methods that the designer should be aware of is that they might partially fill some groups or some participants might miss their sessions. This may cause difficulty in conducting a successful session for these unfilled groups. To overcome this, the system might redistribute the unfilled groups' members to other groups before starting the session. If the unfilled group size is >= minimum size, then the system will allow that group to exist. Otherwise that group will be terminated and its members are distributed to other groups by using their maximum capacity (maximum size). In the last grouping method, only the last group may suffer from being unfilled.

3.5 The Collaboration Script Layer

A collaboration script is needed to organise and synchronise tasks in a certain order according to the activity's basic scenario, as described in Section 2.5. This methodology has originated from an approach to describe any pedagogical technique using a pattern language [54]. The scripts of various Collaborative Learning patterns are embedded in their solution section. The solution section usually describes the procedure of a series of steps the participants need to take [129]. Formalising these steps enables the CSCL system to facilitate collaboration sessions in a synchronised way [90]. In Beehive, the collaboration script is presented as a play's scenario. Each step in the script can be presented as a scene. The instructor's role is presented as an orchestrator and the

learners' roles as actors. This methodology is important in simplifying the script representation to enable designers with little technical background to learn and use. As illustrated in Figure 18, each scene is composed of five elements: Instruction, Actor, Timing, Environment, and Audience Setting. The Instruction is basically the information presented to participants during the scene. The Actor represents the role that will be performed in that scene. The Timing defines the timeframe of that scene. The Tools represents the tools and resources needed to facilitate that scene. The Audience Setting defines how audiences are grouped to watch the actor'/s' performance.

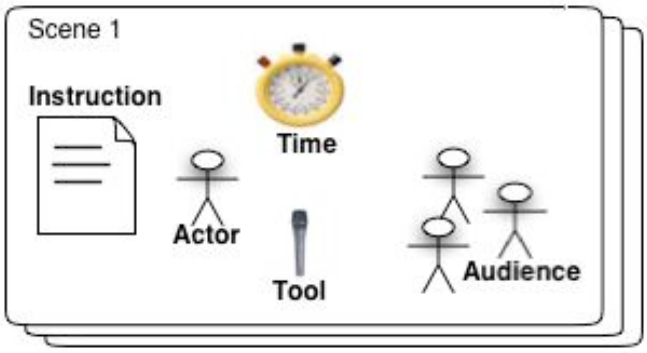


Figure 18: Scene's elements

The Instruction element is important to clarify the task that needs to be performed in that activity. The instructions are prompted to learners in each step. For instance, in the Group Nomination pattern, the first prompted instruction is the global instruction "Start posting your ideas", then "Discuss these ideas", until the last step where a specific instruction "Submit your solution" is prompted to the chairperson.

To act according to a script, participants have to know their roles in the session's script. Therefore, their roles should be indicated on the computer screen, in order to prevent role confusion [137]. Depending on the role, participants can have different access rights to the collaboration tools [16]. In the last step of the Group Nomination Technique, participants with the role of "chairperson" will be the only ones who can submit the selected idea to the facilitator and to the other groups. Therefore the tool

presented to chairpersons in that step will contain, for instance, the send button, while others do not receive it.

Audience and Actor settings are driven in Beehive initially from the different pedagogical models identified by Paulsen [106], who identified three types (*One-Alone, One-to-Many, Many-to-Many*). The limitation to this model is that it classifies the whole technique in a single setting, while it is obvious that inside a single collaboration technique there might be several settings. For instance, a session might start as a *One-to-One* setting (peer discussion) and then changes to *Many-to-Many* (entire session discussion). An extension is needed to cover different types of settings. First, the *Some-to-Some* setting represents small group collaboration where all participants perform in front of each other. *Some-to-Many* indicates that multiples roles perform in front of the entire session. *One-alone* indicates that participants will engage individually in the activity, such as participants privately reflecting on a certain issue. *All-Alone* indicates that all participants will be just an audience, such as participants reading the session info. In the *One-to-Some* setting, a specific role will perform in front of his group, while in the *One-to-Many* setting, he performs in front of the entire session (for example, the teacher presenting in front of the class). In the *Many-to-One or Some-to-One* settings, learners perform at the front in a single role (for example, learners submit their results to a teacher). All 10 group settings are demonstrated in Figure 19.

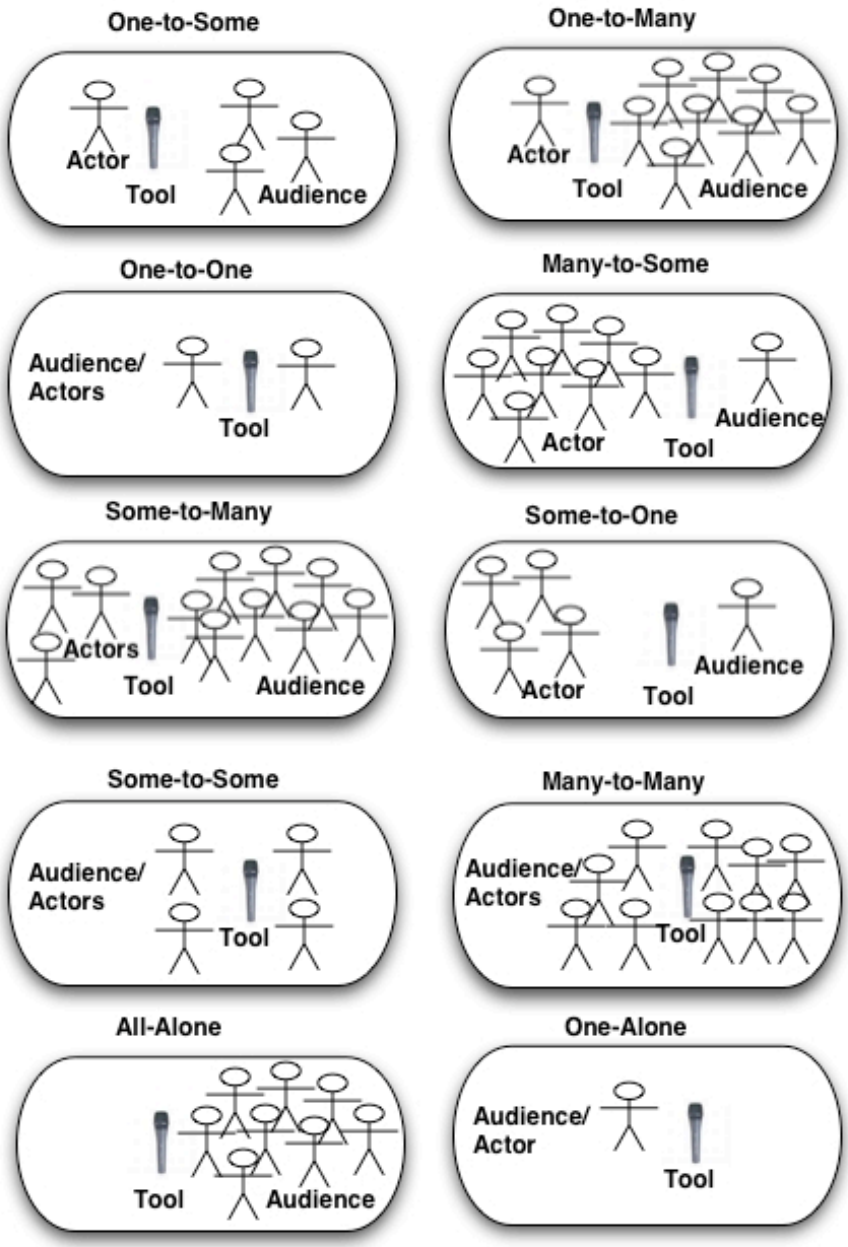


Figure 19: Group settings

Four examples of how roles and group settings change are demonstrated in Figure 20. These collaborative learning patterns are described in detail in Section 2.3.3. In the Think-Pair-Share pattern, learners start by reading a posted problem and writing

their solutions individually. This scene represents the *One-Alone* setting. The next scene consists of discussing the solutions with peers. This is represented by the *One-to-One* setting, in which the group size is two. The third scene represents all learners sharing their solutions with others. This is represented by the *Many-to-Many* setting. In the Jigsaw pattern, there are two scenes that have the Some-to-Some setting. The first one represents the expert group setting and the second one represents the jigsaw group setting. In the Pyramid pattern there are two scenes with different group sizes, for instance the first size might be four while the other might be eight.

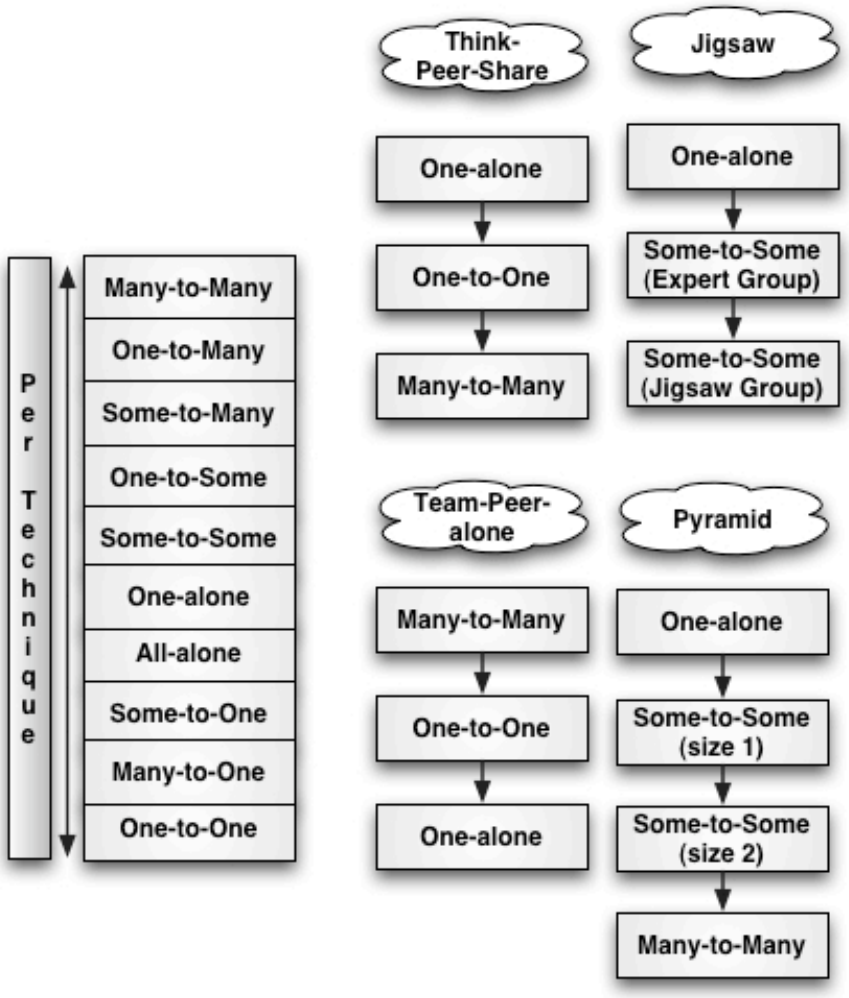


Figure 20: Group setting examples

In the Team-Pair-Share pattern, it is obvious that just by reversing the audience setting, a different technique can be formed (for example, Think-Pair-Share).

According to these various group settings, the Audience settings can be identified as: "individually", "peer", "small group", and "wide group", where the "wide group" represents the entire class. On the other hand, the Actor can be identified as: "none", "facilitator", "certain role", and "all audience". The scripting dimensions are presented in Figure 21. As shown in the figure, the Audience settings have five forms, while the Actor/s have four types. If the actor shows as none, this means that participants are not contributing (for example, reading info, thinking privately, etc.). The facilitator could be the actor, which means that he is the performer in that scene (for example, slide presentation, drawing, etc.). Similarly, a certain role could be the actor. Finally, the "all" means that all group members in that scene will be also the actors (for example, group discussions in which all participants contribute and watch). The Instruction and The Duration are not core components in the script and for simplicity they were deleted from the graph.

This approach of applying three dimensions to each scene (Tools, Actor, Audience) produces more flexibility in scripting. For example, in LAMS (Section 2.2.3) the script used is mainly one dimension (Tools). The COLLAGE moved into the two-dimensional scripting (Tools and Audience). Only Beehive describes the script in three dimensions (refer to Section 5.5 for more elaboration on this). This approach allows for a flexible description of the online collaboration scenario, similar to face-to-face collaboration setting. A full elaboration of this comparison is done in Section

5.5.

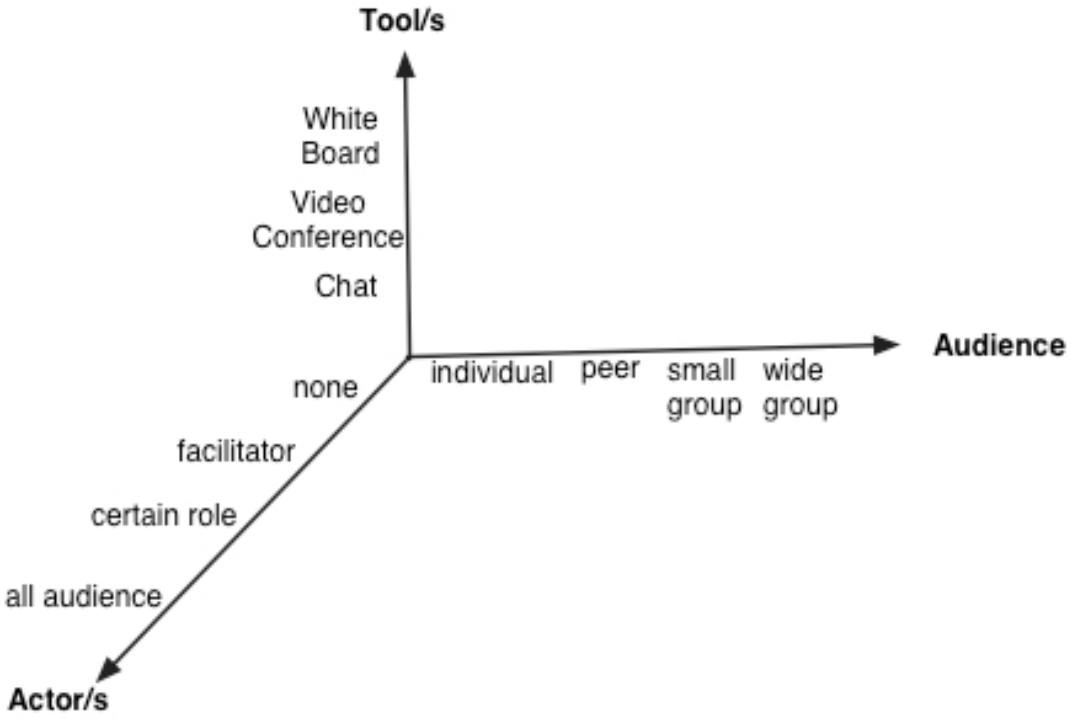


Figure 21: The 3D scripting

Figure 22 presents the play's scenario of the Group Nomination pattern. The play is composed of five scenes. Each scene defines how participants should collaborate.

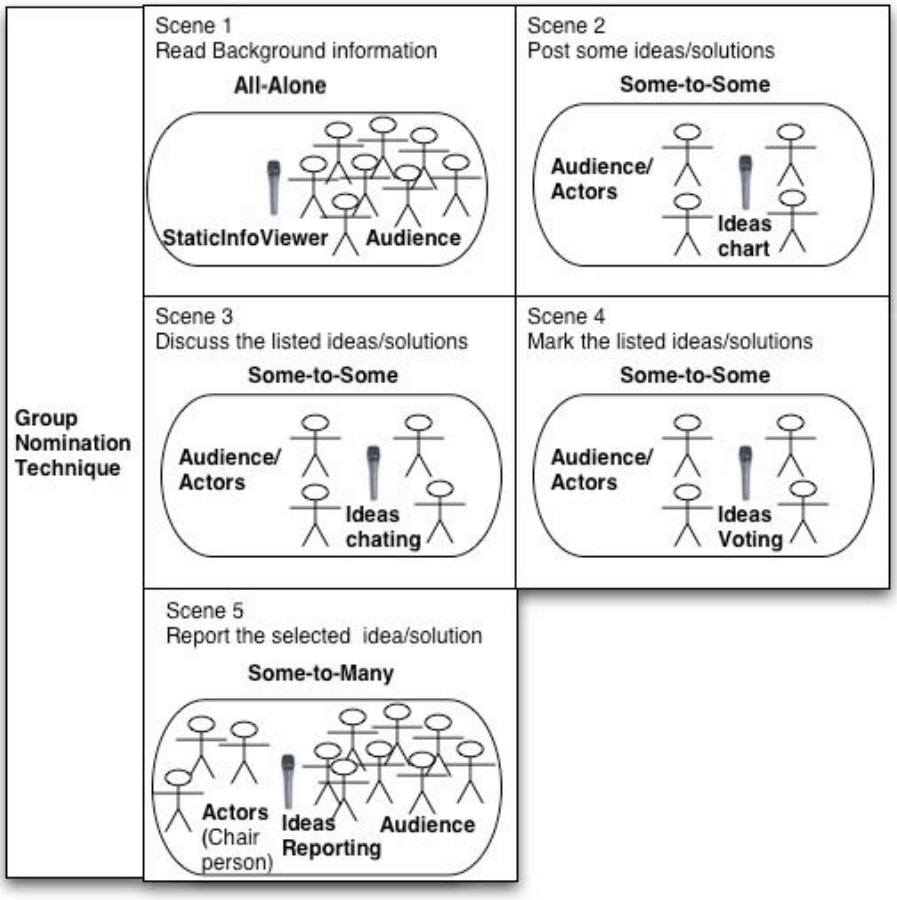


Figure 22: Scene representations for the Group Nomination pattern

An example script of the Group Nomination pattern is presented in Table 7. This formal representation enables systems to facilitate Group Nomination sessions in a systematic way.

Scene	Instruction	Actor	Audience	Tool	Timing
1	Read the problem statement	None	Individually	Static Info Viewer	2 min
2	Post your solutions	All	Small Group	Ideas Chart	5 min
3	Discuss the listed solutions	All	Small Group	Group Discussion	10 min
4	Mark the listed solutions from 1-10	All	Small Group	Idea Voting	1 min
5	Report the selected solution	Chair-Person	Wide Group	Result reporting	1 min

Table 7: The Group Nomination script

In each language there are rules that should be followed to create proper sentences. Similarly in the collaboration scripting, some rules should be followed to produce a proper scripting language. Nevertheless, rules should be kept as simple as possible to enable easy adoption. As shown in Figure 23, the first rule in Beehive is that there are different levels of scripting (none, low, medium, high). The lowest level (no scripting) indicates that the related tasks (Management & Supporting Tasks) are not included in the script and their CSCL components appear all the time during the session. The low level indicates that related tasks (Individual Tasks) are allowed to be sequenced, but they are fixed to certain types of scripting parameters (Actor = "all" and Audience = "individually"). The medium level indicates that related tasks (Feedback &

Assessment and Resource Provision) are partially scripted (some elements in the script are fixed to certain types). The high-level scripting allows full flexibility in defining the Actor and Audience elements.

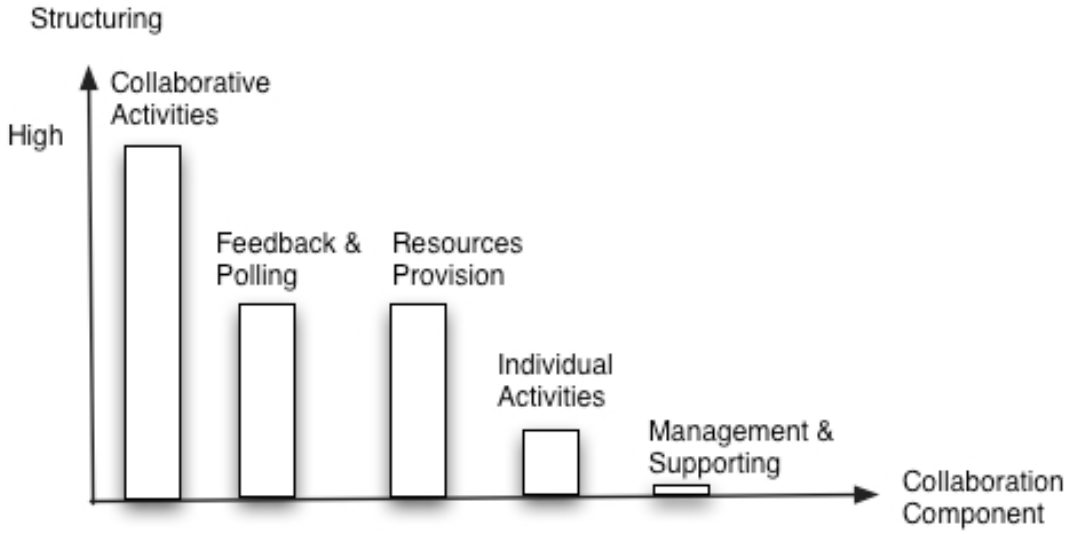


Figure 23: Structuring levels

The second rule in Beehive is that collaboration tasks (for example, Collaborative Drawing, Writing, and Annotating) can step down to be private tasks by changing their Audience element from "small/wide group" to "individually". For instance, the Collaborative Writing Task can be changed to the Individual Reflection Task by simply changing the Audience to "individually". There is a restriction to this rule. Some Collaboration tasks (for example, Presenting, Posting, Discussing, Conferencing, and Icebreaking) cannot have "individually" for their Audience, due to their collaborative nature (cannot be done individually).

The third rule is that the Feedback tasks (Resolution Reporting, Answering Questions, and Survey/Poll Filling) cannot have the "Facilitator" as its Actor since feedback is usually associated with learners, but still he can use audio to guide participants during the session.

The fourth rule is that the Actor in the Resources Provision tasks can be only the facilitator/instructor whereby he provides sessions content as a part of his session design.

4 Beehive Implementation

This chapter explains in detail Beehive's architecture and design elements. Section 4.1 describes the Beehive's main components (the Design Application and the Run-time application). It describes the two platforms (dotLRN and Media Server) that have been used for the implementation of these two components, and it explains the benefits of their integration. Section 4.2 describes all the design aspects of Beehive's authoring environment, and lists the various design options that instructors can follow. Section 4.3 describes all the design aspects of Beehive's run-time environment (components, user interface, synchronisation, etc.). Section 2.6.3 describes Beehive's middleware environment based on IMS LD standards. It points to the current limitations of using IMS LD in representing collaborative learning designs, followed by a description of our proposed IMS LD extensions. Section 4.5 describes the various design elements in Beehive, such as reusability, interoperability, scalability, etc.

4.1 Beehive's Design Architecture

In Beehive, teachers should be able to engage in designing pedagogical collaboration tools according to their collaborative learning schemas. They should be able to prepare and configure the appropriate environment by choosing and configuring a session's parameters, integrating learning materials, and defining the session's progress. As shown in Figure 24, Beehive is divided into two main environments: the authoring environment and the run-time environment. Teachers can include new activities in a course by

selecting and customising stored collaborative techniques. The XML LD schema that represents the design is produced by the authoring environment and is interpreted by the Media Server. The XML LD schema is based on the IMS LD specification (refer to Section 2.6.3). The Run-time application generates all components' instances according to the design's *environment* section and presents them to participants according to the *activity structure* section.

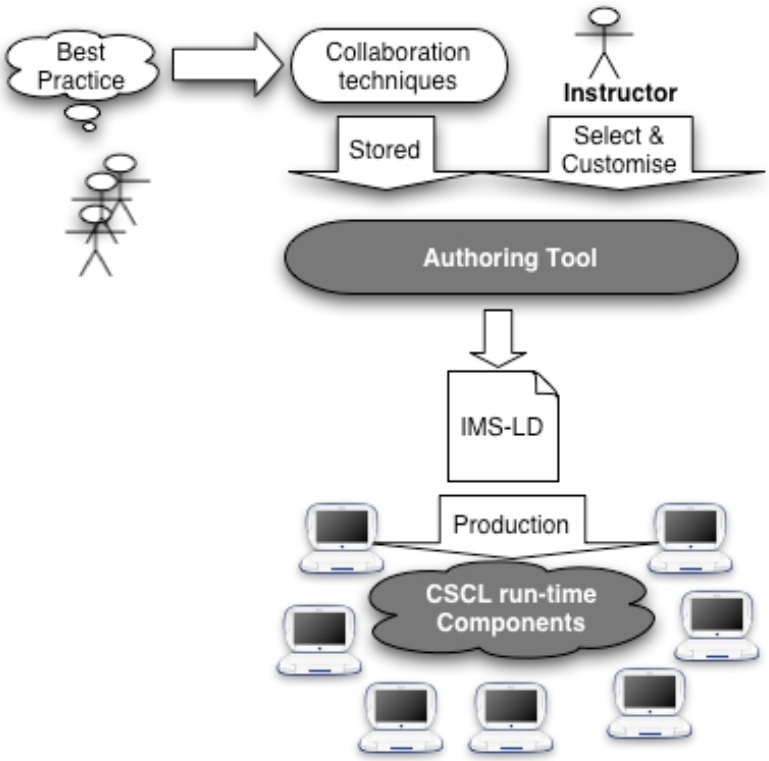


Figure 24: Modelling the designing and implementation process

Our authoring environment is an application within the dotLRN platform. dotLRN is an open-source LMS originally developed by MIT using the openACS [100] web application framework and now managed by the dotLRN Consortium. dotLRN enables us to build highly customisable applications and to use the provided portal framework to install the authoring package. In the dotLRN admin portal, teachers can include new activities in a course by selecting a collaborative pattern already available in

the techniques' repository or by designing new collaborative learning activities if they prefer.

When a teacher finishes an activity design, the design information is stored in the database to be parsed by the run-time environment. Teachers can edit, delete, duplicate (to provide time flexibility), or view the simulations of created sessions. They can also view the sessions' transcripts after learners finish their sessions, as shown in Figure 25.

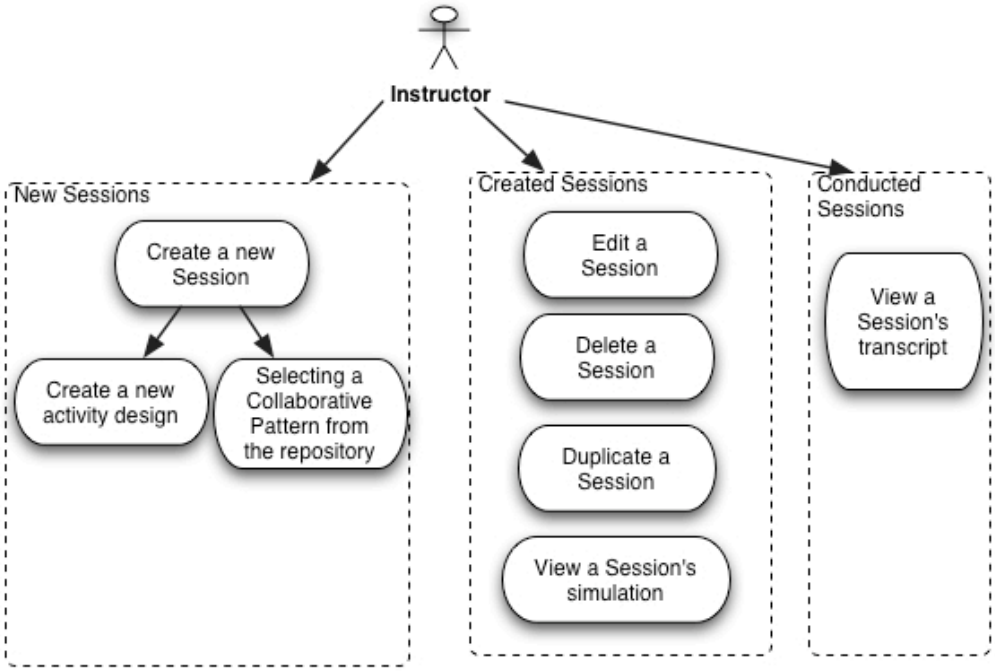


Figure 25: Beehive's design and management options

The run-time environment is implemented by using two applications: Flash MX application and Media Server application [2]. We have chosen the Flash Media Server for several reasons. First, it has a flexible scripting environment (ActionScript), which provides a simple object model with high-level abstractions so users can control the logic on both sides (the client application and the media server application). Second, it contains pre-built components, such as chat, audio, video-conferencing, virtual white board, which are essential in building standard communication applications. It also allows the customisation of the source code of these components to allow the wider

variety needed in our application. Third, the Macromedia Flash player client is already pre-installed on most browsers, so students and teachers do not need to install new plug-ins. The essential part in the Media Server applications is shared objects. The shared object synchronises data between participants in real-time. That has been efficiently used to build a wider range of components needed in the collaborative learning applications, such as Guiding, Asking, Timing, etc.

The XML LD schema, which represents the design, is based on the IMS LD specification.

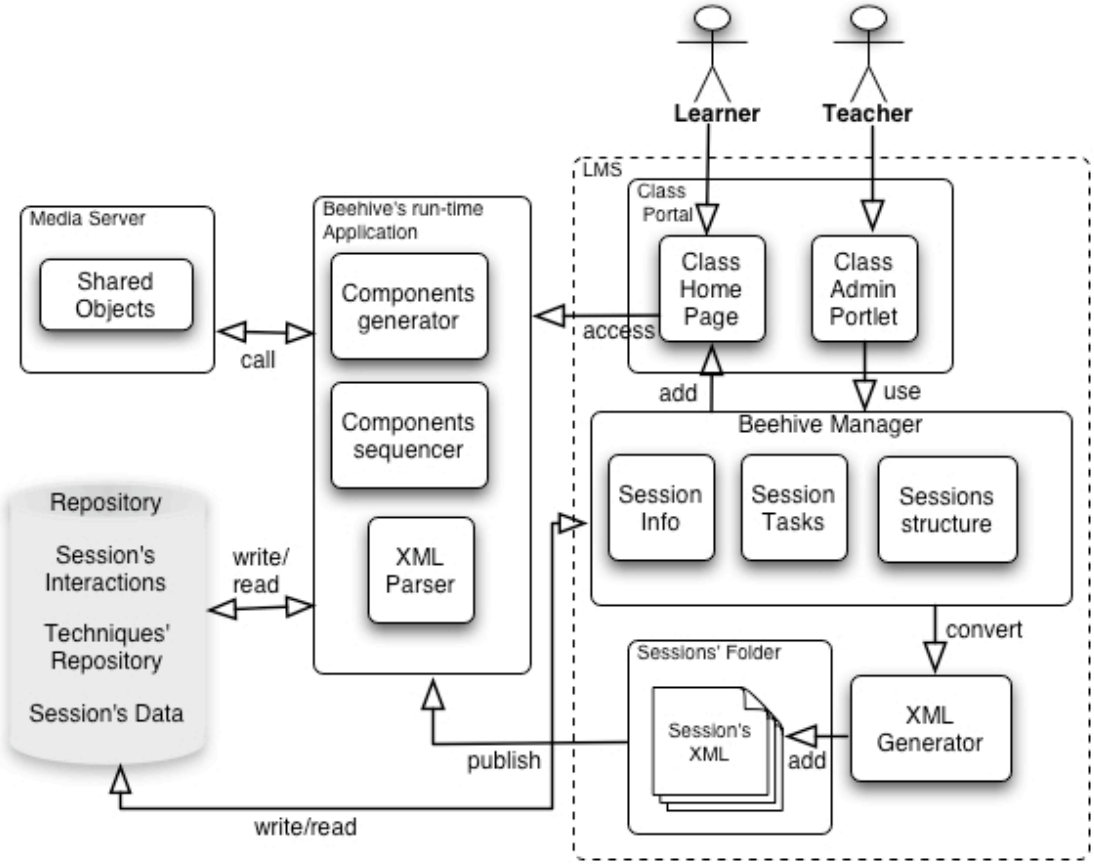


Figure 26: Beehive's components diagram

The information flow in Beehive is described in Figure 26. dotLRN consists of a portal framework, which uses the OpenACS Portal architecture to generate the user's portal interface. The class admin portal allows teachers to access the Beehive manager

page to add new activities to their courses, change, or delete their designs. The Beehive manager includes three main components: The session's info page where teachers can fill in all session related information, such as title, date, description, etc; the session tasks page, from which they can select the required tasks for that session; the session scripting page, where they can sequence the selected tasks according to the activity's scenario.

When a teacher finishes the design, all design information will be stored in the session's data repository while the activity pattern information will be saved inside the technique's repository for reusability. Finally, the activity link will be shown in the class home page, where all links will be listed according to their dates.

The XML Generator module uses the tDom XML parser to publish the XML LD schema. All published XML files will be saved inside the course session folder. The session link points to the "Flash Application" which provides the run-time environment needed for implementing the design. The *user id* and *session id* are also passed to the Flash Application through that link (by using HTTP GET). The Flash Application parses the session's XML and creates all components' instances according to the *environment* section inside the XML, and then it presents these components to viewers according to the *activity structure* section. The Flash Application uses the RTMP protocol to connect to the shared objects inside the Media Server to activate the required synchronous communication components. All activity interactions are saved in Beehive's database.

An assessment phase should follow each session, giving the teacher the opportunity to evaluate the session's outcomes and to identify problems that can help in his future designs. This can be accomplished by using the Beehive's transcript component.

Another central modelling aspect in Beehive's authoring environment is capturing the collaboration script within the scripting page. Figure 27 represents the UML diagram [30] for the proposed collaboration script described in Section 3.5. In this figure the central component is the collaborative activity, whereby the collaboration

script should contain at least one activity. Each collaborative activity is composed of the five components (Actor, Audience, Duration, Instruction, and Environment).

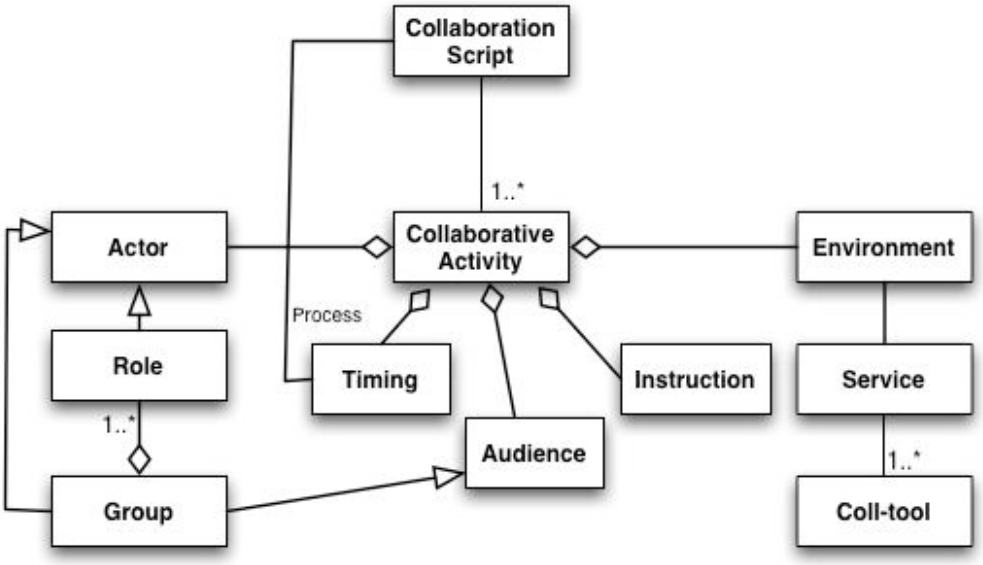


Figure 27: Collaboration script's UML diagram

The Instruction element clarifies each collaborative activity. The Actor is a subtype of Role, whereby groups are composed of at least one role. The audience defines the groups' setting in each collaborative activity. The Tools component is a part of the Environment and defines the tool/s needed in each collaborative activity. Finally, the Timing component in the script represents the recommended duration of each collaborative activity. The timing triggers a transition to the next collaborative activity.

4.2 Authoring Environment

The main objective of the authoring environment is to encourage teachers to engage in designing and reusing collaborative learning designs. Developing CSCL designs is a difficult task for instructors, since these designs follow complex technical specifications. Therefore, the authoring application should be built to be a user-friendly tool. Its interface should be influenced by the collaborative design process in a wizard type that hides many technical details (for example, XML tagging and interpreting). The Beehive's authoring application enables teachers to create pedagogically sound scenarios that can be interpreted by players who are capable of executing these scenarios and setting up the appropriate technological environment. Depending on the learning objectives, what kind of problem students are asked to solve, and the context of the problem, teachers have three design options, presented in Figure 28.

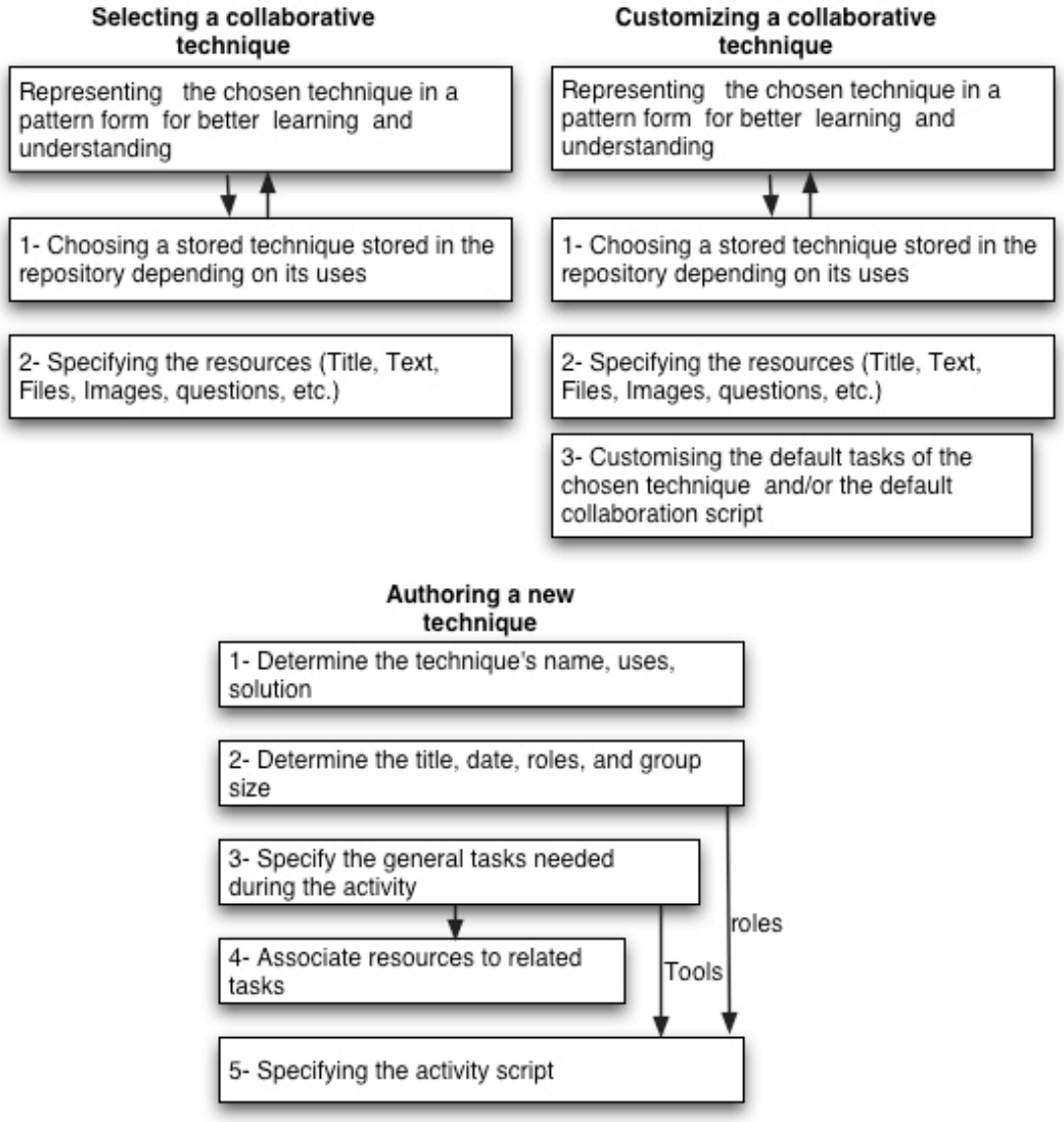


Figure 28: Modelling the three options of authoring process

The first option is that the instructor selects an existing collaborative learning design from the collaborative patterns repository, as shown in Figure 29. All the patterns are listed with their main objectives to provide quick indexing.

Pattern information			Uses
Brainstorm	Create session	View Simulation	Idea creation in a short period
Buzz Group	Create session	View Simulation	A topic group discussion for a short period to gather potential solutions to a problem
Case Studes	Create session	View Simulation	Helping participants to develop skills in identifying concerns, analysing problems, and considering solutions
Debate	Create session	View Simulation	A controversial topic clarification or decision-making
Debate -Three Person Team Debate Format	Create session		
Debate -Two Person Team Debate Format	Create session		
Debate -A typical Oxford format	Create session		
Debate - Modified Oxford format	Create session		
Debate - A typical Cambridge format	Create session		
Group Discussion	Create session	View Simulation	Knowledge-sharing
Group Nomination technique	Create session	View Simulation	Creating and choosing the best ideas/solutions
Jigsaw	Create session	View Simulation	Complex Problem solving
Presentation	Create session	View Simulation	Content that requires presentation
Pro/Contra	Create session		Understanding and presenting different sides views
Problem Based Learning PBL	Create session	View Simulation	to actively involve with problems coming from real practice
Pyramid	Create session	View Simulation	Content that requires individual reflection and then multiple levels of group discussion and explanation
Role Playing	Create session	View Simulation	Training/ Understanding the different aspects of a topic
Round Table discussion	Create session		Knowledge-sharing with a balanced participation
Team Pair Solo	Create session	View Simulation	Content that requires group discussion, peer explanation, and finally individual reflection
Think Pair Share	Create session		Content that requires individual reflection, peer discussion and groups solution-sharing

Figure 29: Beehive's collaborative patterns repository

There are currently 14 collaborative patterns from which to choose. For more information, the instructor can read the pattern representation (as shown in Table 1) to get more information. It is necessary for instructors to be familiar with the structure of the available collaborative techniques in order to plan a feasible design. As shown in

Figure 30, the first step after the instructor selects a certain technique is to fill in the session title and the session starting time, since at synchronous sessions, participants should join and start at the same specified time. The system also presents the default roles and group structure (how students would be grouped) for that technique. For instance, in the Group Nomination Technique, the default group size is five (four participants and one chairperson).

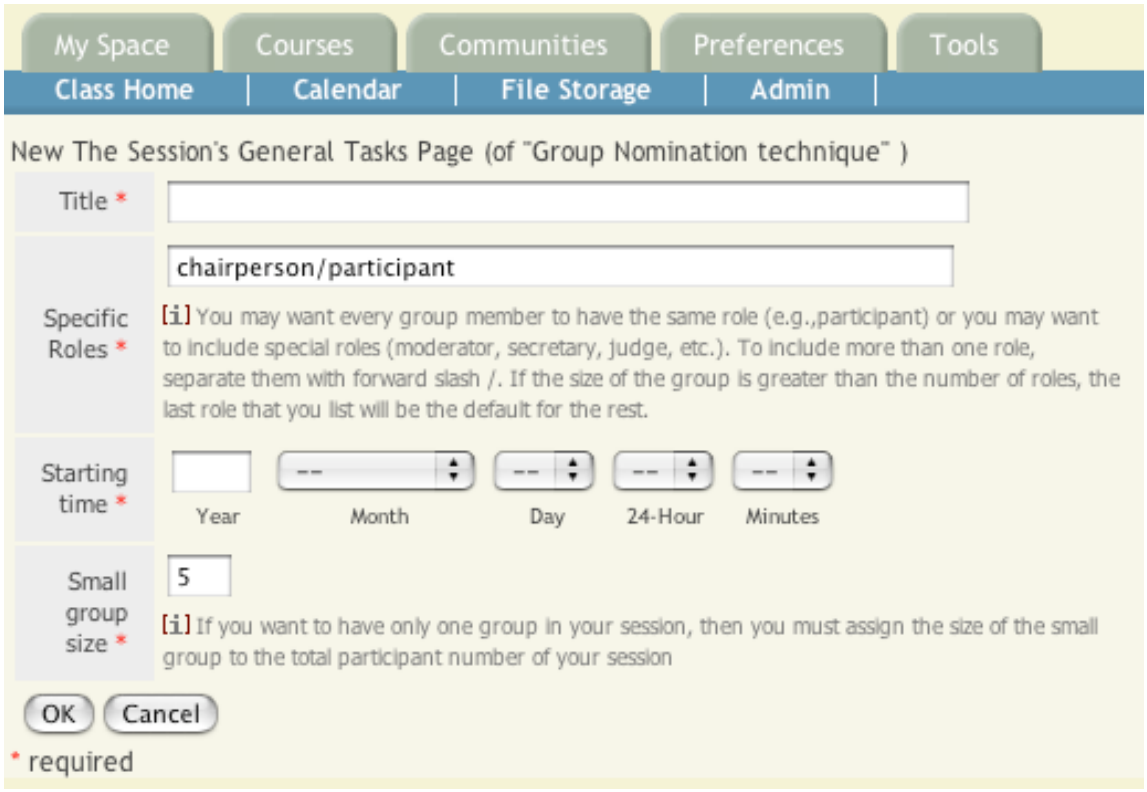


Figure 30: The session's info' page for Group Nomination Technique

The system then presents a list of tasks associated with the technique. Tasks are presented in five categories, as described in Section 3.2. The default tasks for the Group Nomination Technique are presented in Figure 31.

Figure 31: Screenshot of the tasks page of the Group Nomination Technique

The next step for instructors is to couple resources (text, images, files, etc.) to the related tasks. Table 8 presents Task patterns that are related to resources.

Task	Resources type
• Providing Objects/Title • Providing Session's Information • Providing Group Information • Providing Role Information • Providing Session's Scenario • Text Discussion	Text

• Idea Selection • Filling a Survey • Reflecting (Annotating, Filling)	
• Image Annotating • Slide Presentation	Image Slide
• File Upload • File-sharing	File/URL link

Table 8: How resources relate to task patterns

According to the Group Nomination Technique tasks, only one task requires a text resource (Session Info Task), as shown in Figure 31.

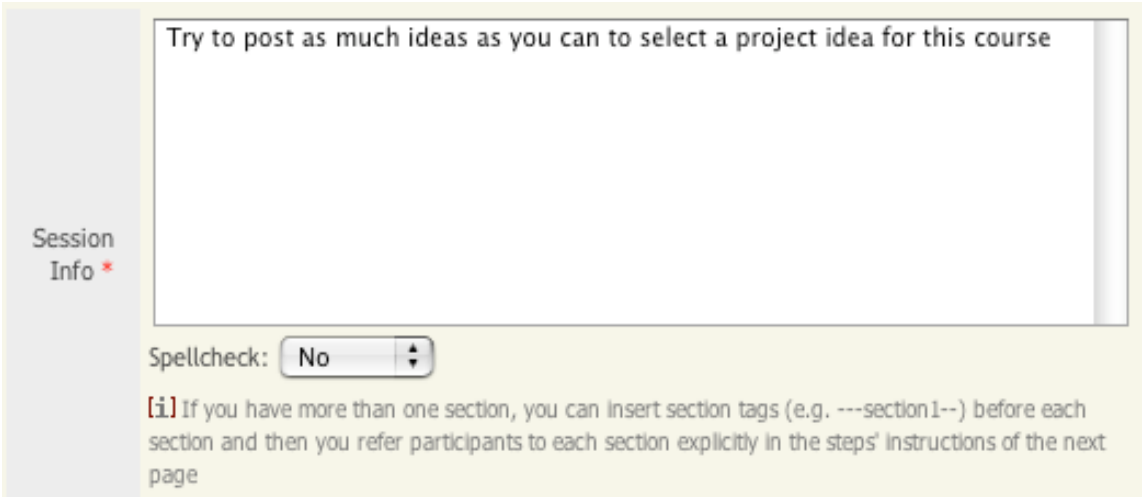


Figure 32: The resource fields of the Group Nomination Technique

The last step is presenting the default script. The Group Nomination Technique default script is shown in Figure 33. At this point the instructor's participation is finalised.

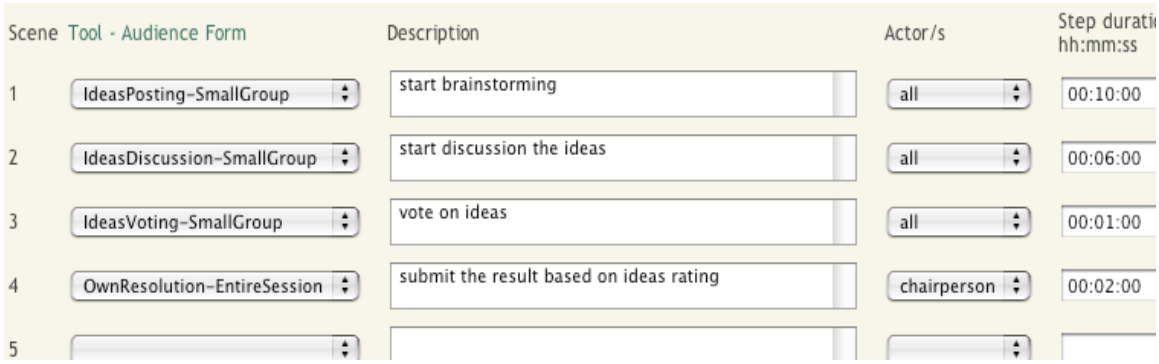


Figure 33: The default script of the Group Nomination Technique

Referring back to Figure 28, instructors have the option to use the exact technique's structure or to customise the technique by changing its default tasks or scripts. Customisation is normally needed, since using collaborative learning techniques is an exploratory process by an instructor, which requires practice, analysis, feedback and continual modification. The third option for the instructor is the ability to design an entirely new collaborative technique. Beehive's authoring application also supports the development of new techniques rather than an exact transfer of techniques. Instructors sometimes prefer to spend time working fluidly with designs rather than picking and choosing. In this option, instructors need to select tasks, define the group structure, associate resources to the related tasks, and to fill out the collaborative flow script.

4.3 Run-time Environment

Figure 34 describes the Beehive's run-time process. The Flash Beehive Application provides the run-time environment needed for the designed activities. First, the Flash Application communicates back to the dotLRN server to obtain other information, including the *session name, facilitator id, and user name*. Then it checks whether the user is the facilitator or learner. Beehive's run-time application grants the facilitator the "floor control" which enables him to control the timer, the guiding, etc. On the other hand, if the user is a learner, then he will be assigned to his role and group according to the grouping method specified in the design. The Flash Application then parses the session's XML LD and loads all components' instances according to the *environment* section inside that XML. Next, it presents the first step's components according to the *activity structure* section. The Flash Application connects to the shared objects within the Media Server to activate the required synchronous communication components.

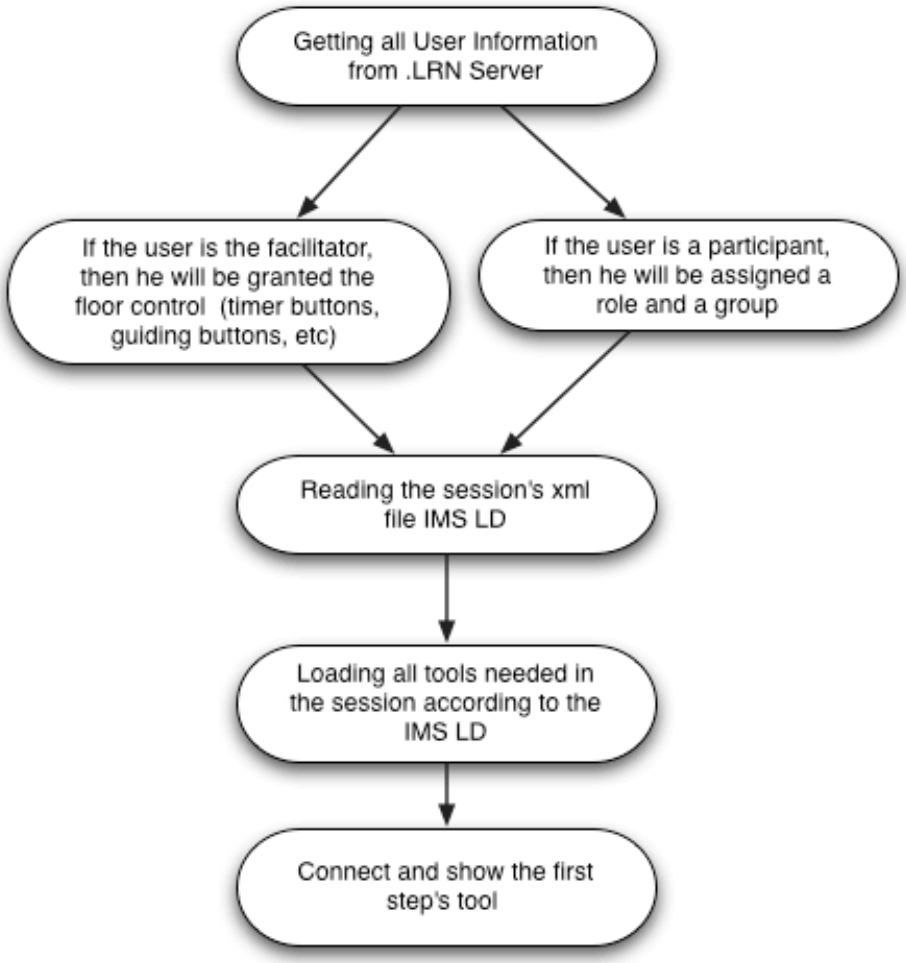


Figure 34: Modelling the run-time setup process

The facilitator is the only one who has the right to advance the session. The first step's components (1st instruction, 1st tool, 1st time) appear on the participants' screen according to the collaboration script. The facilitator timer synchronises the transition of the following steps, as shown in Figure 35. When the time reaches the step's time limit (visible to all participants), the facilitator's timer triggers the timer-shared object on the Media Server. The Media Server sends a request to all clients' views to proceed to the next step. Clients can use the navigation arrows (shown in Figure 37) to go back locally to the previous stage (for example, to review their group's contributions in an earlier stage) but they cannot exceed the current step.

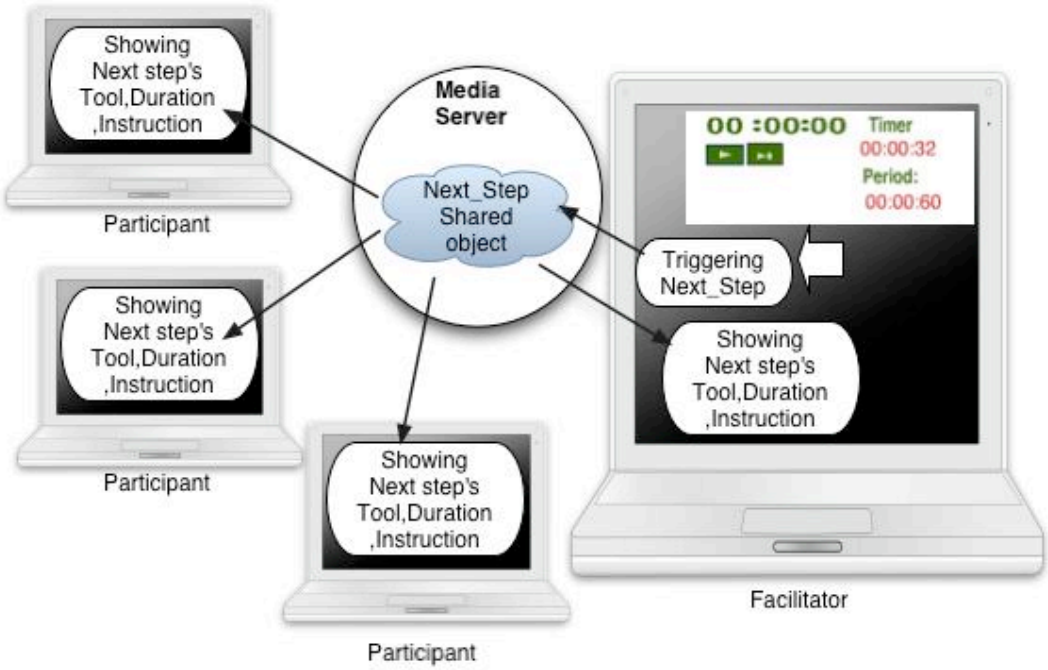


Figure 35: Synchronising the run-time process

Each CSCL component (Chat, Vote, etc.) is linked to a related shared object on the Media Server side. As shown in Figure 36, when the status of the shared component changes, it sends a request to the Media Server to apply the same changes to all clients connected through the same connection channel, according to the Audience setting of that step.

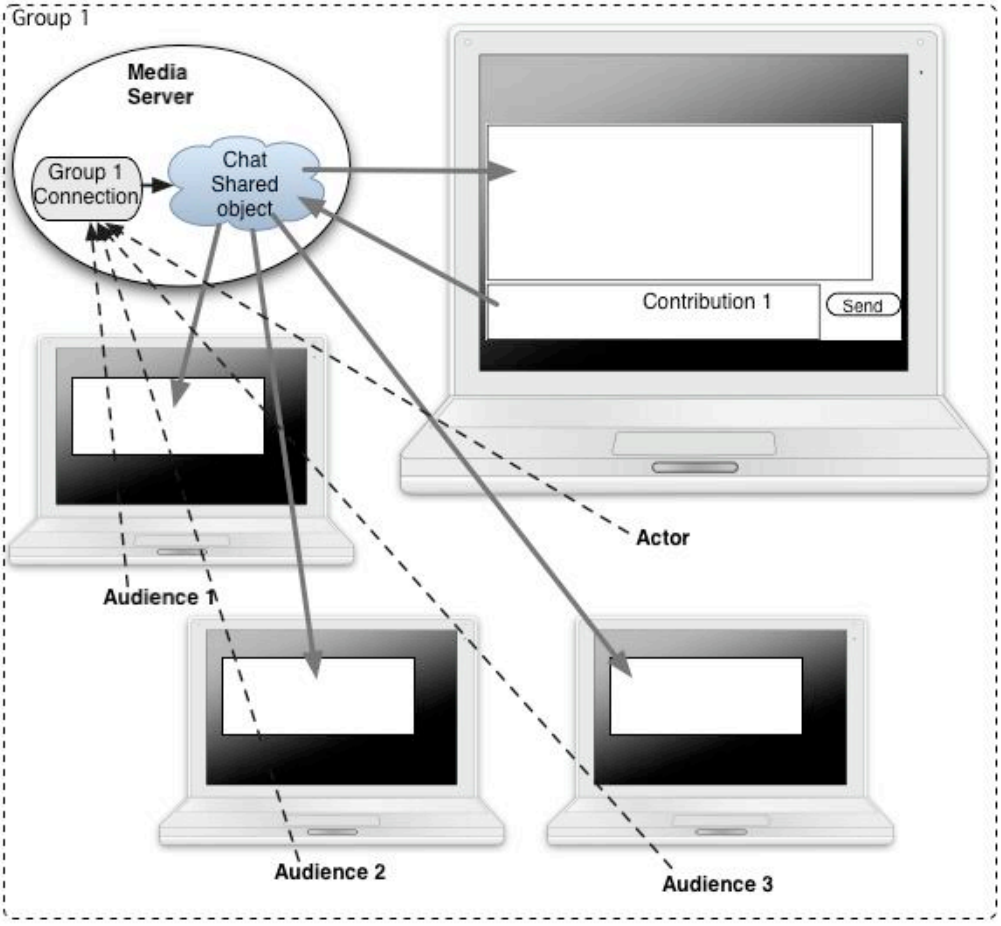


Figure 36: How actors and audience interact with the Chat component

The main screen in Beehive's run-time application is divided into two frames: Background and Stage, as presented in Figure 37. Similar to a play, the Background section (the lower section) contains all supportive elements that constantly appear during the play, while the Stage section (the upper section) contains the main components that actors usually use to perform. The Background section contains all components related to the Support Tasks (Timing, Guiding, Tracking, etc.). All other components appear in the Stage section, according to the session's script. For instance, Figure 37 demonstrates that the Stage contains the Ideas Chart component, according to the first step in the Group Nomination Technique's script.

There are three view representations in the Beehive's run-time application: Facilitator View, Actor View, and Audience View. The Facilitator View allows facilitators to manage and guide the session. For instance, the Timer component view for facilitators contains the control buttons (forward and pause), while participants only view the time remaining of the active step. The Actor View enables an actor to send his contribution (for example, send buttons), while the Audience View only allows the audience to view the actor's contributions.

This type of user interface representation prevents components from being overwhelming and facilitates participants according to their assigned roles.

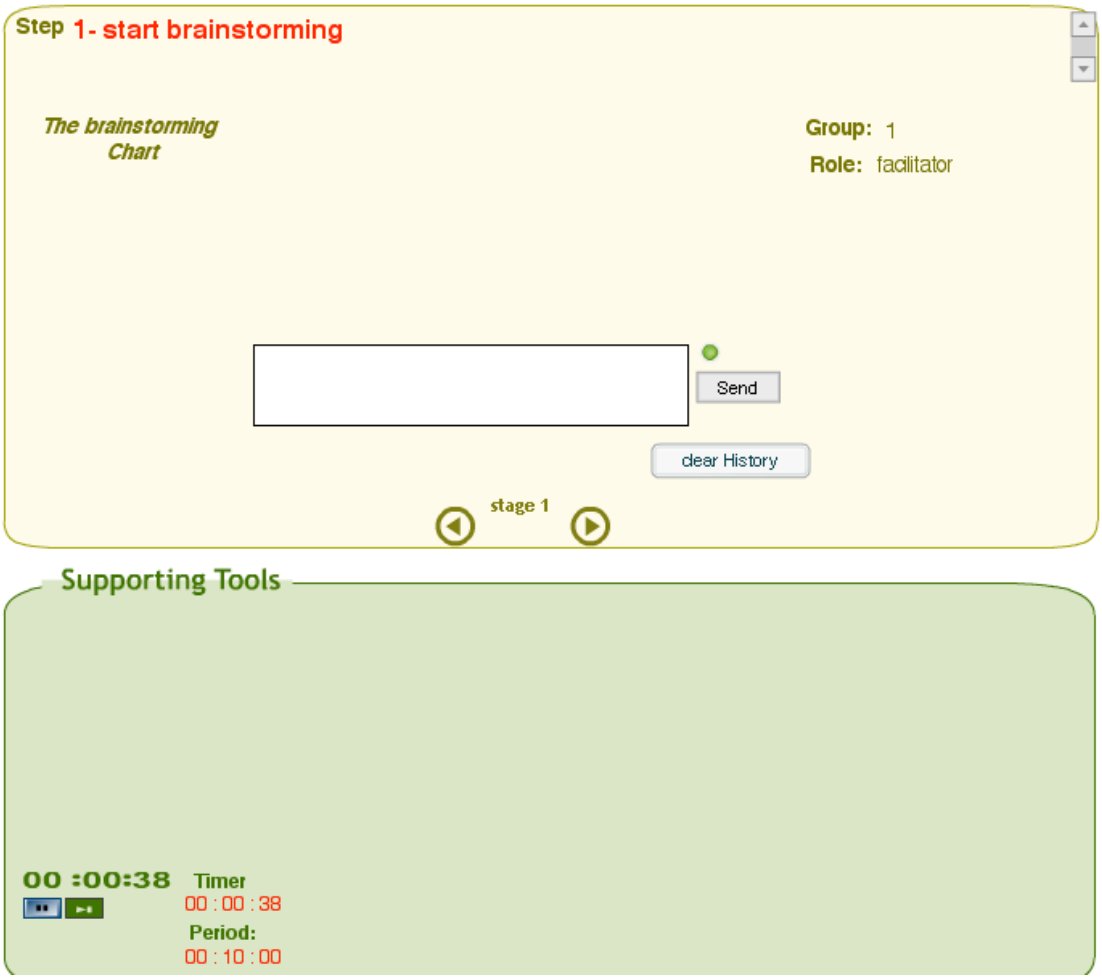


Figure 37: A screenshot of the facilitator's run-time view

4.4 Middleware Environment

We have chosen the IMS LD specification for its flexibility and its ability to describe most learning designs in a formal way. A problem with IMS LD is that it provides no means to specify how a group would interact inside a synchronous collaborative learning session [79]. Here we propose three levels of extensions. The first one is to the IMS LD Service, consisting of a special type called Collaboration Tool Service that defines more tools needed for the collaborative activities. The second one is to the IMS LD activity definition, consisting of a special type called Collaboration Activity, which specifies how a group would interact inside a single collaborative activity step. The third extension is to the IMS LD components definition, consisting of a Group component that specifies the group structure and grouping method.

The Services in IMS LD consist only of Send-mail and Conference services. Effective collaboration needs more than these components. Figure 38 presents the Collaboration Tool Service schema. As discussed in Section 3.3, there are four core components (the Static-Info viewer, Dynamic-Info viewer, Discussion, and Vote). These are needed for implementing most of the collaborative learning patterns. Some additional components (tracking, timing, etc.) are needed to support the session's performance. A Static-Info component is used to hold information that does not change during the session (for example, the title). The Dynamic-Info component is used to hold information that changes during the session (for example, guiding information).

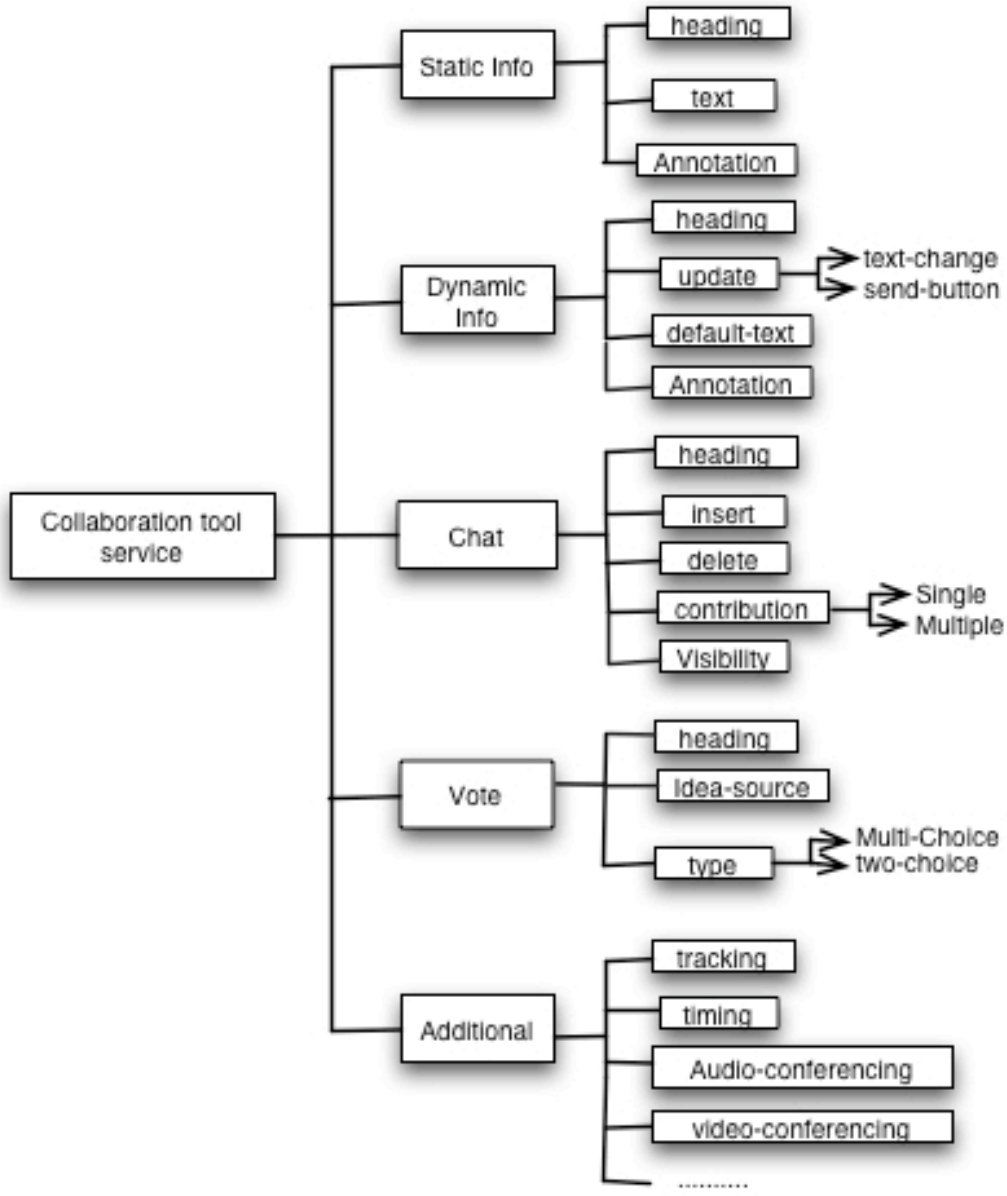


Figure 38: The collaboration tool service schema

For example, a debate scenario over the legality of fox-hunting in Britain might consist of the following tasks. First the teacher's tasks might be to provide a title, the session information, controlling the time, and guidance. Second, the learners' tasks might be discussing different sides and voting on these sides at the end of the discussion. According to this scenario, we need to include:

- Two Static-Info components; one for the title and the other for the session information.
- One Dynamic-Info component for the guiding task, whereby the target is selectable during the run time and is triggered when the facilitator clicks a send button.
- One discussion component is needed for debating, whereby the type of discussion is text-based, and the role of each contributor is inserted before his text contribution.
- One Voting component for the sides voting, whereby the source of the voting question is the facilitator, the type is yes/no voting and again the target is all the group members, which means that voting will happen within each group.
- A Timing component to enable the facilitator to control the timing.

The XML LD schema presenting this part is as follows:

```
<staticinfo>
        <heading> Title </heading>
        <text> Argument for fox-hunting </text>
</staticinfo>
<staticinfo>
        <heading> Session Info </heading>
        <text> Before the breeding season, there are 250,000 foxes in Britain. The
number doubles when the cubs are born, and over the following year it falls back to
250,000. This means 250,000 foxes die each year. The 250,000 fox deaths each year
include deaths from natural causes, road kills and about 100,000 killed by shooting and
snaring.
        </text>
        <annotating>”false”</annotating>
</staticinfo>
<dynamicinfo>
         <heading>Guiding</heading>
        <default-text>”none”</default-text>
```

```
        <updating-method>”sendButton”</updating-method>
        <annotating>”false”</annotating>
</dynamicinfo>
<chat>
        <heading> Arguments for fox hunting </heading>
        <contribution-type>”multiple”</contribution-type>
        <contribution-visibility>”always”</contribution-visibility>
        <insert>”role”</insert>
        <delete>”none”</delete>
</chat>
<vote>
        <voting-type>”two-choice”</voting-type>
        <heading>Sides Voting</heading>
        <ideas-source>”instructor”</ideas-source>
        <ideas-text>Do you think that fox-hunting should be illegal?</ideas-text>
</vote>
<timing>
    <owner>”facilitator”</owner>
</timing>
```

The second proposed extension is the Collaboration Activity, as shown in Figure 39. It describes the collaboration script in a formal way, as discussed in Section 3.5. For example, the debating scenario of the previous example might look like this [54]:

```
<collaboration-activity identifier="step 1">
      <coll-tool-service-ref ref="discussion"/>
        <actor roleref="proposer"/>
        <instruction>Propose the motion</instruction>
        <imsld:time-limit>000500</time-limit>
        <audience>"small group”</audience>
</collaboration-activity>
```

```
<collaboration-activity identifier="step 2">
          <coll-tool-service-ref ref="discussion"/>
          <actor roleref="seconder(p)"/>
          <instruction>Second the Proposer</instruction>
       <time-limit>000300</time-limit>
       <audience>"small group”</audience>
</collaboration-activity>
.
.
<collaboration-activity identifier="step 8">
          <coll-tool-service-ref ref=" vote"/>
          <role roleref="all"/>
          <instruction>answer  the question</imsld:instruction>
        <time-limit>000100</imsld:time-limit>
        <audience>"small group”</audience>
</collaboration-activity>
```

According to this part of the schema, the first step's instruction is "Propose the motion", which will be shown to all participants in each group. Only participants with the "proposer" role can send information via the discussion tool for a specific amount of time. The second step is for the second proposer to second the proposer. The last step is for all to vote, and they will instantly see the voting results. This script structure can be shared and reused in other debate activities in different domains by simply inserting new text in the text elements (title, info, and voting question text). For example, the previous debate structure can be reused in discussing the issue of illegal workers in the USA and their impact on the USA economy in the working class.

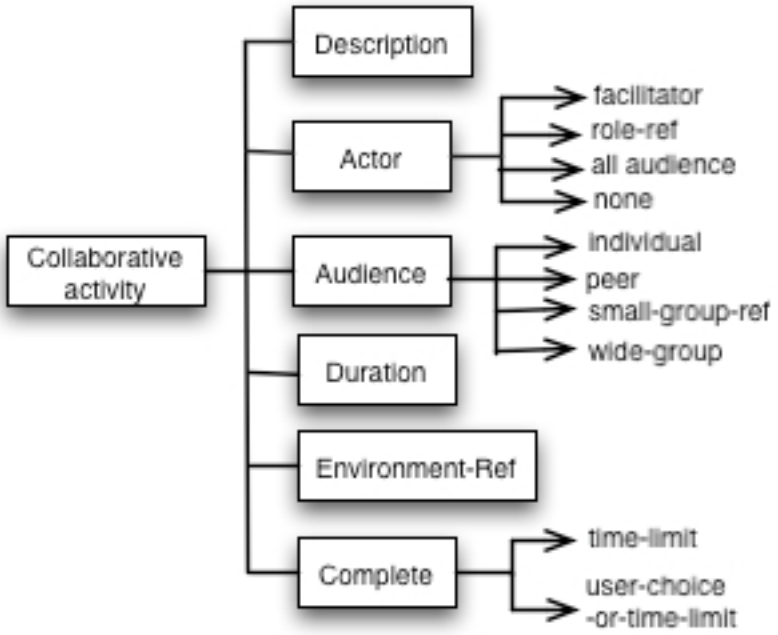


Figure 39: The collaborative activity schema

Figure 40 shows the last proposed extension, where it describes the group structure and grouping method. For the previous debate example, the group structure might be as follows: Size: 10 participants. Structure: proposer, second proposer, opposer, second opposer, speakers (six). The XML LD schema presenting this part is as follows:

```
<groups identifier="Debate">
        <participant roleref="proposer"/>
        <participant roleref="seconder(p)"/>
        <participant roleref="opposer"/>
        <participant roleref="seconder(o)"/>
        <participant roleref="speakers"/>
        <default-participant roleref="speakers"/>
        <size maxnumber="10"  minnumber="10" defaultnumber="10"/>
        <imsld:grouping-method>”random”</imsld:grouping-method>
</imsld:groups>
```

According to the above schema, the system will assign participants to roles on the basis of first come first served. So, for example, the system will assign the first student as a proposer, then the second as a seconder proposer and so on. If roles are fewer than the size of the group then the system will assign the rest according to the default role, which is in this example the speaker.

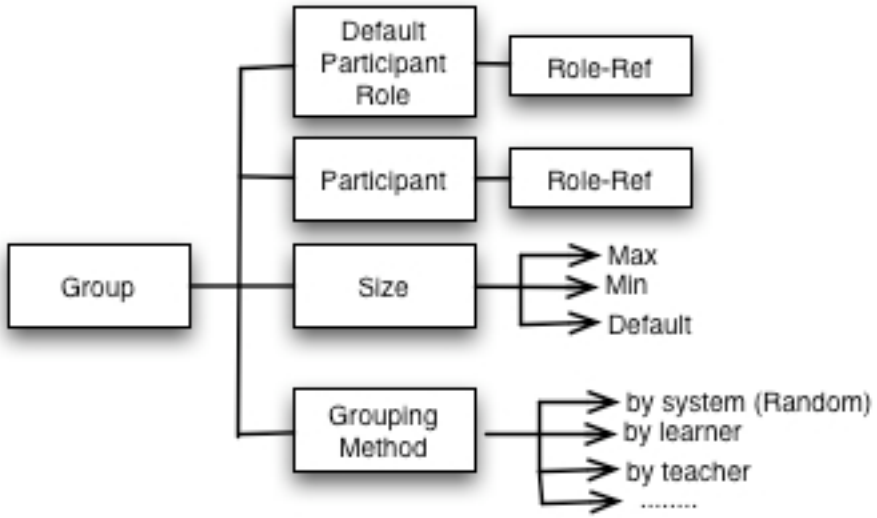


Figure 40: The group structure schema

4.5 Beehive's Design Elements

A number of important design elements that are common in application frameworks, such as Modularity, Design Reusability, Integration, Interoperability, Scalability, etc. are included in Beehive's design. Modularity and Design Reusability were discussed in detail in the previous sections of this chapter, while the rest of the design elements are discussed in the following sections.

4.5.1 Integration and Interoperability

In Beehive, the integration between the dotLRN and Media Server provided many advantages. It has allowed designers to benefits from both systems' functionalities. Beehive reuses many components in dotLRN and Media Server. dotLRN provides the user management functionalities and the asynchronous tools (Forum, Calendar, etc.), while Media Server provides synchronous tools. Even though these asynchronous tools are not a part of Beehive, designers can still use the Outside Task pattern in Beehive to point participants to these asynchronous tools (for example, "use the dotLRN forum to post your findings").

System interoperability is important to allow more flexibility in system selection. Other users should be allowed to use their LMS systems as long as they follow the standards. For instance, Beehive is fully integrated with both dotLRN and Sakai [112].

In Beehive, the interoperability between the authoring system and run-time system is carried out by IMS LD. The communication between the two systems is accomplished in a similar way to the AICC guidelines that describe the data communication between an LMS and a lesson. These data usually describes the lesson status, student score, student information, etc. The AICC LMS Adapter uses the HACP (HTTP AICC Communication Protocol) for launching learning content and exchanging data with the LMS by a simple HTTP Post. We have defined similar guidelines for launching collaborative learning sessions and exchanging data between the Beehive's

authoring application and Beehive's run-time application. We propose the following protocol that defines the communication mechanism:

1. When the LMS creates the session's link, it passes with it the *session's id*, user's information (*user id, user role*), the LMS URL (for example, dotLRN Address), the Media server URL (for example, Flash Media server), the published XML folder path, and the resource files folder path.
2. When the facilitator activates the session, the session status in the LMS database is changed to "active", allowing participants to access the session.
3. When the student, for example, posts an idea in a Brainstorming session, the idea will be stored in the LMS database along with the other information (*course Id, student Id, session Id, interaction time,* and *interaction type*). These interaction data can be accessed through the LMS transcript for further assessment.
4. When students finish the session, the session status changes to "completed".

The login is undertaken through LMS where it handles the login process. All login data are encrypted in the LMS database. The sessions' data are considered to be not highly sensitive, so using the cookie management to verify user's identity is considered to be sufficient at this stage. The security level can be raised by simply re-entering user login information on the Media Server side.

4.5.2 Scalability

To provide scalability, Beehive recommends a four-layer repository, as shown in Figure 41. The top layer provides instructors with a pool of collaborative learning patterns from which to choose from. As mentioned earlier, there are 14 collaborative learning patterns. For more scalability, instructors can develop new collaborative patterns. All new patterns are stored for later retrieval and reuse. This scalability can be taken a step further by providing a central repository through which a larger community of instructors can share and reuse their designs. As described in Section 3.2, 29 task patterns are currently stored in the second layer. Instructional designers can identify

new tasks and then map them to the appropriate software components, along with any configurations, according to Section 3.3. For instance, when we added the Text Discussion Task Pattern to our framework, we selected three software components out of the 12 components, Text Viewer, Audio, and Text Chat, which are required to support texts that need to be discussed part-by-part. The Text Viewer is used to contain the text, the Text Chat is used for discussion, and the Audio is used to provide discussion flexibility. Additional configurations were also made. First, the Text Viewer Annotation parameter is set, to permit annotation during discussion. Second, the Text Chat four parameters are set as following: *Insert= "none", Delete = "none", Contribution Type = "multiple", and Contribution Visibility = "always"*. Instructors can use this new task to design even a wider range of collaborative patterns.

Software components are stored in the third layer, where there are currently 12 software components (refer to Section 3.3). Developers can still create new software components by collaborating with instructional designers. These software components can be reused to assemble new tasks that can in turn be used to assemble new collaborative patterns.

The lowest layer represents the interaction storage, where all sessions' run-time data and interactions are stored to be presented later in the session's transcript, as shown in Figure 42. Session transcript can be used as a piece of artifact that both the session's assessor and participants can review after the session's termination.

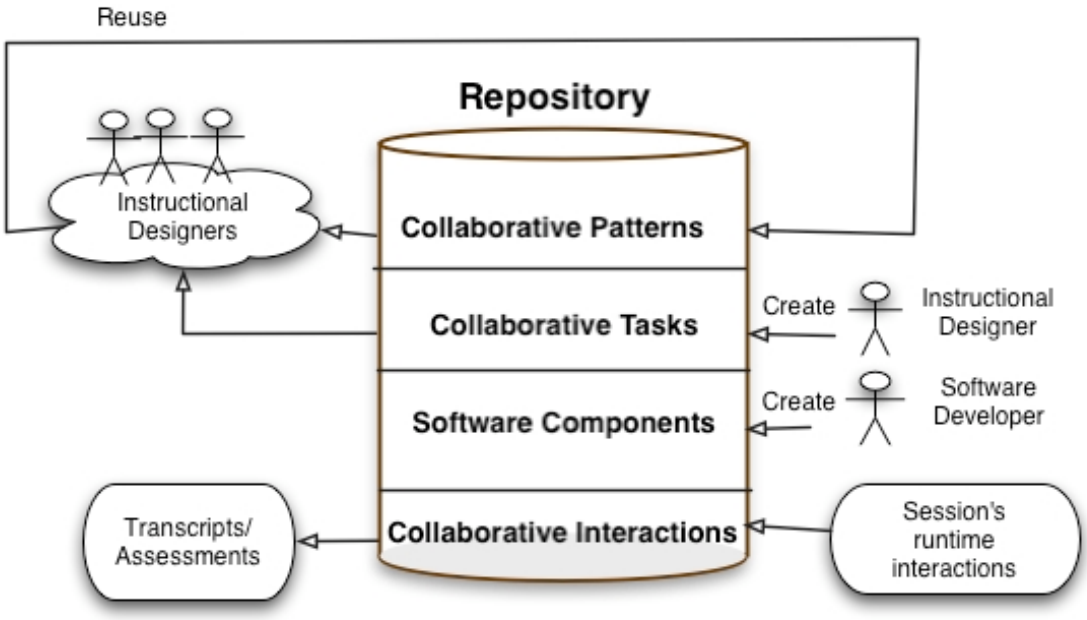


Figure 41: The 4-layer repository

The first, second and last layer are usually located at the authoring environment, while the third layer is usually located at the run-time environment, where all software components are contained within its library.

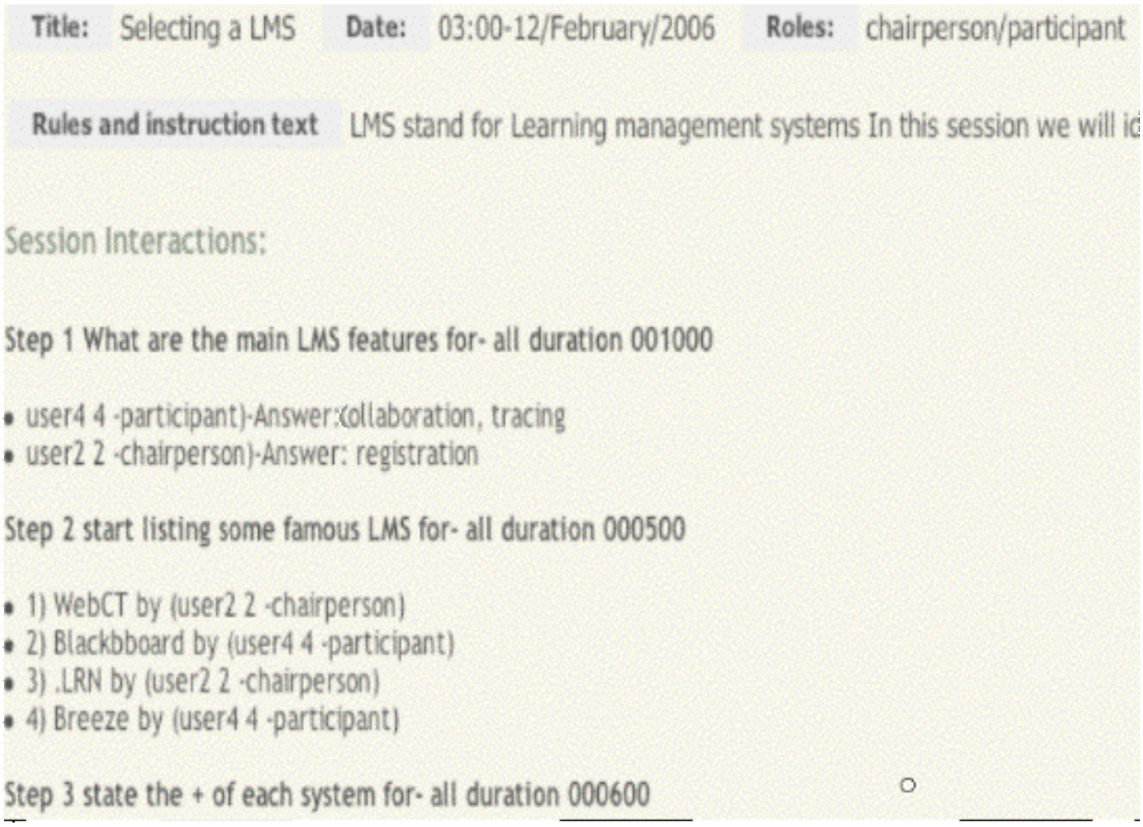


Figure 42: Screenshot of a session's transcript

The Beehive's authoring environment also provides a wider flexibility to designers in terms of working together. As shown in Figure 43, Beehive enables not only instructors to engage in designing, but also learners. Constructivist theory points out that learners need some space and liberty to formulate goals and make errors. Otherwise they will not develop general problem-solving skills [72]. Beehive allows learners to construct their learning process in a very flexible way. A separate collaboration session can be used for this purpose, in which learners can meet to plan for their next session strategy. They can appoint the chairperson, for example, to implement the design on the Beehive's authoring application by granting the chairperson the facilitator's rights.

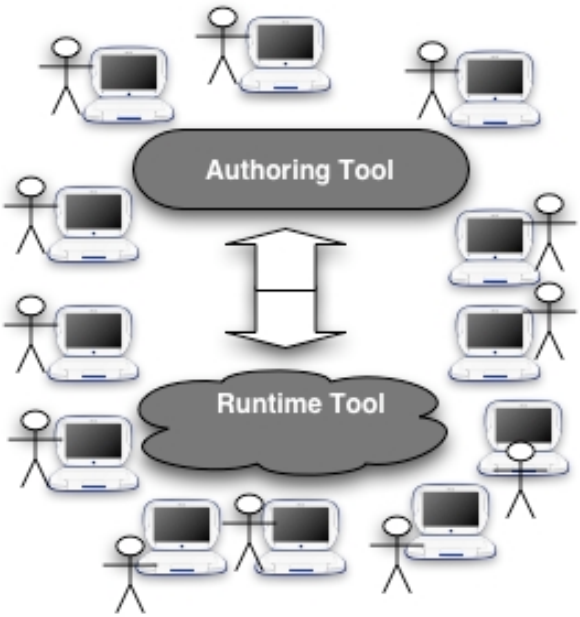


Figure 43: Collaborative design process

Due to recent advances in mobile technology, Beehive could have even more scalability. The advances and widespread adoption of mobile systems can provide remote users with more flexibility to access collaborative learning sessions. Beehive can effectively be deployed in new mobile systems (since flash players are preinstalled in most browsers) by just customizing the front-end interface.

4.5.3 Simulation

The simulator component in Beehive enables instructors to move in both sides (Design and Run-time). A simulated execution of the specified collaboration process can give the designer more profound feedback on what things can work and what does not. Running the simulation according to information on a session's sequence produces artifacts before applying the whole design to a real experiment [90].

The Beehive simulator is a Flash application that reads the same session's IMS LD schema and generates a similar run-time environment. The simulator application communicates with the Media Server to generate all required components' instances and

couples them with the resources, as specified in the schema. The primary extension in the simulator application is that all components are related to mini movie-clips that simulate how users might interact within these components. When a specific component is shown during a certain stage (for example, Idea Chart), a recorded movie clip that statically emulates how a learner sends his ideas will be played. Also, the timeframe of each stage is shortened in the simulation to represent a quicker review of the whole process.

4.5.4 Reporting the Design's Graduate Attributes

As discussed earlier in Section 2.1.1, today's knowledge-based economies are looking for people who can think critically and strategically to solve problems. Graduate attributes are something the organisational sector is looking for when employing new graduates. Many researchers have indicated that collaborative learning generates a rich environment of graduate attributes [68].

There are several approaches to identify what type of attributes are embedded in curricula and courses. The first approach is evaluating learners' own perceptions on what types of graduate attributes they think the course has. The second approach is consulting the course designers on what types of graduate attributes they think their course enhances. The limitation of these approaches is that there maybe a difference between what designers hope and what actually happens [68]. Since in Beehive the collaborative learning design is exposed and formalised, collaboration is not viewed as a black box any more. Collaboration designs can be analysed to identify the collaborative component types to gain a better understanding of the developed graduate attributes. Formalising collaborative learning designs enables systems to map certain design components to certain attributes in a systematic way. For example, the Discussion task usually enhances critical thinking skills, while the Idea Listing task may enhance creative thinking skills (refer to Section 2.3.3).

Beehive can generate a report on what type of graduate attributes the collaborative design might have. This allows instructors to adjust their designs to include

the desired attributes before deploying them. Beehive can currently identify 11 graduate attributes. These attributes are called Collaboration Attributes since they relate mainly to collaborative tasks. They are categorised into three categories: Communication Skills, Higher Thinking Skills, and Social Interaction Skills.

Figure 44 shows the six identified graduate attributes related to the Group Nomination Technique:

- *Written communication* (related to the Ideas Discussion)
- *Creative thinking* (related to the Ideas Posting)
- *Analytical thinking* (related to the Ideas Discussion)
- *Reflecting* (related to the Ideas Voting)
- *Time management* (related to the Timer)
- *Decision-making* (related to the Idea Selection)

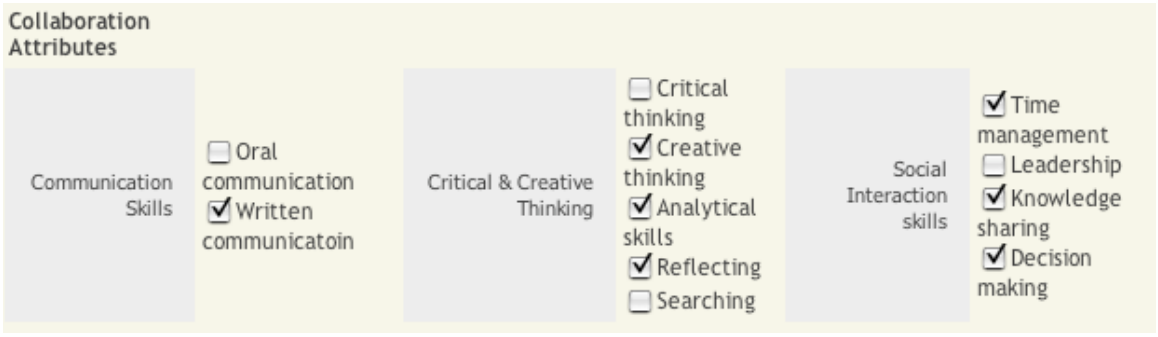


Figure 44: The collaboration attribute in the Group Nomination Technique

5 Beehive Evaluation

The process of developing and then evaluating a CSCL system is challenging. Evaluating outcomes associated with CSCL is challenging due to various reasons, including interface usability, coordinating groups of users, learning outcomes, collaboration attributes, etc. [93].

We followed a multi-faceted evaluation approach to evaluate whether our proposed pedagogical application framework can actually *assist software developers in developing computer systems* that can be used to *support instructors in creating and reusing pedagogical patterns,* and flexible enough *to enable instructors to design a wide range of collaborative activities* that involve *small numbers working on complex tasks*. This approach incorporates both quantitative and qualitative methods [99]. These methods include interviews, surveys, focus groups, direct observation, system logs, and software usability analysis.

The following sections apply the multi-faceted evaluation according to the above order, in which each piece contributes a various perspective to the overall evaluation. Section 5.1 describes how Beehive *assists software developers* in minimising their efforts to create a wide range of pedagogical collaboration tools based on software components' reusability. Section 5.2 presents how Beehive has guided some designers in *reusing some of the pedagogical patterns* within different contexts. Then it presents how it manages to support designers in *creating* an even wider range of pedagogical patterns. Section 5.3 presents three case evaluations that demonstrate how different

users within different contexts were successfully engaged in defining and *designing their collaborative activity* sessions according to particular goals. Section 5.4 evaluates users' (designers/learners) perceptions and experiences involving *small numbers working in complex tasks* by using both feedback and a survey. Section 5.5 evaluates Beehive against other CSCL applications to demonstrate how it is different from the others in approaching and implementing pedagogical designs.

5.1 Evaluating the Pedagogical Collaboration Tools' Development

In order to evaluate whether Beehive has benefited developers in minimising their efforts to create a wide range of pedagogical collaboration tools, the resultant CSCL application should meet the reusability criteria on the following levels:

- Reusability of CSCL Components in a larger set of pedagogical collaboration tools.
- Reusability of Software Components in a larger set of CSCL Components.

The reusability of CSCL components was discussed in detail in Section 3.3, in which 29 CSCL components (that furnish 29 tasks) were used to assemble a larger set of pedagogical collaboration tools. The Beehive repository currently contains 16 pedagogical collaboration tools that support 16 different pedagogical patterns. To test the scalability and the flexibility, more patterns were modelled whereby some patterns have different versions. For instance, we modelled four different types of face-to-face Debate Pattern (Two-Person-Team, Three-Person-Team, Oxford-Debate-Format, and Cambridge-Debate-Format) and included them all in the repository. In addition to that, 21 "free-style" designs were implemented during the course of this project, and some of these designs are presented in the following sections.

This has benefited software developers in reducing their efforts since the number and the complexity level of these CSCL Component is *far less* than the potential number of specialised pedagogical collaboration tools.

The Software Components suggested in Beehive are also fewer than the CSCL Components, in which 12 Software Components are used to assemble 29 CSCL components, as presented in Section 3.3.

To illustrate how new CSCL Components and Tools might be developed by *reusing* some of the already existing Software Components, we present the following case. When we added the Text Discussion Task Pattern at a later stage of this project, we used the pattern's description as a guideline for the implementation of that particular CSCL Component. We decided to reuse some of the existing Software Components to support that task (Text Viewer, Audio, Text Chat), as shown in Figure 45. This new CSCL Component has been reused in assembling the following pedagogical collaboration tools: Case Study tool, and Think-Pair-Share tool.

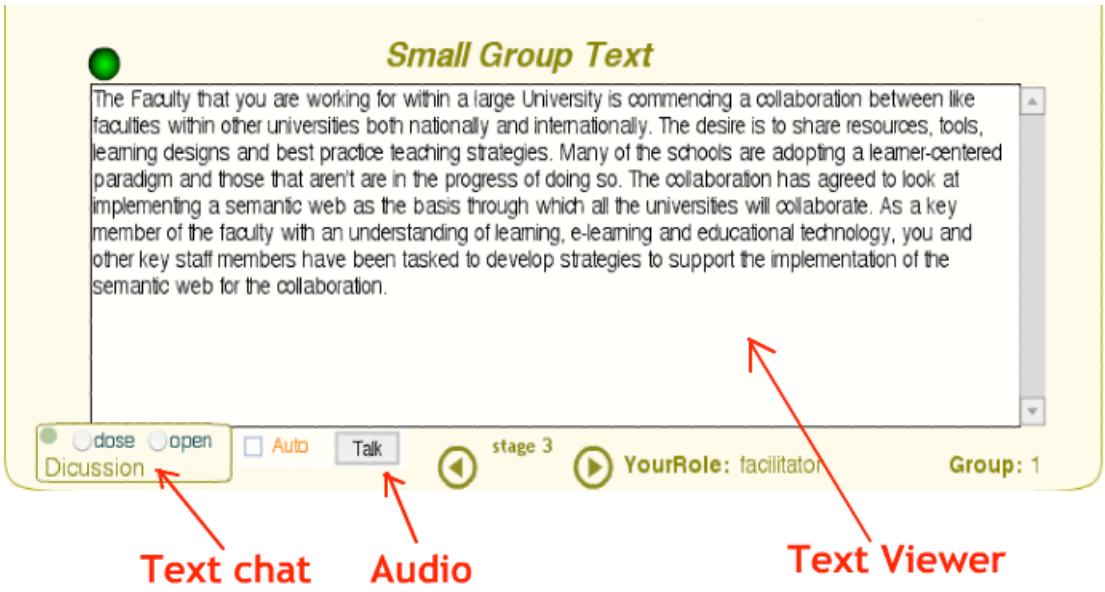


Figure 45: The Text Discussion Component

Eventually some new tasks may require developing new Software Components according to their own specifications. The current components suggested in Beehive cannot handle all techniques and designs, however, since they are based on pattern concepts, they can support a considerable wide range of synchronous collaborative

learning situations. Also a detailed comparison against other systems was done in Section 5.5 that describes the available design elements in Beehive.

5.2 Evaluating Beehive's Patterns Reusability and Scalability

As specified earlier, one of Beehive's main objectives was to support instructors in creating and reusing pedagogical patterns. In order to evaluate this objective, we used the *practical experimental* and the *observation* methods [58].

We selected the *observation* method in order to observe how designers use Beehive's pattern repository for reusability. In a couple of instances, the Group Nomination Pattern was used but in different contexts. As described in Section 2.3.3, the Group Nomination Pattern is normally used to generate many ideas and then to select the best idea among them. In the first instance, one of the users chose this pattern to enable his students to select a project idea for a website they intend to built at the end of a course. In the second instance, another user selected the same pattern, but in a different discipline. It had been used by a group of participants to list and choose the most appropriate LMS for a sport-training platform. In both cases designers had chosen the same pattern's default tasks and structure but in different contexts. Only the second designer added an extra task, requiring learners to read background information on LMS. The first user had prior background knowledge on the Group Nomination Technique while the second user had to read the Group Nomination Pattern's information before selecting and using that technique.

To evaluate whether Beehive is capable of developing new pedagogical patterns, we designed synthetic tests. We conducted two tests to verify whether Beehive can successfully model and implement new pedagogical patterns that already exist in the both environments (face-to-face and online).

For the online we selected two patterns: the "What is … ?" Pattern [86], and the PBL Pattern [73]. For the first pattern, we followed the pattern scenario that was

implement in LAMS [86] on the "What is greatness?" topic. This pattern has five main steps as follows:

- Step 1: Learners individually consider the question "In your opinion, what is greatness?" Each learner reads the question and reports his response where all learners can see all responses.
- Step 2: Each learner is asked to discuss how his own answer differs from those of other learners.
- Step 3: Learners are asked to choose up to five "great people" from a list of 20.
- Step 4: All groups are given content about greatness to consider.
- Step 5: Each group writes a report, and submits it to the instructor.

This pattern was implemented by the group's facilitator by simply mapping the suggested steps, by selecting their corresponded tasks patterns in the same consecutive order: Private Reflecting, Group Discussions, Idea Selection, Providing session info, and Resolution Reporting. Secondly, the teacher provided the resources (a list of 20 great people and the greatness content). The collaboration script page was filled in the collaboration script page within Beehive's authoring application according to the above scenario, and as a result, Beehive's run-time application successfully implemented that pattern.

In the other trial, the Problem Based Learning PBL, which is not a collaborative learning technique but still a famous type of learning, was implemented in RELOAD [73]. The Problem Based Learning is mainly used to actively involve learners with problems coming from real practices, and according to Koper, the PBL common scenario has the following steps:

1. Learners start by reading the problem description.
2. Learners try to identify concepts and parts of the problem that needs clarification.
3. Learners define the problem.
4. Learners brainstorm the solutions.
5. Learners structure solutions or causes.

6. Learners state learning objectives.
7. Directing each learner's studies towards learning objectives.
8. Each learner reports things learned.

This pattern was successfully implemented in Beehive, and the task patterns that represent this scenario were selected according to this consecutive order: Private Thinking, Individual Writing, Group Discussing, Idea Posting, Group Discussing, Result Reporting, Reading Role Info, Result Reporting.

Regarding the face-to-face environment, we successfully modelled four more pedagogical patterns: Round Table Discussion, Online Panel, Case Study, and Pro/Contra Pattern and included these new patterns in Beehive's repository.

5.3 Evaluating Beehive's Scripting Flexibility

Another key objective in Beehive is to enable instructors to engage in scripting their designs without much technical experience. To evaluate this objective we used three qualitative methods: Focus Group, Interviews, and System Logs. We selected these qualitative methods for their flexibility, sensitivity and the ability of obtaining meaningful conclusions in each case [99].

In this section, we present three evaluation cases that present how users in different situations and disciplines flexibly scripted their designs by reusing several iterations of pedagogical design patterns or by reusing a lower level of patterns, task patterns.

5.3.1 Teaching/Learning Case

In this evaluation, two of qualitative methods were used: **Focus Group**, and **System Logs** [21]. The **Focus Group** method was used to evaluate participants' approaches in designing a complex collaborative learning activity using the Beehive. The Focus Group method has many advantages. It is a useful tool to establish perspectives on issues, it can be used with a link to other evaluation methods, and it enables evaluators to obtain a large amount of interactive information on a topic within a short time. **System log** data

is a step-by-step recording of users' interaction with a software program. The findings from both methods are presented in narrative form, often with actual participants' quotations, to illustrate their perspectives.

This evaluation study was conducted within an online teaching/learning setting. It was carried out within the EDBC 5021 course that is taught at postgraduate level at the faculty of Education and Social Work in the University of Sydney. The course basically seeks to identify and compare important conceptual frameworks that relate technology with technology with learning. This course is an ideal for Beehive evaluation since it focuses on various modern educational technologies that are used, and which can be analysed from a number of perspectives, including classical information theory, psychological media, and communication theories. Beehive was one of the three use case applications required in that course.

Beehive was used in week 11: 12 participants (who were acting here as designers) started by defining their overall objective in using Beehive. Participants represented various backgrounds and experiences (students, teachers, educational technologists, managers, etc.).

The overall meeting objective was to develop a one-hour online lesson on the semantic web topic, using Beehive. It consists of:

- 15-minute lecture
- 20-minute collaborative sessions
- 20-minute other collaborative session

The groups were divided by the instructor into two (males and females). Each group needed to design plans for the collaborative sessions. After each report, groups should assess:

- Are the suggested pedagogical technique(s) appropriate?
- Can Beehive do it?
- Can it be achieved by a group of students within 20 minutes?
- What do students need to know to be able to do it?

At the beginning, participants started discussing the appropriate techniques that would be needed, by using a general chat tool (not within Beehive). The selection had been developed through steps of discussions. First, they agreed on the session's topic, which was about "learner centredness".

Second, they agreed that before selecting a technique, they needed to define exactly what learners needed to learn and achieve by the end of the session (defining their objectives):

"We need a technique that allows for exchange of ideas but then it is only the ideas coming out of one head."

"We can start by individual read, exchange thoughts in group, and then share responses with entire session."

"I think I agree, in some pretty complex problems people might be best to start individually."

In the third stage, participants started to discuss the appropriate techniques to choose. Some of them suggested that they needed to look at Beehive first to see what things Beehive can offer in order to gain a better understanding of what to do next (similar to their traditional approach in learning how to use new software). The real concern in this regard was the time limitation and the fear that Beehive would dictate learning outcomes:

"I suspect it would take too long ..."

"Why should the system dictate learning outcomes?"

Some of them relied on their experience and suggested a technique that they had used in the face-to-face environment:

"We want to see a group GENERATE something new. What about using group nomination?"

"Is not the brainstorm listing things? In this case it is a list of ways Semantic Web might support learner centredness."

Nevertheless, they were seeking more flexibility. Some participants stated that they need a technique that is appropriate to their objective, which was not among Beehive's techniques:

> *"You know I can not see a method for the way I would do it manually. If I did it manually, I would give a set of alternatives then get people to vote on it."*
>
> *"It would have been nice to see the outcomes from the activities and have them be able to translate nicely into inputs into other activities and then we find the techniques to support that process."*
>
> *"I think brainstorm needs to be followed with some sort of corrective activity, or else some may be left with incorrect assumptions."*

The next suggested approach for them was to combine more than one technique to reach their objectives:

> *"How might Semantic Web supports learner-centredness? We want first the group to have a sense of learner-centredness emerging. Then try to post some concrete 'examples' (fictional). Then elaborate from examples to form a group discussion. The scenario could be: start with brainstorming to exchange thoughts in group, then move to group discussion, then idea prioritising, and then share responses with entire session."*
>
> *"I think to do the same with subtopics (for example, challenges)."*

At the end of the session, the plan was reported as following:

> **The plan for the 20-minute collaborative activity**: presenting a hypothetical case study for Sydney University.
>
> **Pedagogical Technique**: Group Nomination + Pyramid
>
> **Output**: the output should be a document outlining the challenges and issues the collaboration might face, and any recommendations to address these challenges and issues.

They successfully implemented the above complex scenario containing 16 different phases via Beehive. The agreed session's script was as follows:

1- Overview (by the facilitator) – two minutes.

2- Watch the lecture presentation "The Educational Web Semantic" (by presenters) – 30 minutes.
3- Open the floor for any questions – three minutes.
4- Read the text and consider: *"The Semantic Web, despite presenting learners with new challenges (such as exposure to too much information), will bring the benefit of greater access to sources of knowledge which could be contextualised and then analszed critically. It will do this by empowering learners with the technological tools to retrieve opposing views on the information that is being sought and encouraging the analysis of both sides of a debate."* – two min
5- Post your individual ideas about how you think the semantic web would achieve this (in small groups) – three minutes.
6- Discuss these ideas – five minutes.
7- Mark these ideas to prioritise them – one minute.
8- Post your ideas on what will change from learners' perspective – four minutes.
9- Mark what is the most significant change – one minute.
10- Each group leader types in the most significant change – one minute.
11- Read the background case: *"The Faculty that you are working for within a large University is commencing a collaboration between like faculties within other universities both nationally and internationally. The desire is to share resources, tools, and learning designs and best-practice teaching strategies. Many of the schools are adopting a learner-centred paradigm and those that are not in the progress of doing so. The collaboration has agreed to look at implementing a semantic web as the basis through which all the universities will collaborate. As a key member of the faculty with an understanding of learning, e-learning and educational technology, you and other key staff members have been tasked to develop strategies to support the implementation of the semantic web for the collaboration."* – two min
12- Identify challenges that the collaboration and the individual faculties may face prior, during, and after – three minutes.

13- Each group leader type in the two most significant challenges – two minutes.

14- Post what might the semantic web offer in term of support training and the day-to-day informational – three minutes.

15- Mark these ideas to prioritise them – one minute.

16- Each group leader types in the highest two ideas – three minutes.

We have found in this evaluation that the participants' main approach in designing their complex activity was by including several iterations of pedagogical design patterns after the online presentation. The core pattern in this case was simply Group Nomination followed by a Pyramid technique. This was to allow learners to start by individual reading, exchange thoughts in their groups (male/female), and then share their responses with the entire session.

The main finding in this case evaluation indicates that Beehive was capable of implementing complex group activity based on collaborative learning settings.

5.3.2 Group-Meeting Case

To evaluate Beehive's scripting flexibility, we used the same two qualitative methods (**Focus Group** and **System log)**, but with different type of users and settings. We chose a group of postgraduate students working in the same research group to conduct one of their meetings in the online environment. The group meets face-to-face on a regular basis. They discuss issues regarding their research progress and application developments. They decided to use Beehive to conduct one of their meetings. The group director posts the initial meeting plan in the group's forum. There were two main issues to discuss, which included many phases. The first phase was to review the writing process of their thesis introduction. The second phase was to plan a picnic arrangement. The session script was as follows:

Phase1

Step 1 is to overview the session's plan (using audio by the facilitator) – three minutes.

Step 2 is to read background info on the general elements the introduction should have – three minutes.

Step 3 is asking each participant to post their key elements of their introduction –three minutes.

Step 4 is asking each participant to post his introduction chapter (peer-to-peer) – one minute.

Step 5 is letting each participant to review his peers' introduction – 25 minutes.

Step 6 is to post each reviewer's comments and suggestions – five minutes.

Phase 2

Step 7 is to agree on the picnic date – three minutes.

Step 8 is to agree on the location – three minutes.

Step 9 is deciding who will bring the drinks – three minutes.

Step 10 is deciding who will bring the food – three minutes.

Step 11 is deciding who will take care of supplying other materials (cups, tissues, etc.) – five minutes.

The session was carried out successful. Participants were generally satisfied with the process, since it was more flexible to meet online, as some of them were away at that time of the year. Also, they did not find it hard to engage and to coordinate their discussions using Beehive's run-time application.

5.3.3 Sport Case

The third case evaluation was conducted outside the teaching and learning discipline. In this study, we used the **Interview** qualitative methods to also evaluate Beehive's scripting flexibility. Online training in sport imposes additional challenges since it is traditionally performed in physical places (for example, courts).

The NSWIS [95] has used Beehive in conducting one of its training sessions. The NSWIS is the New South Wales sporting centre striving to improve NSW representation on national teams and to assist athletes to perform at world standard. Its services ensure that athletes have access to leading-edge coaching and sports technology. Its

mobile/regional program offers a comprehensive spectrum of support services to NSWIS athletes in their home environments. Fortunately, the NSWIS has decided to invest in Beehive in order to assist it in achieving these goals.

We interviewed the netball coach, who had little Internet technology experience, to evaluate her perception of how to use Beehive in conducting such session. The netball group members were scattered in various places (Sydney, Wollongong, Orange, and Newcastle). The session was designed to teach a particular skill, that is, "how to make one breaking zone from backline throwing". The online session was designed to be conducted after performing a physical training session, showing trainees that skill, in the court. The online session objective was to identify things that did not go well after viewing several videos.

The decided session structure followed the PBL technique (but with some customisation) as follows:

1. Trainees are asked to watch a video section.
2. Trainees need to identify what went wrong.
3. Trainees are asked to brainstorm some solutions.
4. Trainees are asked to discuss these solutions within their group.
5. Trainees are asked to draw on a still image how to apply these solutions.
6. The chairperson is asked to report on his group's consideration.

This process was repeated three times for three different videos, each took 15 minutes and each group size was three. Beehive was successfully used in conducting this challenging online group training without many technological difficulties.

As a conclusion, the overall evaluation from these three cases suggests that Beehive is capable of enabling various users (teachers, researchers, trainers, etc.) to engage in scripting flexible designs without much technical experience.

5.4 Users' Experience

In this section, we evaluate users' experience by mixing both qualitative and quantitative methods. We capitalise on the strength of using multiple approaches, thus minimising

biases across methods. In the qualitative evaluation, we used **observation** and **open-ended questions** to evaluate participants' responses with quotations. Feedback responses expressed with quotations reveal the respondent thoughts about participants' experiences and their basic perceptions. For the quantitative evaluation we used **closed questions** to evaluate users' perspective and experience through structured answers.

We start by presenting some of the designers' feedback and comments among themselves that reflect their experience after using Beehive's authoring application. We present some of the comments recorded in the sessions' logs of the Teaching/Learning case that reflect some of the advantages of using Beehive:

1- Beehive encourages teachers to define a session's tasks before going into detail:

"Personally I like to start with what tasks I want to do then go to the tool and design the interaction from my requirements."

2- Beehive enables teachers to relate to their experiences in selecting the proper technique:

"I'm a fan of pyramid discussions in the language classroom - works very well."

3- Beehive enables teacher to engage more effectively in the design:

" I can use the script page to control the session's workflow."

Also, some of the participants' encouraging comments were also recorded in relation to their experiences:

"It looks like it maps to real-time activities in classrooms rather than documents on a webpage."

"It is good to use a tool that allowed the benefits of collaboration to emerge so things that would draw on what a group can achieve over individual issues."

Next we present some of the learners' feedback and comments that reflect their experiences after engaging in Beehive sessions:

"It was quite easy to use Beehive since I needed to follow the instruction in the top of my screen."

"This session was helpful and in the same time I did it from the home."

"My peer had found many interesting points that I can benefit from, I wish we could do the same with other chapters."

"I realised that I need some one beside me to read my writing to point me to my weakness."

"I enjoyed this session, I felt **more** active."

For further evaluation, instructors were asked three open-ended questions: what they thought the purpose of Beehive was, how they went about using it to design student-learning activities, why they approached using Beehive in the way they did.

When asked what they thought the purpose of Beehive was, some instructors responded like this:

"Beehive is for developing the skills of students in the group."

"To provide patterns for on-line teaching in an accessible way and to automate processes of student group work."

"Model and structure face-to-face learning activities in an on-line way and provide a tool to manage on-line classes."

When asked the same question, other trainee teachers in the same class said:

"To enable students to work collaboratively over the Internet. To share ideas and come up with group decisions."

"To facilitate online collaborative learning sessions to promote student understanding."

While the first group of quotations are not necessarily inaccurate (although the concept of trying to transfer a face-to-face activity to the online context suggests an undeveloped understanding about the way online learning can help student learning), the instructors first and foremost wanted a tool to improve the effectiveness of student learning. In contrast, the second group of quotations showed an awareness that the main purpose of Beehive was to enable students to share ideas, and to "promote student understanding".

When asked how they approached using Beehive, the same first group of teachers responded:

> *"As a trainee teacher, I sit, watch, and do what I am told with the tools in Beehive."*
>
> *"I just pick the tasks I'll need, set the scenario and information and slides and sequence the rest of the tasks."*

The second group of teachers said things like:

> *"I prefer to do some preparation about student needs before I use the program. I like to have the script written before I go to Beehive."*
>
> *"I like to design learning activities in relation to student learning outcomes, especially when selecting the pedagogical techniques and selecting the tasks."*

We have noticed a consistent theme in the comments made by the trainee instructors in relation to their approach to using Beehive. Comments such as those in the first group tended to focus more on the software itself, "the tools in Beehive" or "set the scenario and information and slides". They did not tend to display any awareness of the learning context in which the online tools would be used. In contrast, the second group of comments about teacher approaches to using Beehive fore-grounded the educational context.

A third phase of evaluation includes closed questions, in which Likert-type rating scales were used to gather designers' opinions. This allowed us to evaluate users' opinions and perspectives in using Beehive in a standardised and more objective way. The sample size for this study was 12 designers. They represent a wide range of users which include:

- Postgraduate students enrolled in the class "Introduction to the Learning Sciences"
- Coaches and athletes at the NSW Institute Fof Sport.
- Researchers involved in group-based meetings.

The intended outcomes of the survey were to gain:

1- A better understanding about the user's familiarity with face-to-face collaborative techniques in different domains.
2- A better understanding about the availability of the operating environment (PC, Internet connection, etc.).
3- A better understanding about the expected learning outcomes of using synchronous collaborative learning.
4- A better understanding of the sufficiency of using Design Patterns, Task Patterns, and Collaboration Script in designing meaningful learning designs.
5- A better understanding about the ergonomics of developing new collaborative tools using Beehive.

The survey (Appendix B) results indicate that all 12 participants had easy access to computers and Internet at the places in which they do their off-campus/work tasks. Second, most of them (10 participants) were quite familiar with some of the face-to-face collaborative techniques. A high percentage of them even conducted some of these techniques during their study (9 out of 12). Nevertheless, very few (only one) had actually conducted collaborative technique in the online environment. This shows that the majority of users are familiar with face-to-face collaborative techniques, but few know how to conduct these techniques in the online.

Interestingly, most of the participants think that effective learning outcomes can be accomplished by using synchronous collaborative learning. About 75 per cent of participants think that conducting collaborative techniques would improve creative and critical thinking skills, while only 58 per cent think that it would improve social skills. The reason behind this low percentage might be that social skills development is traditionally accomplished in the face-to-face environment. In general, 83 per cent think that Beehive would help in improving learning outcomes.

In the same sense, they think that using Design Patterns, Task Patterns, and Collaboration Script would have a positive influence on designing meaningful learning experience. Almost 58 per cent of participants think that presenting tasks as patterns helped them to understand more about these tasks. Only 41 per cent think that designing

a collaboration script is easy. In response, more documentation and multimedia animations were published to aid participants to understand more how to design collaboration scripts. Sixty six per cent stated that saving their designs in Beehive's repository would help others to reuse the successful techniques. In general, participants (75 per cent) felt that using Beehive would encourage them to apply synchronous collaborative learning in their designs and 83 per cent think that Beehive hides the technical difficulties of choosing and configuring appropriate tools needed to implement their designs. The majority of participants also felt that developing new collaborative techniques using Beehive was not so complicated (66 per cent). Still one third did not share this view since they find it hard to engage smoothly in designing new sessions before gaining more experience on using the stored patterns that Beehive has.

5.5 Evaluating Beehive's Design Elements

In this section we compare Beehive with different CSCL applications, according to several software design elements but from the pedagogical perspective, since CSCL technology cannot be designed and evaluated in isolation from pedagogy. Designing such software must address a number of technical and pedagogical issues, such as representation, granularity in sequence, design process, scripting, etc.

We started by selecting the CSCL applications to be compared with Beehive. We chose four applications that represent a wide spectrum of collaboration tools. The first application was Breeze, which represents collaboration tools that have communication components embedded in their main interfaces, such as Centra, Webex, interwise, Skype, ConferenceXp, etc. The second application was LAMS. It represents tools that allow users to design collaborative sessions by sequencing collaboration components. LAMS is currently the only software of this kind. The third application was RELOAD LD, which represents applications that are compliant with IMS LD, such as dotLRN IMS LD, CopperCore, ASK LDT, etc. Finally, the COLLAGE application could be categorised within the IMS LD category, but it has an important extension in reusing stored pedagogical techniques.

- **Representation**

The user interface of various tools is essential in adopting these solutions in teaching and training environments. As shown in Figure 46, LAMS enables teachers to drag and drop various tools and sequence them by simply drawing lines between them. These tools represent various mini-activities such as reflection, discussion, resource-sharing, etc. For group activity tools, the teacher needs to define the group sizes. Only one role needs to be specified in some tools (for example, Chat&Scribe), that is, the group leader who is responsible for submitting his group's work.

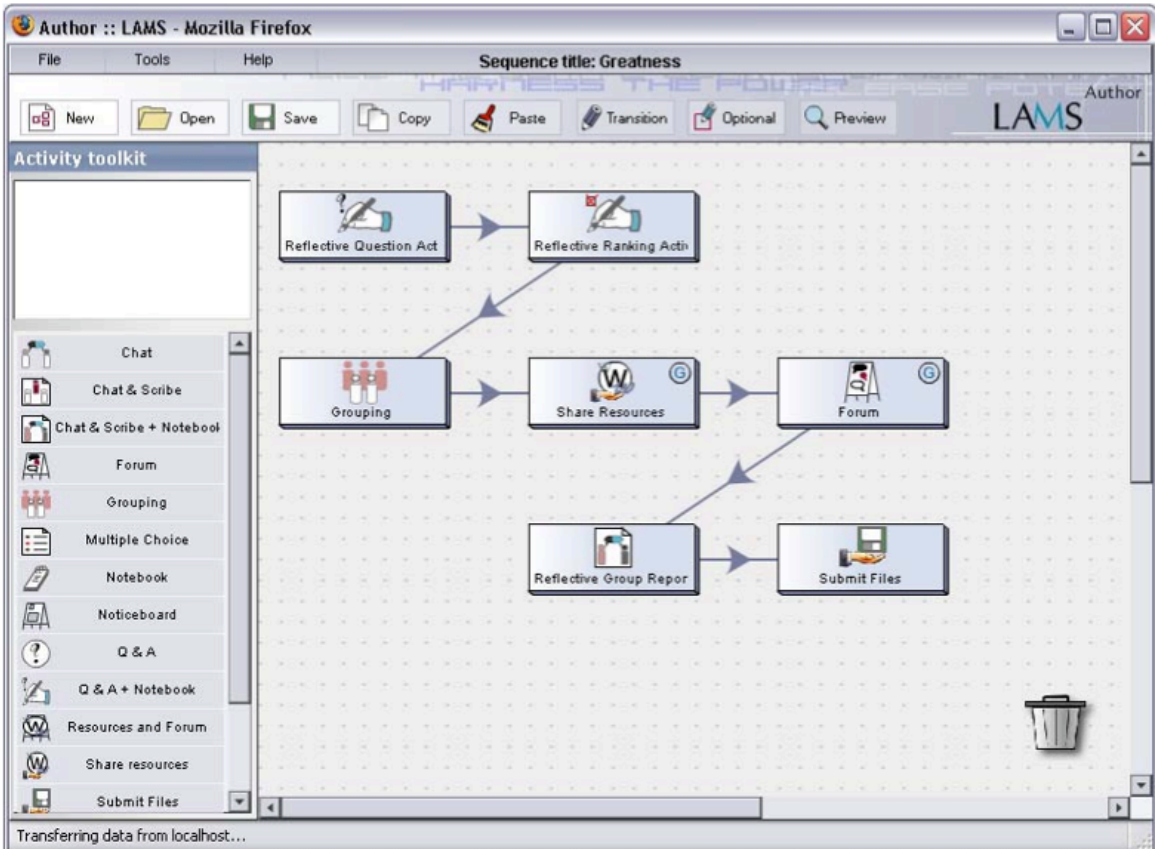


Figure 46: LAMS screenshot

RELOAD LD (as shown in Figure 47) uses a representation with various layouts (General, Role, Environments, Activities, Methods, and Resources). They allow users to define the various elements needed to implement activities in a direct way. In each

activity, instructors need at least to define the task description to be carried out. Tasks are flexible to all kind of tasks. They may be online or offline. Offline tasks are carried out outside RELOAD, such as visiting the library, doing an experiment, etc. The online tasks might be associated with just reading content or collaborating with peers or both. Teachers can add resource materials that are associated with that task. Also, a collaboration service may be added to that activity (Forum, Chat, etc.). For simple activity structures with only one type of role, teachers with a basic technical background can use RELOAD to add a sequence of activities in a simple way. But for activities that require various roles and multiple structures, they need instructional designers with deep IMS LD knowledge.

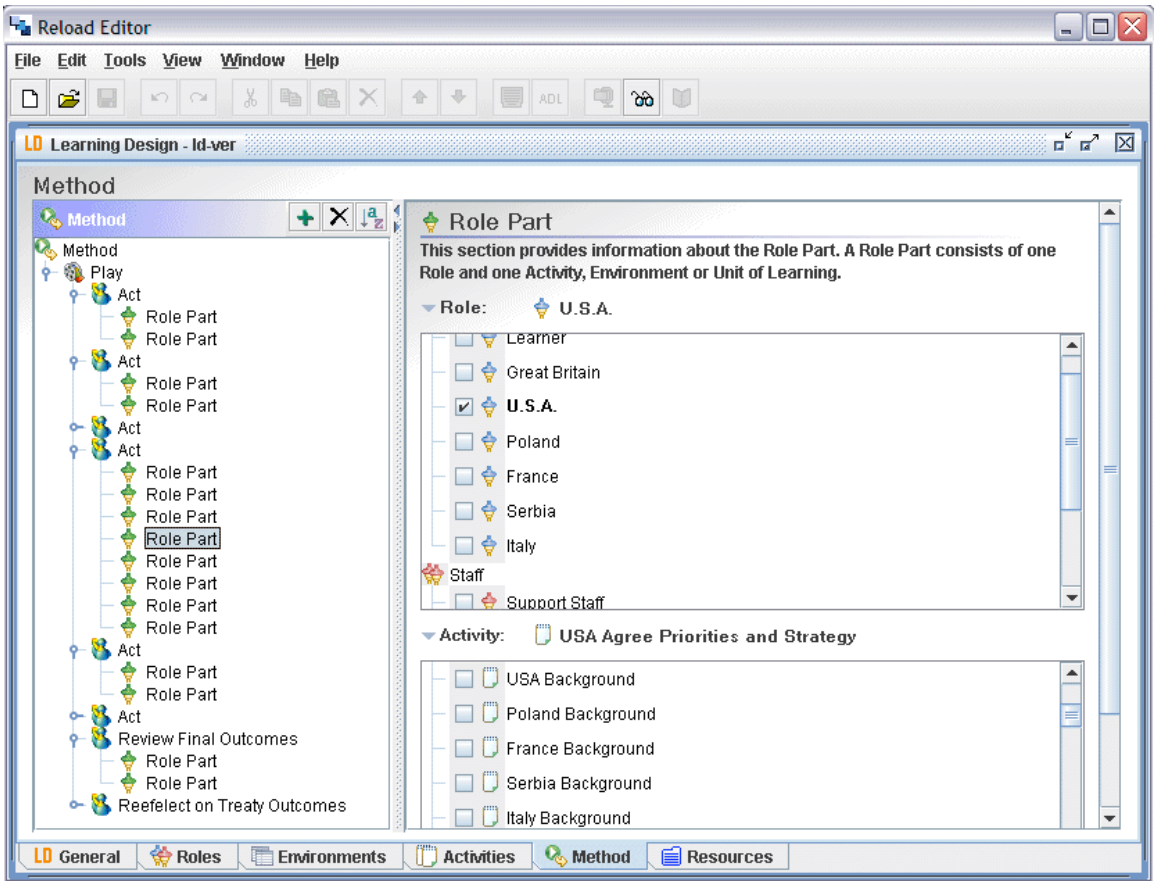


Figure 47: RELOAD LD screenshot

The COLLAGE representation is directed by selecting one of its four pedagogical patterns (Brainstorm, Tips, Pyramid, and Jigsaw). Each pattern has a visual representation (as shown in Figure 48) where teachers can easily specify and edit patterns' parameters (for example, group size, objective, etc.) and couple any required resources. COLLAGE is implemented on the top of IMS LD specifications. COLLAGE is easier to understand and use than RELOAD LD, since it guides users with a wizard-type interface in selecting and implementing these pedagogical patterns.

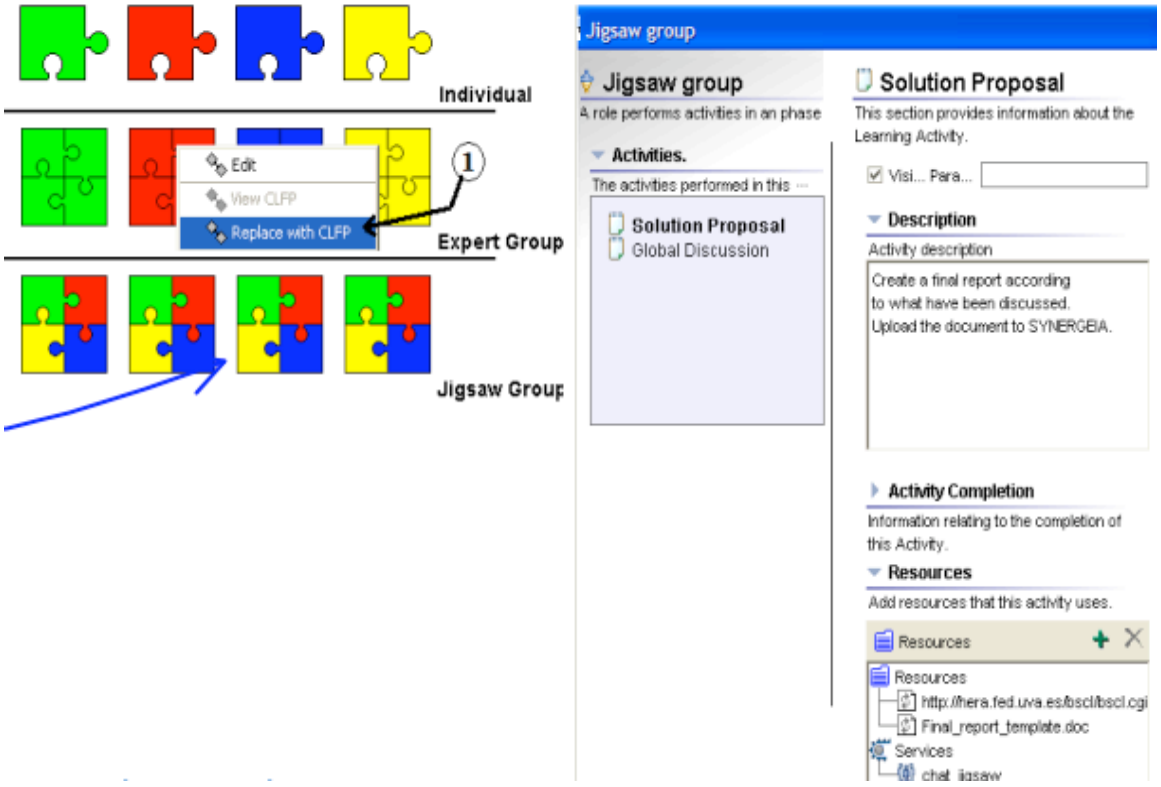


Figure 48: COLLAGE screenshot

Breeze has three main layouts to choose from (*Collaboration, Discussion*, and *Sharing*). Each layout has default components, which hide the technical difficulty in selecting these components. The *Collaboration* layout, as shown in Figure 49, has five components (Camera &Voice, Chat, Note, File-sharing, and Whiteboard). This layout allows participants to collaborate by using various types of communication: text, audio,

video, graphs, etc., and exchange resources by using the file-sharing components. The *Discussion* layout also has five components (Camera &Voice, Chat, Discussion, Note-box, Poll). It has the basic text/audio/video communication tools but in different sizes (the Chat component is bigger since the focus here is on the text discussion). The last layout is the *Share,* which still allows participants to communicate, but the main focus is on desktop-sharing and application-sharing. The main limitation in Breeze is that the teacher cannot customise these layouts according to their own needs in a direct way (for example, having the File-sharing and Poll in the same layout).

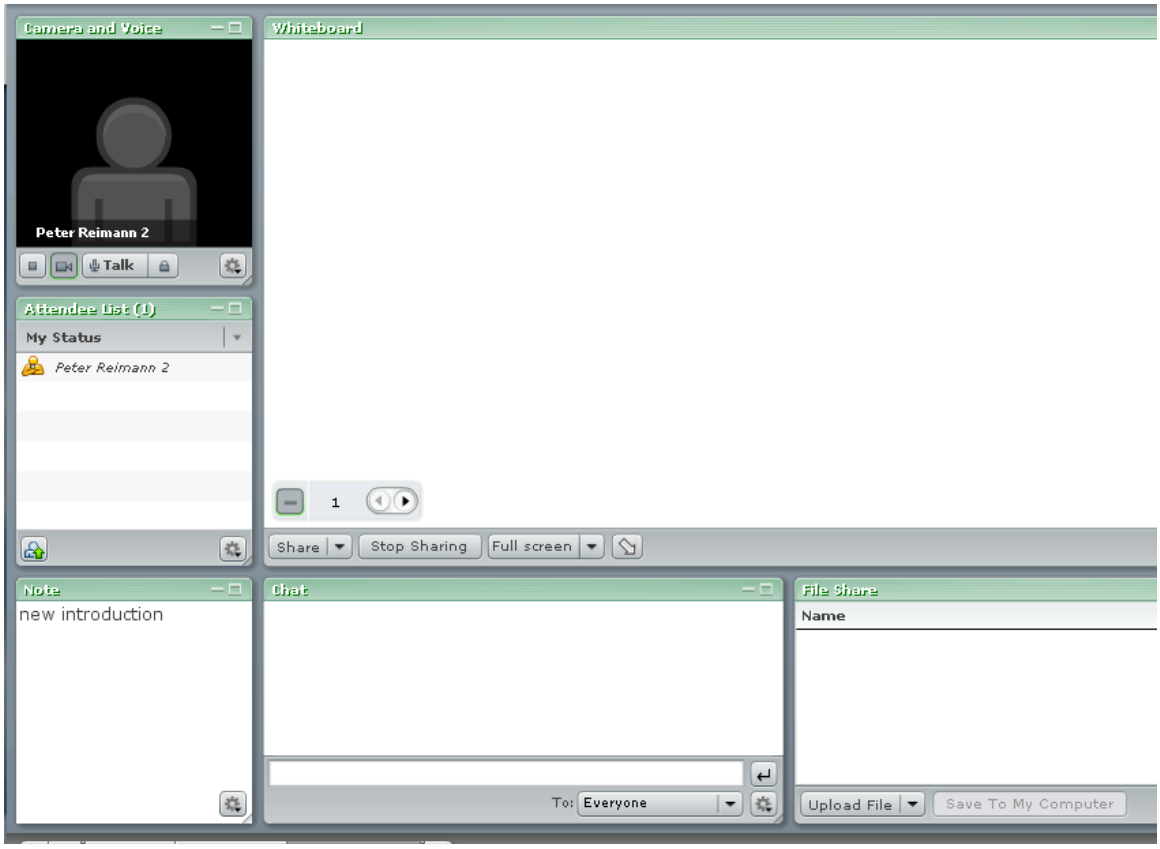


Figure 49: Breeze screenshot

Beehive's representation has more options. The first option is presenting a pedagogical wizard that guides teachers to select and implement a wide range of pedagogical patterns. The second option is presenting a general-type wizard that enables

teachers to construct new techniques. This option is more difficult to use and needs some background in Beehive.

- **Granularity in Sequences**

The level of granularity of the components offered by the system determines what can be reused. Beehive, LAMS, and RELOAD offer designers a "coarse" granularity whereby each phase describes a major activity in the session (Discuss, Brainstorm, Debate, Search, etc.).

RELOAD allows for flexible sequencing while COLLAGE only enables designers to select from a set of predefined sequences and limits the assembly of new structures by the composition of these predefined sequences. A smaller granularity is offered only by Beehive. It allows breaking a single phase into smaller phases, for example, structuring the Debate activity: who will speak first and who will speak next and so on. Breeze does not offer sequencing. It provides a set of tools for collaboration that is shown at all times. Also the granularity is very high, in which it provides a bundle of tools in each setting.

The type of tools, either synchronous or asynchronous, provided for sequencing is also important. RELOAD and COLLAGE provide four major collaboration tools (Chat, Forum, E-mail, Conference). Breeze offers more: Camera & Audio, Chat, Note, File Submitter, Whiteboard, Desktop-sharing, and Polling. LAMS has a few more: Chat, File Submitter, Notebook, Q/A, Polling, Forum, Multiple Choice, Share Resources, Chat & Scribe, Resources & Forum, Notebook & Q/A, Notebook & Polling, and Notebook & Chat & Scribe. LAMS currently provides no means for audio/video communication between learners. Beehive has only synchronous tools and assumes the asynchronous are provided by the LMS container. It includes Text Chat, Audio Chat, Video-conferencing, Idea Listing, Idea Marking, Collaborative Text Writing, Collaborative Writing, Collaborative Annotating, Collaborative Presenting, Notebook, Timer, File Submitter, Tracker, Survey, and Q/A.

- **The Design Process**

The way teachers design their activities is important to their success. In LAMS the designer starts by selecting and sequencing the tools, requiring a minimum technical background by using a graphical user interface with a drag-and-drop editor. The main challenge to this is that the focus of attention is the tool and not the pedagogical situation. In Breeze the designers select one of three layouts according to the session's objective, while in COLLAGE the designer selects a stored pedagogical technique structure for which he needs to understand the technique and appropriate scenarios. In RELOAD the designer composes the activities using IMS LD elements (resources and services), making it more difficult for users who do not know the specification. In Beehive the designer starts by selecting a stored pedagogical structure or by selecting a session's tasks (to minimise the technical difficulty).

In some cases it is of interest to be able to design the activities in a collaborative fashion. LAMS, RELOAD and COLLAGE only allow the teacher to design the session. In Beehive both teachers and participants may engage in designing the collaborative session, giving more freedom to the learners to design their session tasks (a feature that follows the socio-constructivist theory).

- **CSCL Scripting**

Collaboration script, as discussed in the previous chapter, describes the mini-activity flow inside pedagogical techniques. LAMS allows for a modelling scripting, but in a low-level format. It basically allows teachers only to sequence and configure specific tools in a linear way. RELOAD enables designers to script learning design but it does not enable them to specify how groups should collaborate inside a single learning activity. Breeze does not allow any kind of scripting. COLLAGE uses a stored script (macro script) in running certain pedagogical techniques. Beehive is the only application that allows flexible scripting (Section 3.5), which allows designers to specify exactly how learners interact inside each step. Beehive allows all levels of scripting (low, intermediate, and high), which enables designers to direct the session's phases in a strict or flexible way. This important feature allows instructors to engage effectively in designing their own designs to achieve the appropriate outcomes.

- **LMS Integration**

Students use an authentication system and login to the LMS, where they find the resources or activities they are looking for. The collaborative learning platforms must be integrated into these LMSs in order to give a seamless experience. LAMS, RELOAD, Breeze, and COLLAGE can be linked to the LMS by external links. Only Beehive is fully integrated with dotLRN and Sakai.

- **Reusability of Pedagogical Patterns**

Of the systems studied, only Beehive and COLLAGE store pedagogical techniques that the designer can reuse. Beehive has a wider range of stored techniques (now 17) while COLLAGE has currently four.

Reusability: Beehive allows full flexibility in structuring collaboration designs and recording successful designs' structures for further reusability. LAMS allows sequences to be stored in a Sequence Collection folder. Its main challenge is that different users might find it difficult to reuse others' sequences in different contexts, since these sequences are not presented as pedagogical patterns. Breeze does not support forming new learning designs other than those specified by the system.

Another way of improving the reusability of designs is to build and use software that supports the specifications, such as IMS LD. Some systems like LAMS are not fully compliant but are coherent with IMS standards. Beehive is also partially compliant and also extends IMS LD. RELOAD and COLLAGE are both fully compliant with IMS LD standards.

- **Prior Pedagogical Knowledge**

Normally, a wide range of users (especially educators) have some background in various pedagogical techniques. Teachers, trainers, and managers are usually familiar with some face-to-face pedagogical technique (Brainstorming, Debate, Group Discussion, etc.), but they are unfamiliar with how to apply these techniques in the online environment. LAMS, Breeze and RELOAD provide a free environment for designing pedagogical techniques, but they do not provide any guidance on how to apply specific techniques.

Beehive and COLLAGE present some of these techniques as patterns and provide the proper background and default settings on how to implement these techniques.

- **Real-time Scaffolding**

Real-time support and feedback are very important for successful learning experiences. They allow the teacher to be "present" for the students, providing help when it is needed. LAMS, RELOAD and COLLAGE provide no means to allow teachers to communicate with learners at run-time. LAMS only allows monitoring. In Beehive, teachers can send comments during the activity to a specific group, role, or participant. Also, it allows more flexibility for facilitators to move from one group to another.

- **Client Application**

For easier adoption, clients' browsers need to have minimum difficulties in installing applications. LAMS, Breeze, and Beehive do not require additional installation. Breeze and Beehive require Flash Player, which is normally installed on most computers and platforms. RELOAD needs clients to install the RELOAD player and to import the designs packages to run them. Running the player properly demands computers with high resources. COLLAGE needs the installation of an additional package on top of RELOAD.

6 Conclusion

We have introduced in this thesis the concept of the pedagogical application framework that combines collaborative learning design patterns with software components. The framework's original objectives were as follows:

- Guiding software developers in creating application software for collaborative learning.
- Supporting teachers in selecting and implementing various collaborative learning techniques in the online environment.
- Implementing not only a limited number of collaborative techniques, but flexible enough to implement an unlimited number of techniques.
- Encouraging instructors to share their successful designs with the larger community.

Beehive's conceptual model was described in detail in Chapter 3. In that chapter, we presented Beehive's various aspects and components. We described 14 face-to-face collaboration techniques that represent a wider range of techniques and demonstrated the modelling of some of these techniques. We successfully used the concept of Software Component described in the "framework" definition in assembling appropriate pedagogical collaboration tools needed to support these techniques.

It was a significant challenge to develop such components, due to the wide conceptual gap between pattern abstractions in the software engineering and the educational discipline. We effectively overcame this challenge by defining 29 task

patterns that can be used to form a wide range of collaborative techniques. The abstractions of these task patterns were driven from modelling face-to-face techniques while their implementation has been described by using the software components. These tasks' main benefit is that instructors can easily identify them, and the system can map them internally to their related software components. This has hidden the technical difficulties from instructors in selecting, configuring, and assembling components needed to facilitate their pedagogical design. In Chapter 3, we also described 12 software components that can be used to facilitate the 29 task patterns. In order to present these components according to the techniques' scenario, we presented a collaboration scripting language, and in order to standardise the representation of this scripting language, we proposed three sets of IMS LD extensions.

As a proof of concept, we implemented Beehive according to the conceptual framework model, as described in Chapter 4. We presented, in this chapter, the design of Beehive's authoring application that has guided users, in reusing, customising, or creating new synchronous pedagogical techniques. To offer open systems, the authoring application has been built using the dotLRN learning management system, while the run-time application has been implemented using the Flash Media Server. We also presented the design of a run-time application that has enabled both teachers and learners to engage effectively in the designed collaborative technique with minimum effort.

In general, Beehive has satisfied the basic CSCL requirements, stated in section 2.2.1:

- Community based: Beehive has permitted learners to be situated in learning communities.
- Simplicity: Instructors have been active in applying technological solutions to their designs. Beehive has taken instructors' pedagogical experiences into account and in the same time has hided technical difficulties.

- Customization: Several users had used the exact technique's structure in their designs but in most cases, they had customized them by changing their default tasks or scripts.
- Mediation: The collaboration scripts have allowed Beehive to mediate the communication and collaboration process of many groups in an automated way.

Also Beehive has fulfilled some of the general application framework attributes, such as Modularity, Extensibility, Reusability, etc. For modularity, Beehive has been divided into two main parts (authoring and run-time). This modularity has helped in interoperability with different systems. For applying the interoperability between the two systems, collaborative learning designs are published according to the IMS LD specifications, while exchanging data messaging is carried out according to our proposed a communication protocol (middleware). To investigate the validity of this middleware, we created an authoring environment for the dotLRN Learning Management System, and in a spin-off of this a project another authoring environment for the Sakai LMS was built by other student. Both authoring environments successfully published XML LD schemas that were interpreted by the same run-time application (Flash application).

For extensibility, the authoring environment was also integrated in two LMS platforms (dotLRN and Sakai). This provided many benefits. It has enabled Beehive to inherit various LMS features (user management, course managements, assessments, etc.).

For reusability, Beehive has been capable of reusing software components in creating new CSCL components that have been used in supporting new design patterns.

As a part of its evaluation, Beehive was used by a group of real users who represent different disciplines, as described in the evaluation chapter. They were able to reuse, customise and create various pedagogical patterns according to their teaching/training situations. They were also able to develop new pedagogical technique tools with relatively minimum effort.

According to the various evaluation studies made in the Evaluation Chapter, Beehive has encouraged teachers/designers to define their session objectives and conceptual plans before starting their session's design. Beehive has also tend to encourage teachers to relate to their teaching experiences in selecting and reusing proper designing techniques, but we noticed that there is a variance between teachers in their perception on the meaning of design reuse; Inexpert users designing simple sessions tend to start their designs by reusing learning design patterns from the patterns repository, while expert users designing long scenarios that involve many phases and steps tend to script their designs in more flexible ways by reusing task patterns, which are lower level of patterns than learning design patterns.

The majority of instructors felt that developing new collaborative techniques using Beehive was not so complicated. The overall evaluation suggests that Beehive is capable of enabling various users (teachers, researchers, trainers, etc.) to engage in scripting flexible designs without much technical experience. Furthermore, only Beehive among similar applications allows micro-scripting, which enables designers to direct the session's phases in a strict or flexible way. This important feature has allowed instructors in engaging effectively in designing their sessions and achieving their goals.

According to the learners' feedback and comments presented in the evaluation chapter, we list some of the following common observations:

- Generally satisfactory feelings were noticed after conducting each session.
- Some learners still felt that they were under time pressure and sometimes uncertain about what they needed to do. Nevertheless, they managed to overcome these issues after engaging in more than one session.
- It was noticeable that the scaffolding level tended to drop over time.
- Confidence in engaging in such online activity increased over time. New users felt more confident when they started the session by performing the Icebreaking Task.

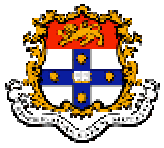
Beehive is currently being used by the NSWIS to deliver online group-based training to its athletes scattered over various locations. This has allowed Beehive to be tested within real-life situations that impose a more challenging environment.

As a conclusion, Beehive has in fact assisted software developers in developing computer supported systems for collaborative learning that can be used to support instructors in creating and reusing pedagogical patterns, and has been flexible enough to enable instructors to design various collaborative activities that involve small numbers working on collaborative tasks.

Although Beehive might not be able to describe all possible situations in synchronous collaborative learning, the current components provided in our framework, since they are based on pattern concepts, can support a considerable wide range of synchronous collaborative learning possibilities.

Appendix A: The Breeze Usability Questionnaire

The University of Sydney



Web Engineering Group, Electrical and Information Engineering

Bldg J03

Telephone +61 9351 8171

Email rafa@ee.usyd.edu.au

Aiman Turani

SURVEY

I. General Information:

The questions in this section ask you about your personal details and previous experience with ICT. Please write down your answers or tick the appropriate boxes.

1- Have you ever participated in the Breeze: Text Chatting _YES, Audio-conferencing __YES, Video-conferencing NO_, Whiteboard Drawing___NO__, Desktop-sharing __NO__?
2- What type of Internet connection you usually use? Broadband
3- Where is your location (city and country)? Sydney
4- What is the usual size of your Breeze group? 5-10

II. Technical feedback regarding Breeze:

Please respond to the following statements by circling the number that most closely reflects your expectations and feelings: from 1- “Strongly disagree”. to 4- “Strongly agree”.

1- The audio was understandable.
2- There were no problems at any point during the session with the sound loudness.
3- There were no problems at any point during the session with the sound quality (for example, with feedback, echo).
4- The quality of the audio varied a great deal throughout a session.
5- The video image was clear.
6- There were no problems with video at any point during the session with the quality of the video.

Appendix B: The Beehive Usability Questionnaire

The University of Sydney

Web Engineering Group, Electrical and Information Engineering
Bldg J03
Telephone +61 9351 8171
Email rafa@ee.usyd.edu.au

Dr. Rafael A. Calvo

SURVEY

Beehive: an Application Framework for Synchronous Collaborative Learning

I. General Information:

The questions in this section ask you about your personal details and previous experience with ICT. Please write down your answers or tick the appropriate boxes.

5- Degree, if any, you are enrolled in: _____
6- Your current occupation: _____
7- Are you familiar with face-to-face collaborative learning techniques (for example, Group Discussion, Brainstorming, Debate, etc.)?

Yes No

8- Have you previously used any of these techniques in your classes?

Yes No

9- Have you taught a course that had any synchronous collaboration component (for example, Chatting, Audio-conferencing, Video-conferencing, Whiteboard drawing)?

Yes No

10- Do you have easy access to a computer and Internet at the place in which you do most of your off-campus study (for example, home)?

Yes No

II. Synchronous Collaborative learning: expectations:

The questions in this section ask you about collaborative learning and the use of Beehive in designing and implementing online collaboration activities. Please respond to the following statements by circling the number that most closely reflects your expectations and feelings: from 1- "Strongly disagree". to 4- "Strongly agree".

		Strongly disagree			Strongly agree
1.	Using pedagogical techniques from the Beehive repository will help me in designing new techniques	1	2	3	4
2.	Presenting tasks as patterns helps me to understand more about these tasks	1	2	3	4
3.	Designing a collaboration script is easy in Beehive	1	2	3	4
4.	Using the session's transcript is important to assess and improve the session's performance	1	2	3	4
5.	Saving my design in Beehive's techniques' repository will help in sharing and reusing successful designs	1	2	3	4
6.	Beehive will save my effort in choosing and configuring the appropriate tools needed to implement synchronous	1	2	3	4

	collaborative learning				
7.	Beehive will improve the social skills (leadership, decision - making, trust-building, conflict-management, etc.) among my students	1	2	3	4
8.	Beehive will improve creative and critical thinking among my students	1	2	3	4
9.	Using Beehive will encourage me to implement synchronous collaborative learning in my classes	1	2	3	4
10.	Beehive will help in improving the e-learning outcomes	1	2	3	4
11.	The main objective of Beehive was clear to me	1	2	3	4

12. What do you like most about Beehive?

_____

_____

_____

_____

_____

_____

13. What concerns do you have about Beehive?

_____

_____

_____

Appendix C: Definitions, Acronyms, Abbreviations

AICC	Aviation Industry CBT Committee
API	Application Program Interface
Beehive	Pedagogical application framework for collaborative learning
CSCL	Computer Supported Collaborative Learning
dotLRN	.LRN Courseware Management System
IMS LD	IMS Learning Design Information Model
LMS	Learning Management System
OpenACS	OpenACS general community
SCORM	Sharable Content Object Reference Model
XML	Extensible Markup Language

Appendix D: Licensing

Thesis Licence:

Attribution-NonCommercial-ShareAlike 2.0

You are free:

- to copy, distribute, display, and perform the work
- to make derivative works

Under the following conditions:

Attribution. You must give the original author credit.

Noncommercial. You may not use this work for commercial purposes.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a licence identical to this one.

- For any reuse or distribution, you must make clear to others the licence terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder. Your fair use and other rights are in no way affected by the above.

License Source Code

Beehive is available under the GNU General Public License [53]

References

1. ADL Advanced Distributed Learning. Available at http://www.adlnet.org (last accessed 22 Jan 2005).
2. Adobe. Available at http://www.adobe.com/products/breeze/index.html (last accessed 22 May 2006).
3. AICC Aviation Industry CBT Committee. Available at http://www.aicc.org (last accessed 25 Jan 2006).
4. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I. and Angel, S. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, NY, 1977.
5. Altenhofen, M. and Schaper, J., Flexible instructional strategies for e-learning. In Proceedings of the *35th Annual Hawaii International Conference on System Sciences* (Hawaii, 2002), 342-351.
6. Angeles, M.d.l. and Duthers, D.D., Coaching Collaboration by Comparing Solution and Tracking Participation. In Proceedings of the *Computer-Supported Collaborative Learning* (Maastricht, Netherlands, 2001), 173-180.
7. Asensio, J.I., Dimitriadis, Y.A., Heredia, M., Martinez, A., Alvarez, F.J., Blasco, M.T. and Osuna, C.A., Collaborative Learning Patterns: Assisting the Development of Component-Based CSCL Applications. In Proceedings of the *12th Euromicro Conference on Parallel, Distributed and Network-based* (Spain, Coruna, 2004), 218-224.

8. Augar, N., Raitman, R. and Zhou, W., Teaching and learning online with wikis. In Proceedings of the *Ascilite* (Perth, Australia, 2004), 95-104.
9. Avgeriou, P., Papasalouros, A., Retalis, S. and Skordalakis, M. Towards a Pattern Language for Learning Management Systems. *Educational Technology & Society*, *6* (2). 2003, 11-24.
10. Baggetun, R., Rusman, E. and Poggi, C., Design Patterns For Collaborative Learning: From Practice To Theory And Back. In Proceedings of the *World Conference on Educational Multimedia, Hypermedia and Telecommunications* (Chesapeake, VA, 2004), 2493-2498.
11. Baumgartner, P. and Bergner, I., Categorization of Virtual Learning Activities. In Proceedings of the *International Workshop ICL2003* (Villach, Austria, 2003).
12. Blackboard. Available at http://www.blackboard.com (last accessed 12 May 2006).
13. Bonk, C.J. and Sugar, W.A. Student role play in the World Forum: Analyses of an arctic learning apprenticeship. *Interactive Learning Environments*, *6* (2). 1998, 1-29.
14. Bonk, C.J. and Wisher, R.A. Applying Collaborative and e-Learning Tools to Military Distance Learning: A Research Framework, United States Army Research Institute for the Behavioral and Social Sciences, Alexandria, 2000.
15. Bote-Lorenzo, M.L., Hernández-Leo, D., Dimitriadis, Y., Asensio-Pérez, J.I., Gómez-Sánchez, E., Vega-Gorgojo, G. and Vaquero-González, L.M. Towards Reusability and Tailorability in Collaborative Learning Systems Using IMS-LD and Grid Services. *Advanced Technology for Learning*, *1* (3). 2004, 129-113.
16. Bouras, C., Triantafillou, V. and Tsiatsos, T. A Framework for Intelligent Virtual Training Environment: The Steps from Specification to Design. *Educational Technology & Society*, *5* (4). 2002.
17. Bourne, J.R. Net-learning: Strategies for on-campus and off-campus networkenabled learning. *Asynchronous Learning Networks*, *2* (2). 1998, 70-88.

18. Boyle, P.G. *Planning Better Programs*. McGraw-Hill Book Company, New York, 1981.
19. Brogan, P. Using the Web for Interactive Teaching and Learning, Macromedia, 2000.
20. Calvo, R.A. and Turani, A., E-learning frameworks: reusing design and implementation. In Proceedings of the *7th IEEE International Conference on Advanced Learning Technologies* (Niigata, Japan, 2007), Submitted.
21. CDC Evaluation of HIV Prevention Programs Using Qualitative Methods. Available at http://www.cdc.gov/healthyyouth/publications/hiv_handbook/index.htm (last accessed March 2007).
22. Centra. Available at http://www.centra.com (last accessed April 2007).
23. COLLAGE Collaborative Learning design Editor. Available at http://gsic.tel.uva.es/collage (last accessed 22 June 2006).
24. ConferenceXp. Available at http://research.microsoft.com/conferencexp/ (last accessed 22 May 2005).
25. Conole, G., Dyke, M., Oliver, M. and Seale, J. Mapping Pedagogy and tools for effective learning design. *Computer & Education*, *43* (2004). 2004, 17-33.
26. Cortez, C., Nussbaum, M., Rodríguez, P. and Rosas, R. Teacher Training with Face-to-Face Computer Supported Collaborative Learning. *Computer Assisted Learning*, *21*. 2005, 171-180.
27. Dalal, S. The Next Generation of Live Learning, 2003. Available at http://www.clomedia.com/content/anmviewer.asp?a=261&print=yes (last accessed 23 April 2005).
28. Dede, C. The evolution of distance education: Emerging technologies and distributed learning. *The American Journal of Distance Education*, *10* (2). 1996, 4-36.
29. Derntl, M., The Person-Centered e-Learning Pattern Repository: Design for Reuse and Extensibility. In Proceedings of the *World Conference on Educational*

Multimedia,Hypermedia & Telecommunications (Lugano, Switzerland, 2004), 3856-3861.

30. Derntl, M. and Hummel, K.A., Modeling context-aware e-learning scenarios. In Proceedings of the *Third IEEE Conference on Pervasive Computing and Communication Workshops* (Kauai Island, Hawaii, USA, 2005), 337-342.
31. Derntl, M. and Motschnig-Pitrik, R., Conceptual Modeling of Reusable Learning Scenarios for Person-Centered e-Learning. In Proceedings of the *International Workshop for Interactive Computer-Aided Learning* (Villach, Austria, 2003), 1-12.
32. Derntl, M. and Motschnig-Pitrik, R., Patterns for Blended, Person-Centered Learning: Strategy, Concepts, Experiences, and Evaluation. In Proceedings of the *ACM Symposium on Applied Computing* (Nicosia, Cyprus, 2004), 916-923.
33. Derntl, M. and Motschnig-Pitrik, R. The Role of Structure, Patterns, and People in Blended Learning. *The Internet and Higher Education*, *8* (2). 2005, 111-130.
34. Derntl, M. and Motschnig-Pitrik, R., Towards a Pattern Language for Person-Centered e-Learning. In Proceedings of the *Society for Information Technology & Teacher Education International Conference* (New Mexico, USA, 2003), 2379-2382.
35. Derntl, M. and Pitrik, R.M., A Pattern Approach to Person-Centered e-Learning Based on Theory-Guided Action Research. In Proceedings of the *Networked Learning Conference* (England, 2004).
36. Dillenbourg, P. Introduction: What Do You Mean By "Collaborative Learning"? in Dillenbourg, P. ed. *Collaborative learning—Cognitive and computational approaches*, Elsevier, Oxford, 1999, 1-19.
37. Dillenbourg, P. Over-Scripting CSCL: The risks of blending collaborative learning with instructional design. in Kirschner, P.A. ed. *Three worlds of CSCL. Can we support CSCL?*, Heerlen:Open Universiteit Nederland, 2002, 61–91.
38. Dillenbourg, P., Baker, M., Blaye, A. and O'Malley, C. The evolution of research on collaborative learning. *Learning in Humans and Machine: Towards an interdisciplinary learning science*. 1996, 189-211.

39. Dimitracopoulou, A. and Petrou, A. Advanced collaborative distance learning systems for young students: Design issues and current trends on new cognitive and metacognitive tools. *Education International Journal*, *4* (11). 2005, 214-224.

40. Dimitriadis, Y.A., Asensio-Pérez, J.-I., Martínez-Monés, A. and Osuna-Gómez, C.A. Component-Based Software Engineering and CSCL in the Field of e-Learning. *UPGRADE*, *4* (5). 2003, 21-27.

41. dotLRN Learn Research Network. Available at http://dotlrn.org (last accessed 2 August, 2005).

42. Dufresne, R. Classtalk: A Classroom Communication System for Active Learning. *Computing in Higher Education*, *7* (2). 1996, 3-47.

43. E-LANE. Available at http://e-lane.org/ (last accessed March 2007).

44. E-LEN A network of e-learning centres. Available at http://www.tisip.no/E-LEN (last accessed 26 May 2004).

45. Fablusi The online role-play simulation platform. Available at http://www.fablusi.com (last accessed 12 April 2006).

46. FACILITATE Unleash creativity with focused idea generation. Available at http://www.facilitate.com/productfunctions_brainstorming.htm (last accessed 18 May 2005).

47. Fayad, M., Schmidt, D. and Johnson, R. *Building Application Frameworks Object-oriented foundations of framework design*. Wiley, 1999.

48. Field, J. New directions for lifelong learning using network technologies. *British Journal of Educational Technology*, *35* (6). 2004, 689-700.

49. Frangenheim, E. *Reflections on Classroom Thinking Strategies*. Paul Chapman Educational Publishing, London, 2005.

50. Gamma, E., Helm, R., Johnson, R. and Vlissides, J. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, New York, 1995.

51. Gifford, B. and Enyedy, N., Activity Centered Design: Towards a Theoretical Framework for CSCL. In Proceedings of the *Third InternationalConference on Computer Support for Collaborative Learning* (Deerfield, IL, 1999), 189-196.

52. Gilbert, J.E. and Han, C.Y. Adapting Instruction in Search of a 'Significant Difference'. *Journal of Network and Computer Applications* (22). 1999, 149-160.

53. GNU General Public License. Available at http://www.gnu.org/copyleft/gpl.html (last accessed 22 April, 2006).

54. Goodyear, P. Educational design and networked learning: patterns, pattern languages and design practice. *Australasian Journal of Educational Technology*, *21* (1). 2005, 82-101.

55. Guerrero, L.A. and Fuller, D.A., Design patterns for collaborative systems. In Proceedings of the *String Processing and Information Retrieval Symposium & International Workshop on Groupware* (Cancun, Mexico, 1999), 270-277.

56. Gurteen, D. Knowledge, Creativity and Innovation. *Journal of Knowledge Management*, *2* (1). 1998, 5-13.

57. Hartman, K., Neuwirth, C., Kiesler, S., Palmquist, M. and Zubrow, D. Patterns of social interaction and learning to write: Some effects of network technologies. *Computer Mediated Communication and The Online Classroom*, *2*. 1995, 47-78.

58. Harvey, J. The LTDI Evaluation Cookbook. Learning Technology Dissemination Initiative, 1998. Available at http://www.icbl.hw.ac.uk/ltdi/cookbook/contents.html (last accessed March 2007).

59. Hernández-Leo, D., Asensio-Pérez, J.I., Dimitriadis, Y., Bote-Lorenzo, M.L., Jorrín-Abellán, I.M. and Villasclaras-Fernández, E.D., Describing Effective Collaborative Learning Flows Using IMS Learning Design. In Proceedings of the *4th IASTED International Conference on Web-Based Education* (Grindelwald, Switzerland, 2005), 273-278.

60. Hernández-Leo, D., Villasclaras-Fernández, E.D., Asensio-Pérez, J.I., Dimitriadis, Y., Bote-Lorenzo, M.L. and Rubia-Avi, B., Linking Collaborative Learning Practice with IMS LD and Service-Oriented Technologies: an Approach Based on Collaborative Learning Flow Patterns. In Proceedings of the

6th IEEE International Conference on Advanced Learning Technologies (Kerkrade, The Netherlands, 2006), 1109-1110.

61. Hernández-Leo, D., Villasclaras-Fernández, E.D., Jorrín-Abellán, I.M., Asensio-Pérez, J.I., Dimitriadis, Y., Ruiz-Requies, I. and Rubia-Avi, B. Collage, a Collaborative Learning Design Editor Based on Patterns. *Special Issue on Learning Design, Educational Technology & Society*, *9* (1). 2006, 58-71.
62. Hernández-Leo, D., Villasclaras-Fernández, E.D., Jorrín-Abellán, I.M., Asensio-Pérez, J.I., Dimitriadis, Y., Ruiz-Requies, I. and Rubia-Avi, B. Reusing IMS-LD Formalized Best Practices in Collaborative Learning Structuring. *Advanced Technology for Learning (extended version of the WBE05 paper)*, *2* (4). 2005, 223-232.
63. Hodgson, V. and McConnell, D. Co-operative learning and developing networks. *Computer Assisted Learning*, *4* (11). 1995, 210-224.
64. Hummel, H., Manderveld, J., Tattersall, C. and Koper, R. Educational modelling language and learning design: new opportunities for instructional reusability and personalised learning. *International Journal of Learning Technology*, *1* (1). 2004, 111-126.
65. IMS IMS Global Learning Consortium. Available at http://www.imsglobal.org/learningdesign/ (last accessed 12 May 2005).
66. IMS IMS Learning Design Information Model. Available at http://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html (last accessed 15 May 2005).
67. Ismail, J. The design of an e-learning system beyond the hype. *Internet and Higher Education* (4). 2002, 329-336.
68. James, B., Lefoe, G. and Hadi, M., Working 'through' graduate attributes: A bottom-up approach. In Proceedings of the *Transforming knowledge into Wisdom: Holistic Approaches to Teaching and Learning, Higher Education Research and Development Society of Australasia* (Miri, Sarawak, 2004), 174-184.

69. Jenkins, J., Wayne, S. and Vadasy, P. Cooperative Learning: Prevalence, Conceptualizations, and the Relationship between Research and Practice. *American Educational Research Journal*, *35* (3). 1997, 419-454.

70. Johnson, R.E. Frameworks = (Components + Patterns). *ACM Communication*, *40* (10). 1997, 39-42.

71. Kiesler, S., Siegel, J. and McGuire, T.W. Social psychological aspects of computer-mediated communication. *American Psychologist,*, *39* (10). 1984, 1123-1134.

72. Klein, P.D. Reopening inquiry into cognitive processes in writing-to-learn. *Educational Psychology Review*, *11* (3). 1999, 203-207.

73. Koper, R. Modeling unit of study from a pedagogical perspective the pedagogical meta-model behind EML, 2001. Available at http://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html (last accessed April, 2006).

74. Koper, R. and Manderveld, J. Educational modelling language: modelling reusable, interoperable, rich and personalised units of learning. *British Journal of Educational Technology*, *35* (5). 2004, 537-551.

75. Koper, R. and Olivier, B. Representing the Learning Design of Units of Learning *Educational Technology & Society*, *7* (3). 2004, 97-111.

76. LAMS The Learning Activity Management System. Available at http://www.lamsinternational.com (last accessed 12 March 2006).

77. Laurillard, D., Design Tools for E-learning. In Proceedings of the *19th Annual Conference of the Australian Society for Computers in Learning in Tertiary Education* (Auckland, New Zealand, 2002), 4-5.

78. Laurillard, D. *Rethinking University Teaching*. Routledge Falmer, 2002.

79. Leo, D.H., Perez, J.I.A. and Dimitriadis, Y.A., IMS Learning Design Support for the Formalization of Collaborative Learning Patterns. In Proceedings of the *4th International Conference on Advanced Learning Technologies* (Joensuu, Finland, 2004), 350-354.

80. Liu, X., Saddik, A.E. and Georganas, N., An implementable architecture of e-learning system. In Proceedings of the *Canadian Conference on Electrical and Computer Engineering* (Quebec, Canada, 2003), 110-114.

81. Lord, T.R. 101 Reasons for using cooperative learning in biology teaching. *The American Biology Teacher*, *63* (1). 2001, 30-38.

82. Ltd., I.I. Change your lifeand career with advanced brainstorming. Available at http://www.brainstorming.co.uk (last accessed 12 May 2006).

83. LTSC Learning Technology Standards Committee. Available at http://ltsc.ieee.org (last accessed 22 Jun 2006).

84. Macromedia. Available at http://www.macromedia.com/software/flashcom/ (last accessed 18 April 2005).

85. Marjanovic, O. Learning and teaching in a synchronous collaborative environment. *Computer Assisted Learning* (15). 1999, 129-138.

86. McAndrew, P., Goodyear, P. and Dalziel, J. Patterns, designs and activities: unifying descriptions of learning structures. *International Journal of Learning Technology*. in press.

87. McArdle, G., Monahan, T., Bertolotto, M. and Mangina, E., A Web-Based Multimedia Virtual Reality Environment for E-Learning. In Proceedings of the *Eurographics04* (Grenoble, France, 2004), 9–13.

88. McCreary, E. Three behavioral models for computer-mediated communication. in Haraism, L.M. ed. *Online education: Perspectives on a new environment*, Praeger, New York, 1990, 117-130.

89. McDonald, J. Is "as good as face-to-face" as good as it gets? *Journal of Asynchronous Learning Networks*, *6* (2). 2002, 10-23.

90. Miao, Y., Hoeksema, K., Hoppe, H.U. and Harrer, A., CSCL Scripts: Modelling Features and Potential Use. In Proceedings of the *Computer Supported Collaborative Learning* (Aipei, Taiwan, 2005), 426–432.

91. Millbank, G., Writing multimedia training with integrated simulation. In Proceedings of the *Writers' Retreat on Interactive Technology and Equipment* (Vancouver, BC., 1994), 75-77.

92. Morch, A.I., Cheung, W.K., Wong, K.C., Liu, J., Lee, C., Lam, M.H. and Tang, J.P., Grounding Collaborative Knowledge Building in Semantics-Based Critiquing. In Proceedings of the *Advances in Web-Based Learning* (Hong Kong, China, 2005), 244-255.

93. Neale, D.C. and Carroll, J.M., Multi-faceted Evaluation for Complex, Distributed Activities. In Proceedings of the *Computer support for collaborative learning* (Mahwah, NJ, 1999), 425-433.

94. Nikolova, I. and Collis, B. Flexible learning and design of instruction. *British Journal of Educational Technology*, *29* (1). 1998, 59-72.

95. NSWIS NSW Institute of Sport. Available at http://www.nswis.com.au/splash/default.aspx (last accessed 22 April 2006).

96. O'Connor, C., Sceiford, E., Wang, G., D., F.-S. and Griffin, O. Departure, Abandonment, and Dropout of E-learning: Dilemma and Solutions, The MASIE Center, 2003.

97. Oblinger, D. Will E-business Shape the Future of Open and Distance Learning? *Open Learning*, *16* (1). 2001, 9-26.

98. Olaniran, B.A. Applying synchronous computer-mediated communication into course design: Some considerations and practical guides. *Campus-Wide Information Systems*, *23* (3). 2006, 210-220.

99. Oliver, M. An Introduction to the Evaluation of Learning Technology. *Educational Technology & Society*, *3* (4). 2000, 20-30.

100. OpenACS Open Architecture Community System. Available at http://openacs.org/ (last accessed 17 April 2005).

101. Osuna-Gómez, C.A., Sheremetov, L.B., Romero-Salcedo, M. and Villa-Vargas, L. Creating Collaborative Learning Applications: A Social Constructive Technique. *Advanced Technology for Learning*, *1* (3). 2004.

102. Palloff, R.M. and Paratt, K. *Building Learning Communities in Cyberspace*. Jossey- Bass Inc, San Francisco, 1999.

103. Palmer, G., Peters, R. and Streetman, R. Cooperative Learning, 2003. Available at http://www.coe.uga.edu/epltt/col.htm (last accessed 22 Augest 2006).

104. Panitz, T. and Panitz, P. Encouraging the use of collaborative learning in higher education. in *University Teaching: International Perspectives*, Garland Publishers, New York, 1998, 161-202.

105. Pask, G. Conversational techniques in the study and practice of educcation. *British Journal of Educational Psychology*, *46*. 1976, 12-25.

106. Paulsen, M.F. The Online Report on Pedagogical Techniques for Computer-Mediated Communication. nki, 1995. Available at http://www.nettskolen.com/forskning/19/cmcped.html (last accessed May 2006).

107. PedagogicalPatterns The Pedagogical Patterns Project. Available at http://www.pedagogicalpatterns.org/ (last accessed 22 May 2006).

108. Pfister, H.-R. and Mühlpfordt, M., Supporting discourse in a synchronous learning environmment: The learning protocol approach. In Proceedings of the *Computer Supported Collaborative Learning* (Boulder, USA, 2002), 581-589.

109. Rafaeli, S. and Sudweeks, F. Networked interactivity. *Computer Mediated Communication*, *2* (4). 1997, 1-6.

110. RELOAD Reusable eLearning Object Authoring & Delivery. Available at http://www.reload.ac.uk/ (last accessed 22 Jun 2006).

111. Romiszowski, A.J. and Jost, K.L., Computer conferencing and the distance learner: Problems of structure and control. In Proceedings of the *Helping Learners at a Distance: Annual Conference on Teaching at a Distance* (Madison, Wisconsin, 1989), 131-137.

112. Sakai. Available at http://sakaiproject.org/ (last accessed 12 May 2006).

113. Schneider, D. and Synteta, P., Conception and implementation of rich pedagogical scenarios through collaborative portal sites. In Proceedings of the

Conference on Open and Online Learning (University of Mauritius, 2003), 243-268.

114. Seffah, A.G., P., Learner-centered software engineering education: from resources to skills and pedagogical patterns. In Proceedings of the *Software Engineering Education and Training* (Covington, Kentucky, USA, 2002), 14-21.

115. Shambaugh, N. and Magliaro, S. A Reflexive Model for Teaching Instructional Design. *Educational Technology Research and Development*, *49* (2). 2001, 69-92.

116. Smith, B.L. and MacGregor, J.T. What is Collaborative Learning? in Goodsell, A.S., Maher, M.R. and Tinto, V. eds. *Collaborative Learning: A Sourcebook for Higher Education*, National Center on Postsecondary Teaching, PA, 1992, 10-29.

117. Smith, M. and Lytle, S.L. The Teacher Research Movement: A Decade Later. *Educational Researcher*, *28* (7). 1999, 15-25.

118. Stacey, E., Learning collaboratively online. In Proceedings of the *18th International Conference on Distance Education* (Pennsylvania State University, Pennsylvania, 1997), 1-11.

119. Stacey, P. People to People, not just people to content. Vancouver, 2003. Available at http://www.imsglobal.org/otf/vancouver/presentations/active%20Content/paulStacey-People%20to%20People.pdf (last accessed 18 May 2005).

120. Svetcov, D. The virtual classroom vs. the real one. *Forbes*, *166* (7). 2000, 3-5.

121. Taran, C. Enabling SMEs to deliver synchronous online training – practical guidelines. *Campus-Wide Information Systems*, *23* (3). 2006, 182-195.

122. Tastle, W., B., W. and Shackleton, P. E-Learning in Higher Education: The Challenge, Effort, and Return on Investment. *International Journal on ELearning*, *4* (2). 2005, 241-251.

123. Tattersall, C. and Koper, R. EML and IMS Learning Design: from LO to LA, 2004. Available at http://hdl.handle.net/1820/107 (last accessed 12 May 2006).

124. Turani, A. and Calvo, R., The Potential Use of Collaboration Scripts in Synchronous Collaborative Learning. In Proceedings of the *International Conference on Interactive Mobile and Computer Aided Learning* (Amman, Jordan, 2007), Accepted.

125. Turani, A. and Calvo, R., Sharing Synchronous Collaborative Learning Structures using IMS Learning Design. In Proceedings of the *International Conference on Information Technology Based Higher Education & Training* (Sydney, 2006).

126. Turani, A., Calvo, R. and Goodyear, P., An Application Framework for Collaborative Learning. In Proceedings of the *International Conference on Web Engineering* (Sydney, 2005), Springer-Verlag, 243-251.

127. Turani, A. and Calvo, R.A. Beehive: a software application for synchronous collaborative learning. *Campus-Wide Information Systems*, *23* (3). 2006, 196-209.

128. Turani, A. and Calvo, R.A., Educational design patterns with Beehive. In Proceedings of the *Conference on Education and Technology in the Middle East* (Manama, Bahrain, 2007), Accepted.

129. Vega-Gorgojo, G., Bote-Lorenzo, M.L., Gómez-Sánchez, E., Dimitriadis, Y. and Asensio-Pérez, J.I., CSCL Scripting Patterns: Hierarchical Relationships and Applicability. In Proceedings of the *6th IEEE International Conference on Advanced Learning Technologies* (Kerkrade, The Netherlands, 2006), 388-392.

130. Vega-Gorgojo, G., Bote-Lorenzo, M.L., Gómez-Sánchez, E., Dimitriadis, Y. and Asensio-Pérez, J.I. A semantic approach to discovering learning services in grid-based collaborative systems. *Future Generation Computer Systems*, *22* (6). 2006, 709-719.

131. Vega-Gorgojo, G., Bote-Lorenzo, M.L., Gómez-Sánchez, E., Dimitriadis, Y. and Asensio-Pérez, J.I., Semantic Description of Collaboration Scripts for Service Oriented CSCL Systems. In Proceedings of the *12th International Conference on Artificial Intelligence in Education* (Amsterdam, 2005), 935-937.

132. Walther, J.B. Interpersonal effects in computer-mediated interaction: A relational perspective. *Communication Research*, *19* (1). 1992, 52-90.

133. Wasson, B. and Ludvigsen, S. Designing for Knowledge Building *ITU Report*, University of Oslo Press, Oslo, Norway, 2003.

134. WebCT Web Course Tools. Available at http://www.webct.com/ (last accessed 23 May 2006).

135. webex Web Conferencing, Online Meetings, and Video Conferencing. Available at http://www.webex.com.au (last accessed 12 April 2006).

136. Wessner, M. and Pfister, H.-R., Group formation in computer-supported collaborative learning. In Proceedings of the *International ACM SIGGROUP Conference on Supporting Group Work* (Boulder, Colorado, USA, 2001), 24-31

137. Wessner, M., Pfister, H.-R. and Miao, Y., Using Learning Protocols to Structure Computer-Supported Cooperative Learning. In Proceedings of the *World Conference on Educational Multimedia, Hypermedia & Telecommunications* (Seattle, Washington, 1999), 471-476.

138. Wessner, M. and Pfister, H.R., Points of cooperation: Integrating cooperative learning into web-based courses. In Proceedings of the *New Technologies for Collaborative Learning* (Awaji-Yumebutai, Japan, 2000), 33-41.

139. Westera, W. Paradoxes in Open, Networked Learning Environments: Toward a Paradigm Shift. *Educational Technology & Society*, *39* (1). 1999, 17-23.

140. Whelan, R.R., The Future of Web-based Learning. In Proceedings of the *International Conference on Information Technology Based Higher Education & Training* (Sydney, 2006).

141. Williams, J.B. and Jacobs, J. Exploring the use of blogs as learning spaces in the higher education sector. *Australasian Journal of Educational Technology*, *20* (2). 2004, 232-247.