Implementing collaborative learning activities in the classroom supported by one-to-one mobile computing: A design-based process

Claudio Alvarez *, Rosa Alarcon, Miguel Nussbaum

Computer Science Department, School of Engineering, Pontificia Universidad Catolica de Chile, Chile

A R T I C L E   I N F O

Article history:
Received 24 November 2010
Received in revised form 8 May 2011
Accepted 3 July 2011
Available online 18 July 2011

Keywords:
Mobile CSCL
Framework
Design-based research
Collaborative learning
Domain-specific
Language

A B S T R A C T

Mobile devices such as PDAs, smartphones and tablet computers are becoming increasingly popular, setting out opportunities for new ways of communicating and collaborating. Research initiatives have ascertained the potential of mobile devices in education, and particularly, the benefits of incorporating them in the classroom for eliciting collaborative learning and active student participation. However, the development of technology-supported learning environments poses challenges to education researchers, practitioners, and software technologists in creating educational tools that respond to real needs of instructors and learners, meet clearly defined didactic purposes, and are practical for the intended audience. This article reports on a technology for facilitating the implementation of collaborative learning environments in the classroom supported by one-to-one mobile computing. The approach encompasses a framework supporting the design and implementation of the mobile software, and a design-based process that guides interdisciplinary efforts utilizing the framework, towards creating effective pedagogical models based on collaborative learning. The proposed design-based process allowed us to develop pedagogical models that respond to real needs of learners and instructors, where development is grounded on rigorous scientific research, allowing to reuse both knowledge and software, and showing an improvement of the mobile software built based on continuous experimentation and evaluation. A case study illustrating the application of the technology is presented and plans for future research are discussed. © 2011 Elsevier Inc. All rights reserved.

1. Introduction

Mobile devices such as smartphones, PDAs, netbooks and tablet PCs are becoming increasingly popular worldwide, exerting a great influence in the way people communicate and access information (Borcea and Lammitchi, 2008; Sharples et al., 2008). These trends have motivated research in a variety of novel applications in mobile learning over the last decade (Frohberg et al., 2009; Pea and Maldonado, 2005), taking advantage of user and device mobility for facilitating learning across multiple contexts, involving different locations, tasks and modes of interaction among users (Sharples et al., 2008). However, mobile technologies have been found to be beneficial for supporting learning activities in the classroom (Roscelle et al., 2010; Zurita and Nussbaum, 2007). Mobile devices have been used in classrooms to support note taking and presentation-support systems (Anderson et al., 2004; Kam et al., 2005), formative assessment tools (Cortez et al., 2005; Valdivia and Nussbaum, 2009), games (Spikol et al., 2008), participatory simulations (Yin et al., 2007), and problem-solving activities (Looi and Chen, 2010; Nussbaum et al., 2009). Particularly, wireless handheld devices (e.g. PDAs and smartphones) and tablet computers, featuring touch screens and digital ink-based input, appear as ideal means for supporting students in collaborative learning activities (Alvarez et al., 2009; Zurita and Nussbaum, 2004).

The wide adoption of mobile devices in society, together with the potential benefits of mobile devices in supporting learning, open the possibility of students using mobile devices in the classroom to engage in lesson plans comprising individual and collaborative learning activities (Dillenbourg and Jermann, 2006; Hernandez-Leo et al., 2006). Clearly, the systematic involvement of mobile devices in the classroom must serve for sound pedagogical purposes and must therefore rely on learning patterns of proven effectiveness (Hernandez-Leo et al., 2006). Depending on the objectives of the learning scenarios, instructors may need to combine and customize different learning patterns (e.g. problem-solving methodologies, formative assessment tools, participatory simulations, etc.) into consistent lesson plans towards the desired learning outcomes (Dillenbourg and Jermann, 2006).

Designing and implementing new kinds of learning activities supported by mobile technologies is an interdisciplinary activity that poses challenges for educational technology researchers and practitioners (Spikol et al., 2009). In this context, one prospect for supporting the development of educational technologies is

* Corresponding author. Tel.: +56 2 9344074.
E-mail addresses: cgalvare@ing.puc.cl (C. Alvarez), ralarcon@ing.puc.cl (R. Alarcon), mm@ing.puc.cl (M. Nussbaum).
0164-1212/ – see front matter © 2011 Elsevier Inc. All rights reserved.
to utilize a design research approach. The design-based research paradigm (Reeves, 2006; The Design-Based Research Collective, 2003; Van den Akker et al., 2006) aims at studying complex problems in real educational settings, by blending empirical educational research with theory-driven design through systematic design and study of instructional strategies and tools. Design experiments are typically test-beds for innovation, conducted in a limited number of settings under highly interventionist practices, and supported by iterative design processes guided by the emergence and testing of conjectures and hypotheses (Cobb et al., 2003).

Design-based projects have advocated following participatory design strategies involving stakeholders from the education domain in the technology design team (Van den Akker et al., 2006). This results in the need to provide a common vocabulary that can support interdisciplinary discourse towards creation of shared knowledge and technology products that are purposeful and effective for users. Particularly, development of mobile software for supporting learning activities in the classroom requires that software developers can understand the important concepts in the domain of interest, and effectively apply them to design and implementation of the pedagogical tools.

Model Driven Engineering (MDE) reduces the gap between problem and software implementation domains through a systematic transformation from problem-level abstractions to software implementations (France and Rumpe, 2007). Based on these principles, Model Driven Architecture (MDA) defines three successive abstraction levels and transformation processes starting with a Computer-Independent Model (CIM) capturing, in this context, general pedagogical principles and learning environment constraints. The CIM is transformed into a Platform-Independent Model (PIM) that captures specification issues that do not change from one technological platform to another, and is transformed into a platform specific model (PSM) that grounds platform-independent issues to a targeted platform. A PIM provides a conceptual perspective and captures domain specific functionality (Tchoumi, 2008).

A software development framework that leverages the concepts and interrelations defined by a PIM for a problem domain and presents them to developers as readily available abstractions through a Domain-Specific Language (DSL) can foster software reuse and nurture software maturity (France and Rumpe, 2007). A DSL is a language that provides notations and constructs tailored to a specific application domain, trading generality for expressiveness and ease of use compared with general-purpose programming languages. DSLs foster productivity, diminish maintenance costs and reduce the amount of domain and programming expertise needed, opening up the application domain to a larger group of software developers. Examples of widely used and highly productive DSLs are Latex, the typesetting system based on a markup language, and SQL, the language that allows the manipulation of databases (Mernik et al., 2005). By means of the DSL, the abstract concepts defined by a PIM model can be instanced through concrete derivatives (e.g. class libraries), either provided by the framework as reusable components, or created by software developers based on cohesive Application Programming Interfaces (APIs), and so cover the implementation of specific functional requirements. In addition, the use of a DSL can facilitate development of automation tools that can assist the end-users (i.e. teachers) in preparing lesson plans that incorporate the instructional designs customized to fit specific pedagogical needs.

This article reports on a technology for facilitating the implementation of collaborative learning environments in the classroom supported by mobile devices (Section 2). The approach is detailed in Section 3, it encompasses a software development framework supporting the implementation and operationalization of the mobile software (Section 3.1). The framework is derived from a PIM model that captures the semantics of the collaborative mobile application (Section 3.2.1) and serves as well as the basis for a DSL and the specification of CSCL scripts (Section 3.2.2). The framework, the DSL and the CSCL scripts, facilitates rapid development of mobile applications, on top of a reusable middleware that encapsulates groupware functions suited for face-to-face collaborative learning activities supported by mobile devices.

A design-based process that guides interdisciplinary efforts utilizing the framework, towards creating effective pedagogical models based on collaborative learning is also presented in Section 4. The process supports domain experts, teachers and software developers collaborating towards designing, implementing and evaluating instructional designs supported by mobile technologies. A case study, detailed in Section 5, demonstrates the application of the framework and the process. Section 6 presents conclusions and future work prospects.

2. Mobile software for supporting collaborative classroom activities

Mobile learning activities have been prominently designed for learning settings different than the classroom (Frohberg et al., 2009); Nevertheless, the low cost and ease of use of mobile devices makes their integration into everyday classroom routines an attractive option (Borcea and Iamnitchi, 2008). According to (Parsons et al., 2007) the most promising feature of mobile learning contexts is that one can collaboratively perform activities. In point of fact, research in learning technologies for the classroom, and in particular, developments in the field of Mobile Computer-Supported Collaborative Learning (MCsCL) have provided evidence of collaborative learning activities supported by mobile devices being effective in classrooms (Roschelle et al., 2010; Yin et al., 2007; Zurita and Nussbaum, 2007).

The learning outcomes of collaborative activities depend upon the extent to which groups actually engage in productive interactions, such as explanation, argumentation/negotiation and mutual regulation (Dillenbourg et al., 2009). Moreover, it has been demonstrated that collaboration is an ability that must be learned and requires practice (Fischer et al., 2006). Hence, to be successful, collaborative learning needs to be supported by adequate scaffolds, no matter whether it is a face-to-face experience (e.g. in the classroom) or a distance learning setting (Dillenbourg et al., 2009).

One way of ensuring that learners engage in fruitful collaboration is to engage them in structured interactions, based on prescribed rules establishing how they should form groups, collaborate, and solve problems (Nussbaum et al., 2009; Mulholland et al., 2009). A formal specification prescribing such learning activities is known as a collaboration script (O’Donnell and Dansereau, 1992). Research in the field of Computer-Supported Collaborative Learning (CSCL) has explored the possibility of using computers to support learners in following scripts towards achieving effective collaboration and positive learning outcomes (Dillenbourg et al., 2009). This latter form of computer-mediated collaboration scripts is known as CSCL scripts.

CSCL scripts inherit modeling concepts from activity-centered design (Dillenbourg, 2002; Engeström, 1987). Activity-centered design in mobile learning conceives learning happening in a network of activities involving different social planes (i.e. individuals, groups, the whole class), tasks, division of labor, rules and roles, and intended learning outcomes (Zurita and Nussbaum, 2007; Uden, 2007). This design theory is well suited for collaborative learning.
scenarios, because it considers social interactions as an integral design component, as opposed to learner-centered design theories in which the focus is on the individual (Uden, 2007).

Different research initiatives have aimed at formalizing domain knowledge of collaborative learning (Brecht et al., 2002; Dillenbourg and Hong, 2008; Miao et al., 2007). However, few efforts have been done towards leveraging this knowledge for facilitating development of mobile software to support collaborative learning in the classroom (Frohberg et al., 2009). An effort aiming at operationalizing the relationship between scripts and technology settings is presented by Tchounikine (2008), where he identifies five perspectives of analysis that include, a structural model describing the script's components, that is, a meta-model, and a formal grammar; an implementation model describing how the script is put into practice, including issues such as group formation policies, task sequencing and orchestration (e.g. data flow and workflow ruling the access to individual data and/or functionalities/tools, etc.); student-oriented models related to training learners on understanding scripts and utilizing the platform on which they are operationalized; a platform specification, which establishes the technical specifications that the technological setting must comply with to operationalize the CSCL script, and the underlying design rationale encompassing a learning hypothesis, pedagogical principles and design decisions.

Tchounikine lays the ground for identifying fundamental components for the design of a flexible script engine at a conceptual level, but neither proposes a structural model in detail nor a specification for a platform. Furthermore, his approach overlooks the fact that the development of instructional designs is of an interdisciplinary nature, thus requiring appropriate integration of different roles (i.e. mainly software developers and domain experts) consistently within a software development process. Miao et al. (2007), on the other hand proposes a detailed structural model, in Tchounikine's terms, that is a general specification language to describe CSCL collaboration scripts. The model main contributions are the explicit introduction of a group entity conveying organizational and behavioral roles, an artifact entity modeling information flow, an action and expression entities as a declarative mechanism for detailing collaborative learning processes, transition and routing activities as to enable the specification of complicated control flows at various levels, and most importantly, a change in the focus from the metaphor of theatrical play and role-part to an activity-centered approach for assigning roles and hence model social interaction.

Activity state, on the other hand, is addressed in a very general fashion in Miao's model. The state is manipulated by means of value getters and setters within predicates relating the model's elements. Nevertheless, no specific details are addressed about a client-server architecture covering state synchronization requirements between multiple clients and the server. Furthermore, no explicit event handling classes and constructs are considered in the proposed model. In point of fact, both activity state management and event handling are of upmost importance in mobile settings, generally requiring sophisticated approaches to deal with numerous possible exceptions at runtime, usually related to network instability and battery power. According to Koole (2009), effective mobile learning, results from the integration of the device, learner, and social aspects, into three criteria, namely, the device as a dynamic mediator of interaction between learners, their environment, their tools and information; the capability of information access and selection, which cannot be taken for granted since networking is typically unreliable in mobile settings affecting also connectivity, document sharing and collaboration tools; and the capability of knowledge navigation and production, which are also restricted by mobile devices affordances such as physical characteristics, input and output mechanisms, information storage and retrieval, processor speed, and error handling. Even though Miao’s declarative control flow constructs provide adequate programmability for dealing with deterministic workflow execution, workflows in mobile settings must afford to explicitly incorporate both synchronous and asynchronous event handling, so as to implement suitable management of network and hardware exceptions. Moreover, asynchronous event handling is necessary whenever loosely coupled interaction scenarios between groups or individuals are considered (Herskovic et al., 2009).

Koole’s, Tchounikine’s and Miao’s are conceptual proposals and hence, they do not explore whether they are suitable in real scenarios for supporting the development and adoption of their approaches. Moreover, the extent to which learning technologies succeed greatly depends on the quality of its adoption by instructors and students (Dillenbourg et al., 2009). Further efforts are required towards providing instructors software tools that facilitate creating lesson plans based on collaborative learning activities supported by mobile devices, as well as means for easily enacting the activities in the classroom. This scenario has motivated us to integrate and extend existing proposals by shifting the focus of mobile devices from information access devices to mediators that enable the execution of collaborative scripting, and by extending current proposals in order to provide support for mobile scenarios. This includes a proper handling of state information, synchronization and optimized use of networking capabilities in order to support collaborative tools and document sharing. Our approach aims to hide software development complexity by relying on a Model Driven Architecture that operationalize CSCL design by means of a Domain Specific Language exposing domain entities and constructs. Finally, an effective platform requires not only software technology for facilitating the implementation of MCSCL environments for the classroom, but also a process that takes into account field experts i.e. instructional designers and developers in an incremental and iterative approach that fosters shared understanding, and reuse.

3. Software technology for enabling CSCL scripts supported by mobile devices in the classroom

Forms of classroom teaching today ascribe to longstanding education traditions dating back to many centuries ago. In these traditions, the teacher has a prominent role as an enlightener, while the students are most of the time expected to work individually in their tasks while interactions with others are discouraged. Contrastingly, current demands of labor and citizenship demand that individuals are trained to develop their social skills, so as to engage in work teams properly, while also being capable of searching information effectively through interacting with new media (Jenkins, 2006). This has motivated the emergence of research trends in the pursuit of interactive learning environments that bring elicitation of qualities related to active, participatory and collaborative learning. The domain of CSCL has exerted prominent influence in current innovations in education, conceiving novel learning environments for fostering active learner involvement, participation and collaboration. During the past decade, numerous research initiatives have proposed theoretical analysis and conceptualization of collaborative learning activities mediated by computer software (Brecht et al., 2002; Dillenbourg and Hong, 2008; Miao et al., 2007; Zurita and Nussbaum, 2007), and particularly, means to support collaboration based on scaffolds, such as CSCL scripts. Based on the domain theories and conceptual models proposed by (Brecht et al., 2002; Dillenbourg, 2002; Miao et al., 2007; Zurita and Nussbaum, 2007) and our previous experience developing face-to-face collaborative learning activities supported by mobile devices, a CSCL script can be broadly described as a medium through which certain social interactions are expected to occur, towards completion of...
well-defined learning objectives. These social interactions are structured through a set of activities through which instructors and/ or learners individually or collaboratively perform creation and actualization of knowledge, through operating on individual or shared artifacts (i.e., data objects), assuming different roles and following specific rules and division of labor, towards fulfillment of specific pedagogical objectives (Dillenbourg and Hong, 2008; Miao et al., 2007).

In our research agenda, we envision the development of pedagogical models for the classroom supported by mobile devices aiming to encourage student participation, active learning, and collaboration (Roschelle et al., 2010; Valdivia and Nussbaum, 2009; Zurita and Nussbaum, 2007). Our pedagogical models based on one-to-one mobile computing are targeted at both K–12 and higher education classrooms; thus, they are constrained to the technological and social conditions that are afforded in these environments. The basic constraints and conditions upon which our pedagogical models are formulated are the following:

- Students as task performers: Students are responsible for their own work, role, and efforts to learn within the group (Roschelle et al., 2010). Each task has a goal that requires students working individually and/ or collaboratively towards its completion.
- Autonomous, offline execution: Unavailability of internet connectivity must not be a barrier to executing learning activities in classrooms. Therefore it must be possible to operate the learning environment relying on local area wireless networks, and resources available in the participating mobile devices (Zurita and Nussbaum, 2004; Sharples et al., 2008).
- Tightly coupled student interaction: Collaborative learning activities require the coordinated effort of students in small groups towards fulfilling a common goal (Roschelle et al., 2010). Group member's actions affect the other members' actions significantly and regularly, and the members need to communicate with each other verbally and with visual contact to manage the interdependence of their actions (Nussbaum et al., 2009). Mobile devices features such as limited battery life, energy consumed when connected to the network and synchronization tasks are directly affected when used as a platform for these kind of interaction.
- The teacher as a guide: The teacher defines the learning goals and supports the students and groups in achieving them through face-to-face verbal interactions (Dillenbourg et al., 2009b).
- Task-driven lesson planning: There's a need for the teacher to present students with a sequence of learning tasks to be solved individually and/or collaboratively, following a particular sequence order (Tchounikine, 2008). Unexpected events (e.g., suspending an activity altogether) must be considered in classroom settings, hence a platform enacting a lesson plan must support a rich set of control flow constructs as well as asynchronous communication and events.
- Monitoring and control: The teacher requires a visualization that reports how the different students and groups are performing in their learning tasks, and tools to control the flow of learning activities (Alvarez et al., 2009; Zurita and Nussbaum, 2007). This requires keeping track of the state of students' activities, which increases the traffic in the network.
- Technology as a scaffolding: The instructional design rely on technology as a means for enforcing rules, coordinating role assignment, managing division of labor and providing the students with visualizations of their shared workspaces (Nussbaum et al., 2009; Roschelle et al., 2010).

Developing a pedagogical model based on these conditions requires that an educational problem or need is clearly identified. The identification of the problem or need will require stating an associated research problem and defining it upon a theoretical foundation. In turn, this will lead to research hypotheses and conjectures aligned with the problem, and the implementation of the software tools necessary to validate such hypotheses experimentally (Roschelle et al., 2010). Consistently, the development of the software tools is of an evolutionary nature, driven by the outcome of the experimental validation of the ruling research hypotheses. This brings challenges in the software development process because requirements are continuously discovered through experimentation and evaluation and therefore a quick response to changes in software requirements is required.

Our strategy for coping with the latter software development challenges relies on reducing software complexity by providing developers a framework that permits specifying a CSCL script as a declarative composition of modular components (i.e., plugins) based on domain-level abstractions, and a middleware that implements common groupware functions found in collaborative applications, such as session management, peer and group management, communication, synchronization and coordination services (Bouassida-Rodriguez et al., 2009). In this way, developers reuse domain knowledge and core groupware functionality, aiming their efforts at designing and implementing features strictly concerned with the requirements of the instructional design. On the following sections, we describe the architecture of the software development framework.

3.1. Framework architecture

Fig. 1 depicts the architecture of the framework. The architecture comprises mobile software that operates in the classroom, and a Learning Management System (LMS) that is used by teachers, researchers and other stakeholders off-classroom, to download software to their mobile devices, prepare class material, lesson plans and generate reports on students’ performance. The mobile software consists of a mobile client that runs on the students’ devices, and a mobile server that may run on the teacher’s mobile device or on dedicated hardware. In the interventions in the classroom, the former alternative was preferred, because of the ease of setup, low cost and minimal technological expertise required.

Both mobile client and server architectures comprise three layers. The top layer (application) provides software developers domain-level abstractions in the form of an interpreted Domain-Specific Language (DSL) (Gasevic et al., 2009), the middle layer (middleware) provides groupware functions including synchronization, coordination, content and awareness components, supporting tightly coupled interaction. The bottom layer addresses low level concerns such as networking and communications functionality. The LMS is preferably implemented as a web application so that it can be accessed by different institutions and other stakeholders not directly involved in the classroom environment, thus facilitating the sharing and reuse of pedagogical content generated by education practitioners, as well as providing access to reports on academic performance.

3.2. Application development

In both mobile client and server architectures, the top layer of the framework provides an API based on PIM model depicted in Fig. 2. Developers can use this API to implement CSCL scripts taking advantage of high-level abstractions that act as a façade to the functionality and services provided by the lower layers of the framework. Both client and server side applications are developed based on this model.

3.2.1. PIM model-based script modeling

According to the PIM model shown in Fig. 2, a script is composed by set of activities following a particular sequencing order.
An activity can be independent and self-contained, hosting an entire learning task, or it can be part of a process containing more related activities. Each activity is related to an environment, which provides the user interface. Data flow between activities and environments is driven by actions and events. Actions are basically controllers available in both client and server APIs, which implement script rules, process user input, control the data flow and prepare the visualization of data, having access to the functionality and services available in the framework’s middleware layer. Actions control the user interface objects hosted in an environment, and these objects notify their state to actions via events. This strategy provides greater flexibility since it decouples visualizations from the control logic.

Transitions between activities in a script may be sequential, conditional, or arbitrary. A script will typically combine different transition schemes. Sequential transitions can be used for executing activities following a strict order, which is commonly used, e.g. for implementing pedagogical designs comprising ordered sets of tasks. Conditional transitions are used to selectively invoke the next activity based on fulfillment of a predicate. With arbitrary transitions, the next activity to be executed may be directly decided by the user, triggered by an event or by an automatic mechanism, for instance, remotely controlled from another device in the network. In a collaboration environment, the latter form of flow control is used; the server communicates, coordinates and synchronizes interrelated activities executed in different devices. Event-driven
activities makes possible to support asynchronous interaction and error handling. Transitions are implemented by means of expressions that relate transition operators and control flow constructs with variables. Variables hold information resulting from the execution of actions and events, and their values are evaluated within expressions.

Under the assumption that the framework is based on a client-server architecture, the server keeps track of the state of the students’ and teacher’s clients, thus the state of peers and groups in relation to the activities and the script itself are considered in the ‘ScriptState’, ‘ActivityState’, ‘GroupState’ and ‘PeerState’ classes of the PIM model shown in Fig. 2. Clients report actions and events to the server which holds state classes for all the clients. Each script must implement explicitly its state components which are stored in the server at various levels of aggregation, namely individual (peer), group, activity and script. The server synchronizes clients via periodic multicast notifications. Clients only accept the information relevant to each one and ignore notifications directed to other clients. The incoming state information is processed at the application level, so that state changes are visualized on the user interface. This task is performed by the lower layers of our middleware and is transparent to developers but can be customized if required. The approach allows devices that have loose connection or need to be replaced during the execution of the learning activities to rejoin the session and recover their state seamlessly. This strategy allows also to provide persistence to activities on the server-side so that tasks such as monitoring the progression of the learning activity at the student and group sizes can be performed by teachers, and even makes possible to force the termination of an activity providing more control for instructors so they can adapt to the situation at hand.

Both individual and collaborative learning activities often require users to generate or manipulate preexisting or emerging Content units. Learning objects are deemed as encapsulated content that is consumed within activities and cannot be modified, whereas user-generated content is dynamically produced through Artifacts within the context of activities (Lejeune et al., 2009; Miao et al., 2007). Activities within a script may access learning objects or artifacts under the assumption that these are always localizable and available, either locally stored in a persistent (i.e. database) or transient form (i.e. cache in process memory), or stored in a remote device. Learning objects and artifacts are addressable by Uniform Resource Identifiers (URIs) or by tags (i.e. “current question”, “current answer”, “last answer” and so on). Data manipulation operations are implemented by Actions, which can be defined as short-lived processes within activities that are driven by user interaction or triggered by automations. Actions may interoperate with environments providing resources such as learning objects or user interface controls.

Students (or peers) are roles, activities requiring student interactions in groups may specify a GroupConfiguration object. The schema specifying how students should be grouped may be predefined before the script is executed, or random with equally sized groups (Zurita et al., 2005). Usually the latter form is preferred because it relieves teachers from the responsibility of specifying the group composition manually and handling exceptions such as latecomers or absent students. Random arrangement facilitates handling such exceptions because no particular student order is expected. In addition, it results in the benefit of creating groups composed by students with different abilities and levels of academic achievement, which is considered beneficial in small group learning activities (Nussbaum et al., 2009; Zurita et al., 2005).

3.2.2. Specifying scripts with a DSL

The software composition approach offered by the framework is based on a DSL derived from the PIM model depicted in Fig. 2. The DSL is based on a XML schema (see Table 1), thus, software developers use the DSL to specify a CSCL script by using XML tags that correspond to objects present in the PIM model. Typically, software developers write code in the DSL along with code in an object-oriented language, e.g. for implementing actions or user interface elements, and encapsulate the object-oriented code in plug-ins, which they can invoke from the DSL (Gasevic et al., 2008; Nierstrasz and Achermann, 2000). Hence, the DSL in practice allows developers assembling an application by instances and configuring native objects (i.e. provided by the framework) and objects provided by plug-ins, in a declarative manner. Once developers have generated code for a CSCL script using the DSL, they can develop administrative tools for a LMS, such as editors and configurators, so that teachers can then integrate the CSCL script into lesson plans utilizing a user-friendly interface. Code generation can be therefore effectively automated in a way that is transparent for teachers and non-programmers.

Scripts based on the DSL can be executed in both mobile client and server parties of the framework architecture by a Player (interpreter). The Player contains the framework middleware, and a DSL interpreter, which can dynamically load any components required by a script at runtime.

4. A development process for collaborative activities supported by mobile devices in the classroom

The Integrative Learning Design (ILD) framework (see Fig. 3) (Bannan-Ritland, 2003) is based on design research principles, and provides a comprehensive, yet flexible, guiding framework that positions design research as a socially constructed, contextualized process, for producing educationally effective interventions with a high likelihood of being used in practice. The ILD has been used as a guiding process in interdisciplinary efforts among education researchers, practitioners and software engineers in the research and development of novel collaborative learning applications for the classroom based on mobile devices (Roschelle et al., 2010) ILD comprises the following four phases: (1) Informed Exploration: identifying needs, studying theories and audience, and defining problems. (2) Enactment: prototyping, implementing and testing the intervention iteratively. (3) Local Evaluation: evaluating the impact of the intervention on its stakeholders and going back to the previous phase if required. (4) Broad Evaluation: publishing and disseminating the findings and addressing concerns regarding the adoption of the intervention by a broader audience.

In utilizing the ILD framework towards developing a pedagogical design for the classroom supported by mobile devices, the proposed process presented in Fig. 4 is based on the experiences reported by (Cortez et al., 2005; Roschelle et al., 2010; Zurita and Nussbaum, 2007). The process comprises 8 stages within the phases established by the ILD Framework, and involves a research team composed by four roles: (1) Researcher: His/her background is related to computer science, education or cognitive psychology and he/she is concerned with the conceptualization and design of the instructional design, its validation and evaluation. (2) Software Developer: He/she is responsible for requirements analysis, software implementation and testing. (3) Teacher: He/she integrates the instructional design into lesson plans (possibly involving other instructional designs) and customizes it with appropriate resources (i.e. content), according to the desired educational outcomes. (4) Student: He/she is actively involved in the learning process and is subject to evaluation for assessing the instructional design’s impact in learning.

In stage 1 of the process (see Fig. 4), researchers conduct a literature survey in the state of the art of learning theories and cor-
responding experimental work, seeking to establish a conceptual basis upon which to define the problem or need. Afterwards, on stage 2 researchers collaborate with teachers in defining the characteristics of the desired CSCL script. The script is specified based on criteria defined by (Dillenbourg, 2002). From the didactical perspective, researchers define the didactic rationale of the script, how it should structure interactions between learners (i.e. coercion degree), how it should be appropriated by teachers and learners, and the extent to which it may be generalizable to different learning domains. Developers assist researchers in defining the instructional design from an operational perspective; namely, describing the set of activities, including the task definition, group composition criteria as well as group changes and regrouping, mechanisms for distribution of activities, resources and roles, modes of interaction (i.e. synchronous/asynchronous, individual/collaborative), and timing (i.e. whether the set of activities can be contained within a single lesson or it spans across lessons, weeks, semesters, etc.). Developers also assist researchers on identifying proper mobile devices affordances considering technical aspects such as physical characteristics (size, weight), input and output capabilities (depending on student’s age), storage requirements, response time (depending on the complexity of the user interface), and error handling. Other issues such as portability, connectivity and activity length helps to determine if the activity is feasible considering battery life and network coverage. The complexity of activities and collaborative authoring tools also impact the user interface design and issues such as aesthetics, metaphors, transparency, cognitive load and I/O mechanisms must be addressed at a design level in this stage (Koole, 2009). The output of Stage 2 is documentation presenting a detailed specification of the CSCL script, including Unified Modeling Language (UML) activity diagrams and user interface mockups of the mobile software (Booch et al., 2007).

Table 1
DSL tag definitions. Each XML tag is briefly described, as well as the attributes it supports, and its children tags.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Attributes</th>
<th>Children tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;script&gt;</td>
<td>The root element of an ELFML script.</td>
<td>id</td>
<td>&lt;meta&gt;, &lt;include&gt;, &lt;teacher&gt;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;student&gt;, &lt;activity&gt;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;environment&gt;</td>
</tr>
<tr>
<td>&lt;meta&gt;</td>
<td>Adds metadata to the script, in the form of [key, value] pairs.</td>
<td>key, value</td>
<td>&lt;plugin&gt;, &lt;file&gt;</td>
</tr>
<tr>
<td>&lt;include&gt;</td>
<td>Specifies plugin and external script dependencies.</td>
<td>source</td>
<td></td>
</tr>
<tr>
<td>&lt;file&gt;</td>
<td>Specifies a file (e.g. external script) that can be included within the declaring script.</td>
<td>name, source</td>
<td></td>
</tr>
<tr>
<td>&lt;plugin&gt;</td>
<td>Includes a plugin that can extend the script engine functionality.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;activity&gt;</td>
<td>Declares an activity. If the entrypoint attribute is set, the script execution starts from that activity.</td>
<td>id, name, isTemplate, environmentId,</td>
<td>&lt;action&gt;, &lt;actionseq&gt;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>entrypoint, baseTemplateId</td>
<td>&lt;eventListener&gt;, &lt;operator&gt;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;variable&gt;, &lt;groupConfiguration&gt;</td>
</tr>
<tr>
<td>&lt;action&gt;</td>
<td>Declares and configures an action.</td>
<td>id, name, pluginId, class, activationMode</td>
<td>&lt;param&gt;</td>
</tr>
<tr>
<td>&lt;actionseq&gt;</td>
<td>Declares an action sequence to be executed within an activity.</td>
<td>id, name, activationMode, automatic</td>
<td></td>
</tr>
<tr>
<td>&lt;param&gt;</td>
<td>A configuration parameter in [key, value] format.</td>
<td>key, value</td>
<td></td>
</tr>
<tr>
<td>&lt;environment&gt;</td>
<td>Specifies an environment.</td>
<td>id, name, isTemplate, baseTemplateId</td>
<td>&lt;contentUnit&gt;, &lt;uiControl&gt;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;eventListener&gt;, &lt;operator&gt;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;groupConfiguration&gt;</td>
</tr>
<tr>
<td>&lt;variable&gt;</td>
<td>Declares a variable and optionally sets its initial value.</td>
<td>name, value, type</td>
<td>&lt;variable&gt;</td>
</tr>
<tr>
<td>&lt;operator&gt;</td>
<td>Declares and configures an operator. The &lt;param&gt; child elements can specify the operands and the variables where to store results.</td>
<td>name, value, type</td>
<td></td>
</tr>
<tr>
<td>&lt;teacher&gt;</td>
<td>May contain one or more &lt;activity&gt; child elements to be executed in the teacher's script.</td>
<td>id, pluginId, class</td>
<td></td>
</tr>
<tr>
<td>&lt;student&gt;</td>
<td>May contain one or more &lt;activity&gt; child elements to be executed in the students' script.</td>
<td>id, pluginId, class</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 3. Integrative learning design process.
In Stages 3A and 3B researchers and software developers design the mobile software required to support the CSCL script and the administrative tools needed to operationalize it; namely, collaborative authoring tools for creating educational content, configuration editors for customizing the operation of the instructional design, evaluation tools for obtaining reports on students’ performance indicators, and integration with learning activity systems for indexation, sharing, archival and searching of instructional designs and every other mobile affordances addressed in the previous stage. For designing the mobile software, developers elaborate UML static structure diagrams based on the PIM model depicted in Fig. 2, so that they can identify the components of each activity in the script and their interrelationships, estimate the necessary effort required to implement the components, and evaluate the feasibility of reusing or adapting components used in previous instructional designs. In companion to the UML static structure diagrams, developers model client-server interactions by means of UML sequence diagrams, which are later encapsulated in the action components that are identified in the static structure diagrams.

The software developers incrementally implement the mobile software and administrative tools for the intervention on stages 4A and 4B, relying on the DSL and middleware provided by the software development framework presented earlier. The administrative tools are implemented as modules for the LMS (see Fig. 1) using an SDK that facilitates development of editors and customization utilities required by teachers to integrate the CSCL script in lesson plans, and operate the lesson plans in the classroom. Preliminary tests are conducted in stage 4C in order to verify and validate the software against the functional requirements, as well as assessing whether it performs satisfactorily. These tests are conducted in a restricted MCSCL environment, involving the software developers, researchers, and comprising a number of mobile devices running the instructional design corresponding to the number of students in a typical classroom (e.g. 45 in a Chilean classroom). Software developers are responsible for taking notice of defects found on the tests and fixing them until the software is mature enough for conducting tests with real teachers and students.

In stages 5A and 5B the researchers design the experiments to be held during the intervention and work on the monitoring and evaluation plan. This plan establishes the diagnostics, work plan and training for teachers, and parameters for assessing the fidelity of the implementation of the instructional design in the experimental scenario, in relation to the objectives of the intervention. Teachers receive training on operating the mobile software and its administrative tools after the monitoring and evaluation plan is prepared (stage 5C). Before an intervention is conducted in the target setting, a formative evaluation is carried out (stage 5D), in which the software is tested with real teachers and students, seeking to ensure that the software operates properly and that the teachers are able to operate it by themselves. The evaluation is also intended to assure that students understand the instructions given to them on how to operate the mobile software and have the required cognitive maturity for the tasks. As the software is tested for the first time with real teachers and students on stage 5D, software defects may be discovered, such as performance-related deficiencies, user interface design flaws and malfunction of features, all which are potentially
5. Application of the ILD-based process: CollPad

Research in collaborative learning applications supported by mobile technologies has been motivated by the need to foster active student participation and development of social skills in the classroom, contrasting with traditional lecturing in which the students tend to be a passive audience and the active role is prominently exerted by the instructor (Nussbaum et al., 2010). Responding to this need, the CollPad script was developed as a tool for increasing student participation by means of eliciting shared knowledge construction, negotiation and discussions in small groups and involving the whole class with the teacher’s mediation. The following sections present an account of the development process of CollPad, based on the process and software technology described previously.

5.1. Stage 1: Literature survey and preliminary studies

Studies from (Zurita and Nussbaum, 2004) have validated the use of mobile devices (i.e. PDAs) as a suitable means for supporting face-to-face collaborative learning activities in small groups, overcoming difficulties that commonly arise in these settings, such as coordinating and synchronizing learners’ interactions, supporting negotiation and discussion, and organizing and distributing learning resources (Zurita and Nussbaum, 2007). Based on these outcomes, Cortez et al. (2005) and Valdivia and Nussbaum (2009) investigated the use of Multiple Choice Questions (MCQs) as a pedagogical model for face-to-face collaborative learning, developing a collaboration script supported by mobile devices. The collaboration script proposed by Cortez et al. (2005) and Valdivia and Nussbaum (2009), also known as Consensus (Roschelle et al., 2010), requires students to share knowledge for solving sets of MCQs, while it provides the instructor feedback that permits him/her adapt his teaching to better address learner’s needs.

The Consensus script has been proven to have a positive impact in learners’ motivation and academic results (Nussbaum et al., 2010; Roschelle et al., 2010) However, its success in assessing upper-level cognitive processes according to Bloom’s scale (Anderson et al., 2001), such as remembering, understanding, applying, analyzing, evaluating and creating depends on the quality of the MCQs used (Paxton, 2000). Regarding limitations in the use of MCQs, the ‘forced choice’ nature of MCQs does not lead to an accurate assessment nor provide insight of what students really are thinking as they develop their answers to the questions. Moreover, the use of MCQs presents limitations in developing and assessing creativity, i.e. cognitive processes in which elements are put together to form a coherent or functional whole, or reorganizing elements into a new pattern or structure (Anderson et al., 2001; Veeravagu et al., 2010).

The need to overcome Consensus’ limitations regarding the ‘forced choice’ nature of MCQs, and further stimulate student participation not only at the level of small groups level but also at classroom level, motivated the design of CollPad (Nussbaum et al., 2009) as a constructivist collaborative problem solving activity based on open-ended task. Contrastingly to the MCQ-based approach of Consensus, CollPad aims at giving learners complete ownership of the process to construct solutions for the given task, encourages testing individual knowledge construction against alternative views based on the premise that knowledge is constructed through different social planes (i.e. individual, small groups and the whole class), and provides opportunity for reflection on both the content learned and the learning process by means of teacher-guided classroom discussions. From the technology usability standpoint, motivation for developing CollPad was based on the opportunity of assessing whether mobile affordances related to ink-based input is a proper means for supporting collaborative work in the classroom based on open-ended tasks.

5.2. Stage 2: Instructional design specification

The design rationale (Dillenbourg, 2002) of the CollPad script is to encourage social interactions towards constructing shared understanding of open-ended tasks, as well as the processes involved in elaborating their solutions (Nussbaum et al., 2009). Social interactions are structured with a high degree of coercion: that is, students and groups follow a strict sequence of phases coordinated and synchronized by the wirelessly networked mobile devices, with the purpose of prompting cognitive and social interactions between the participants involved, which might otherwise not occur (Nussbaum et al., 2009). In addition, the script is performed by the technology so that learners follow it without need to memorize it, thus appropriation of CollPad does not demand a significant cognitive load on learners. CollPad aims at supporting a high degree of generalizability, by affording teaching and learning in any knowledge domain in which knowledge representations can be generated using paper and pencil.

The flow of activities of the CollPad script is indicated in Fig. 5. Initially, students are randomly assigned to small groups (preferably of 3 students), indicated on their devices’ screens. Once groups are organized, the teacher delivers the students an open-ended task. In every group the students work on the given task individually, writing a solution with digital ink-based input (i.e. a stylus) on a virtual piece of paper displayed on their handheld devices (Individual Response phase in Fig. 5). When all the students in a group finish writing their solutions, the system shows all group members a visualization containing their individual answers (Collective Decision phase in Fig. 5). Students have to submit in agreement one of the available answers, or choose to write a new answer collectively.
Applications submitted in 1970 to the Universidad de Chile, which implement a structured modeling approach for educational software development. The process involves several phases:

1. **Problem Statement**
2. **Individual Answer**
3. **Collective Decision**
4. **New Proposal**
5. **In-Class Discussion**
6. **Conclusion**

**Fig. 5. Phases of the CollPad script.**

(NEW PROPOSAL PHASE IN FIG. 5). The script will urge the students to come to an agreement if they do not agree. If the students choose to write a new solution, the script randomly assigns the scribe role to one of the learners in the group, while his/her companions become reviewers. The scribe writes a new solution with which the reviewers must agree on accepting. If the reviewers do not accept the new solution, the script randomly chooses another learner to become the scribe and the process is repeated.

As the groups submit their solutions, the teacher can evaluate them and select a subset of them to start a whole class discussion (IN-CLASS DISCUSSION IN FIG. 5). When the discussion starts, the teacher may call for a vote on the selected answers, and pick random students from each group, who must defend the answer submitted by their groups verbally. In this way, answers are analyzed and debated by the students with the teacher’s mediation, until the whole classroom agrees on a final response to the original problem (CONCLUSION). From the instructional design perspective, this process does not require storing content on the mobile devices but requires that the students’ choices can be shared, as well as for the teacher to coordinate students progress on the task. Students that will use the application are K12 and university undergraduates which are familiar with mobile devices UI widgets.

5.3. **Stage 3: Software Design**

CollPad was initially developed as a course project in the Mobile Applications Workshop in the School of Engineering of Pontificia Universidad Catolica de Chile. The software development team was composed by three undergraduate computer engineering students. In the span of one semester, the students were able to design and implement CollPad using the framework described on Section 3.

The methodology used for designing the mobile software supporting the CollPad script is illustrated in Fig. 6. The design process involved creating an activity model consisting of UML activity diagrams describing the flow of activities in both teacher’s and students’ scripts. Based on the description of the script developed in the previous stage by education researchers and the activity model, the software developers produced complete user interface mock-ups for both scripts, in order to visually identify the script’s software components. Finally, a structural model for the scripts based on the PIM model in Fig. 2 was created by the software developers, in order to specify the software components required for implementing the scripts and their relationships.

Mockups of the Individual Answer and Collective Decision phases of the students’ script are shown in Fig. 6. On the Individual Answer phase, the learner writes his/her solution for the given task, i.e. “Draw a map of South America”, using a virtual sketchpad, and on the Collective Decision phase, all the solutions generated by the group members are displayed as eligible options, so the group members must agree unanimously on submitting one of the answers already generated, or choose to generate a new one. If the choice is not unanimous, a message urging the learners to agree is displayed, as shown on the mockup of the Collective Decision phase in Fig. 6.

Having created complete user interface mockups for the teacher’s and students’ scripts, and validated them with education researchers, software developers created a structural model (i.e. class diagram) based on the PIM model described in Fig. 2. Based on the description of the Script, the activity diagram, and user interface mockups, software developers could identify the components of activities and environments in the script and specify them as derived classes of the PIM model presented in Fig. 2.

For initial evaluation purposes, a LMS was not required for CollPad, thus there was no need of developing companion administrative tools (e.g. task editors, report generators). Teachers were expected to always rely on creating tasks on-the-fly in the classroom using the mobile device, rather than using pre-generated content stored in a LMS. Due to our state handling mechanism, participants generated content (e.g. question made by teachers, answers given by students) must be locally cached in the devices.

5.4. **Stage 4: Software Implementation**

The identification of script components provided by the structural model, and the elaboration of UML sequence diagrams for modeling of client-server interactions, were used as a basis for implementing the script code utilizing the DSL. The example in Fig. 7 presents a sequence diagram depicting client-server interaction in the Collective Decision phase of the CollPad students’ script. The objects involved in the sequence diagram are activities executing on the client and server sides, the continuous message lines indicate invocation of actions within activities, and the dashed message lines indicate events being raised. In the sequence shown, a student selects one of the alternatives on his/her device, and his/her choice is submitted to the server, i.e. the submitChoice action of the ‘CollectiveDecisionProcessing’ activity is invoked from the ‘CollectiveDecision’ activity. The student’s choice is then processed by the
Fig. 6. Software design methodology.

Fig. 7. Sequence diagram depicting client-server interaction in the Collective Decision phase.
'updateState' action in the 'StateManagement' server activity and the global state of the script is updated accordingly and notified by the server to all the clients. The students' clients react depending on the result of the state update; if all the students in the group have made their choices and they have come to a unanimous decision, the Collective Decision activity ends, and the following activity is decided depending on whether the students chose to write a new answer (i.e., invoke the 'New Proposal' phase) or not. Otherwise, if the students' choices are not unanimous, the sequence must be repeated, i.e., the sequence loops until the group reaches consensus. Analogously, the teacher's client is updated with the script state, so that the teacher can visualize the group progress.

Fig. 8 presents a simplified version of the CollPad students' script. Lines 2 and 3 specify the plug-ins required by the script and lines 5 and 20 declare the 'StateProcessing' and 'CollectiveDecision' activities involved in the sequence previously explained. Each activity declares the actions and event handlers required locally by the activities, as well as for client-server interaction. The 'CollectiveDecisionEnvironment' declared on line 13 contains the user interface controls utilized in the 'CollectiveDecisionActivity'. User interface controls and actions such as the 'ProcessStateUpdateAction' on line 8 are declared with a qualified name indicating the plug-in to which they belong.

For implementing the plug-ins required by the CollPad scripts, developers could re-use the middleware functionality provided by the framework so their work was focused on creating plug-ins incorporating the logic (i.e., actions) required by the script, and implementing support for ink-based artifacts by means of proper user interface controls. The framework used was based on Windows Mobile, and the development tools included Microsoft Embedded C++ version 4.0, and the Microsoft Pocket PC 2003 software development kit. The test bed for the software was based on 30 mobile clients and one mobile server, being all devices HP iPaq h4150, interconnected over an infrastructure-based IEEE 802.11b wireless network.

Prior to performing an evaluation with students and teachers, a usability analysis was conducted with education and psychology researchers. It consisted on informal conversations where researches used the software and spontaneously commented on learnability, that is, how hard was to follow actions described by the workflow and whether the depth of the navigation tree was appropriated. This lead to simplifications to the user interface that enhanced usability for instructors. The approach was to make the software as simple as possible, minimizing the number of steps necessary to use each feature and simplifying onscreen instructions.

5.5. Stage 5: Evaluation preparation

Local evaluation of CollPad was planned for K-12 and university level education, with the purpose of validating CollPad's pedagogical model and technological approach on different demographics and knowledge domains. At K-12 level, CollPad trials were planned for three schools in the UK, involving five 6th grade teachers (Galloway, 2007), and in Chile with three math high school teachers in two schools (Nussbaum et al., 2009). At university level, CollPad trials were planned for two computer science courses, i.e., Knowledge Management and Human–Computer Interaction, imparted at the School of Engineering of Pontificia Universidad Catolica de Chile (Alvarez et al., 2009).

Concerning educational aspects, local evaluation pursued validating whether CollPad's constructivist scaffolding could effectively foster collaborative work towards effective resolution of open-ended tasks, elicit verbalization of ideas and negotiation of meaning in small group and class group levels, and provide instructors insight into learners' thinking. Regarding technological aspects, local evaluation sought assessing the use of PDAs as a means for supporting collaborative work based on open-ended tasks, and determining whether instructors could adopt and integrate the technology into their daily teaching.

Preparations for conducting local evaluation on CollPad included elaborating a plan for instructor training, emphasizing on instructors' practice in exerting the role of mediating and guiding group and classroom discussions. In addition, clear guidelines and examples for the design of tasks were prepared by education researchers, in order to facilitate the instructor's work in creating adequate tasks for CollPad. Observation forms and questionnaires for surveys and interviews were elaborated as part of the monitoring and evaluation plan for the intervention (Alvarez et al., 2009, 2011), intended to adequately capture users' feedback for assessing CollPad's fulfillment of its expected educational value.

Prior to conducting evaluation with the target audiences, CollPad was subject to formative evaluation with K-12 and university-level students for 5 weeks, with teachers and students recruited for these testing purposes exclusively, i.e. they did not participate in further evaluations conducted later. Evaluation consisted of usage of the system and informal feedback, verbally reporting functionality failures and usability limitations. Software defects encountered were mostly related with server-side synchronization of concurrent transactions and state processing, which caused clients to adopt states inconsistent with the script design. Scalability issues were also detected, as users experienced notorious latency in the responsiveness of the user interface with an increasing number of mobile devices in the classroom. These issues could be solved by properly serializing server-side transaction processing, and optimizing the use of network bandwidth by compressing the ink-based artifacts exchanged by the mobile devices.

5.6. Stage 6: Lesson plans creation

The instructors involved in the trials created tasks for CollPad in the different subjects that were covered, based on the guidelines and examples prepared by education researchers. The tasks were designed by the instructors focusing on strengthening the students' abilities in understanding and critically analyzing scenarios based on concepts taught in class. In most cases, more than one answer to the task was acceptable, thus the instructors were responsible for steering the in-class discussions towards converging to the best possible answer.

5.7. Stage 7: Lesson plans evaluation

The Knowledge Management and Human–Computer Interaction courses imparted at the School of Engineering of Pontificia Universidad Catolica de Chile (Alvarez et al., 2009), consisted of about 16 three-hour lectures, that lasted for one semester and involved 27 and 28 undergraduate senior students, respectively. 12 CollPad sessions were conducted on the Knowledge Management course, and 9 in the Human–Computer Interaction course. CollPad activities were carried out in the courses in almost every class, always using the same methodology. Groups of three students were formed randomly from among those present, which would then have to respond to a question set by the professor on a subject related to the material being covered in that class. Once all of the groups had submitted their answers, the professor selected the four answers that appeared to be the most interesting and the system randomly named a student from each group whose response was so chosen to defend it. The entire class then voted on the answer they believed was the correct one. As an incentive the professor awarded extra marks to the members of the four groups whose responses were selected, the highest award being given to the group whose answer received the most votes.
To measure the quality of the CollPad activities, we defined four variables with scores high, medium and low, namely, amplitude indicating the number of different viewpoints that may be present in the analysis of the activity; discussion-oriented, defining whether the activity allows for discussion of the solution among the students in which they can develop their answers, or whether the solution is exact and therefore not open to discussion; coverage, indicating how much of the course material intended for reinforcement is covered by the activity; and scope, indicating whether the scope of the question is merely conceptual or encourages the students to apply the material in other contexts. Twenty-one activities were evaluated considering both courses, more than 60% of the activities scored high on at least three of the four variables.

In addition, we used an open questionnaire to evaluate students' views on CollPad's usability. The questionnaire presented four questions and was performed some time after the second of the two course experiences with the system to ensure there would be no relationship between the students' course performance and their opinions on the collaborative activities. 47 students were involved in CollPad's experiences but only 29 answered the questionnaire. Questions included, what is CollPad?, name up to 4 attributes of CollPad, indicate your positive and negative experiences with CollPad activities carried out in classes, under what circumstances would you recommend the use of this experience in these classes?

38% of answers for the first question mentioned only CollPad's technical characteristics, 38% referred to its collaborative aspects and 24% cited both. For the second question, students mostly reported collaboration, group work, conversation and analysis attributes, attributes linked to evaluation were infrequently mentioned, or not cited at all. For the third question, the majority stressed as positive those experiences related to the system's purpose (e.g. collaboration, group work, learning). Most of the negative characteristics they cited had to do with technical difficulties (e.g. network problems, complicated interface, small screen). As for the negative conceptual aspects, most of them involved the particular methodology used in the courses, and especially the method of evaluation, rather than the main concepts underlying the CollPad model. Finally, for the fourth question, the attributes most frequently named were: development of ability to analyze, group work learning, and development of social and collaborative skills.

The trials in K-12 schools involved class groups with 20 to 45 students and lasted for one month, including three schools in the UK and involving five 6th grade teachers for a month (Galloway, 2007; Whitley, 2007). In Chile three math high school teachers were involved in two schools also for a month. Children had experience using PocketPCs for some time, however, CollPad required a couple of hours of training for teachers in managing the software's operation sequence, and the connection to the wireless network. Teachers had direct methodological support in the first session, and on average, after four trials in front of the students they felt comfortable with the methodology and the tool that supported it. In UK CollPad was used from Math to Art to English with great fluency, in Chile it was used in Math and Science.

Teacher training in the K-12 trials required a couple of hours in managing the sequence of operation of CollPad, and the
connection of the machines to the wireless network. On the other hand, the instructor involved in the trials in computer science courses was closely involved in the design of CollPad; therefore, he did not require prior training. K–12 teachers had direct methodological support in the first session; the guidance was mainly directed to the In-Class Discussion Phase. On average, after four trials in front of the students they felt comfortable with the methodology and the tool that supported it. In all trials, technical support was required to set the router and the machines. In the UK and Chile K–12 trials the children in the trial had experience of using the PocketPCs for some time, and so they were familiar with the basic operation of the device. In the university level trials, students only required briefing and assistance during the first CollPad session.

In the month of the K–12 trials, all the teachers could integrate the technology into their daily teaching. They understood the pedagogy behind the model, and learned how to develop appropriate content to foster discussion. Both K–12 and university level instructors reported that there was greater involvement of all pupils in discussion using the CollPad script, which they found impossible to achieve using teacher moderation alone. Both the instructors and students highlighted that students learned to give their personal opinion on topics they would never normally tackle, to converge in a discussion and be an active learner inside the classroom. Instructors observed that CollPad offers insight into learners’ thinking. After group work finishes the instructor instantly receives their outputs, allowing close tracking of progress. Some indicated that in-class management is therefore improved helping the instructor to plan their interventions in the class, and target his/her time.

5.8. Stage 8: Dissemination of results and findings

Results of experiences evaluating CollPad’s educational value in K–12 and university levels have been reported in previous works. For instance, in (Nussbaum et al., 2009), we applied design-based research to the design and validation of CollPad. The resulting system was tested in classrooms in UK and Chile, showing that the model is welcomed both by teachers and pupils, and meets its objectives of ensuring greater interaction between class members who do not normally work together, and fostering students’ participation in discussion based activities. In (Alvarez et al., 2009), we reported on the testing of the abovementioned system in two computer science courses imparted in the School of Engineering of Pontificia Universidad Catolica de Chile; namely, Knowledge Management and Human-Computer Interaction courses. The system’s objectives was to support the teaching of regular CS subjects as well as to foster the development of students’ communication and social skills. Qualitative results of the experience showed that students found the tool effective in creating an environment that promotes communication, interpersonal and decision-making skills. In (Alvarez et al., 2011), we reported on a comparative study that explored the potential of tablet-style and regular netbook computers in the classroom, for determining which of the two device types is more suitable for work with the proposed system. Our findings indicate that tablet PCs strengthen collective discourse capabilities and facilitate a richer and more natural body language, were preferred to netbooks and that tablet’s digital ink and paper promoted greater self-confidence in expressing ideas when compared to netbooks’ traditional vertical screen and keyboard arrangement.

The described experiments evaluated CollPad’s impact on students, instructors and the effect of using different mobile devices in the operational settings. The results obtained have led to ascertaining CollPad’s capacity of fostering a participatory environment in the classroom and providing instructors insight into learners’ thinking, while fostering the development of social skills through face-to-face collaboration and teacher-guided discussions.

5.9. Stage 9: Develop plan for broader audience

The advent of inexpensive netbooks aimed at education (Alvarez et al., 2011) and their massive adoption in emerging markets has motivated plans for conducting a large-scale evaluation of CollPad in K–12 education. A version of CollPad aimed at the Intel Classmate netbook computers (Cramer et al., 2009) is currently being developed for broad evaluation in Latin American schools, which includes integration with a LMS allowing teachers to create and share tasks for CollPad, as well as evaluating the results of the lesson plans.

CollPad is being recommended by the School of Engineering of Pontificia Universidad Catolica de Chile to its staff members, as a valuable pedagogical practice that they can adopt for fostering active and collaborative learning of engineering subjects. Guidelines on the use of CollPad and reports on the experiences conducted in computer science courses have been divulged in the School of Engineering, with the purpose of promoting its adoption for daily teaching.

5.10. Reuse of instructional and software design

Instructional design, middleware and software developed based on the framework can be reused for creating new applications. For instance, knowledge, abstractions, instructional design and experience obtained in CollPad’s design and development were reused in the design of CollBoard facilitating the tasks of instructional designers by providing them with a setup and experience to rework and extend. The middleware provided by the framework, as well as digital-ink compliant user interface components previously developed for CollPad were reused, thus facilitating the implementation of CollBoard’s script. Since scripts’ transitions guide the data and control flow it hard to take a subset of a script, written in XML, and reuse it as is within a different script. Further work will deepen into scripts design in order to increase modularity and reduce data flow dependencies so that composition of software by straightforwardly reusing script modules can be possible.

Mobile devices with digital ink-based input have been ascertained as an effective means for guiding the students throughout the knowledge construction process that is afforded by CollPad, as well as supporting CollPad’s need of a generic means for constructing knowledge representations based on handwriting and drawings. However, in this latter regard, digitally augmented classroom media, such as digital pens and interactive whiteboards, can be easily integrated in the classroom offering seamless interaction capabilities (Steimle et al., 2008), thus opening the possibility of making the technology supporting CollPad’s knowledge construction process transparent for the users.

The CollBoard script (Alvarez et al., 2010) was conceived based on CollPad, defining a problem-solving process that involves individual work supported by digital pens and instructor-mediated discussions supported by interactive whiteboards. Students use digital pens and traditional paper for generating answers to the open-ended task, so their answers are then evaluated by the instructor on paper or on a computer, and later presented in the interactive whiteboard for an instructor-mediated discussion. By using the interactive whiteboard in the discussion, the students may collectively construct a new answer to the task combining elements from the individual answers, create an entirely new answer or work on a new task proposed by the instructor.

6. Conclusions and future work

Technology does not have an intrinsic effect on learning outcomes; rather, technology can be used as a tool supporting teaching
methods that are effective towards the educational objectives and target audience (Dillenbourg, 2008). The development of novel learning environments is therefore an activity that requires close collaboration between education researchers, technologists and the end-users of the technology. The joint interdisciplinary efforts must lead to designing pedagogical models based on real needs of instructors and learners, and implementing technological tools that meet the pedagogical purposes and are practical and productive for the intended users. Under these conditions, the experience advocating design-based research strategies, and particularly, the ILD Framework, has been positive towards creating sound pedagogical strategies for face-to-face collaborative learning supported by mobile technologies. Advocating these practices, real user characteristics, needs and problems have been considered, the solutions have been grounded on rigorous scientific research, and the solutions are improved based on continuous experimentation and evaluation.

The development of pedagogical models based on design-based methodologies, such as the ILD Framework, is of an iterative nature, demanding continuous experimentation in the target educational settings, and evaluation of the experiences for ensuring that the interventions meet the desired educational objectives. Consequently, development of software tools supporting the pedagogical models is driven by requirements of an evolutionary nature, as pedagogical models evolve through design-based experimentation and evaluation. It is therefore valuable that the domain knowledge relevant to software development can be made available to software developers in the form of reusable software assets, based on a coherent set of abstractions representing concepts and rules defined by the problem domain. In this way, software design is accomplished resting on a consistent body of fundamental knowledge, and the resulting software architecture is predictable, given that the PIM model defines the structural components of the software and their relationships. Therefore, the software architecture can be easier to understand for software developers.

Our middleware can provide support for activities outside the classroom, hence exploiting location and context awareness. However, we have focused so far on learning activities in the classroom environment. Consequently, greater emphasis has been given to mobile features dealing with dynamic configuration of groups (message interchange within networked sub-groups), and the constraints posed by the use of mobile devices such as network bandwidth, unreliable connectivity, state handling, synchronization, ad-hoc networks, and limited battery life. The emphasis is placed also in handling non-deterministic flow of activities, as the flow could be interrupted or modified according to the evolution of the class. This flexibility is supported by event-based control flow support.

Domain abstractions (PIM) and flexible software composition afforded by domain specific languages appear as a viable option for achieving the abovementioned benefits. However, abstractions must assure a sensible degree of completeness and consistency in their capacity of reflecting the problem domain. The software development framework presented on this paper is based on analysis of research literature in instructional design theories relevant to the domain of collaborative learning activities in the classroom (Miao et al., 2007; Zurita and Nussbaum, 2007) and considers assumptions of the researchers based on their own experience developing learning environments for the classroom based on mobile technologies. Nevertheless, this does not deny the possibility of extending the framework to support scenarios that further leverage user mobility, such as learning activities that combine outdoor and indoor locations (Spikol et al., 2008), activities based on augmented reality (Liu et al., 2010), and pervasive learning applications (Spikol et al., 2009). In the future, design-based processes for creating pedagogical models supporting these scenarios will be conducted, and the software development framework will be extended to enact them.

Acknowledgement

Research supported by the Center for Research on Educational Policy and Practice, Grant CIE01–CONICYT.

References


Dillenbourg, P., 2008. Integrating technologies into educational ecosystems. Distance Education 29 (2), 127–140.


Claudio Alvarez received a PhD from the Pontificia Universidad Católica de Chile. His research interests include collaborative systems, model-driven engineering and design-based research methods in education.

Rosa Alarcón is a full professor of computer science at Pontificia Universidad Católica de Chile. Her research interests include software engineering, human-computer-interfaces and multiagent systems. She received a Ph.D. from the Pontificia Universidad Católica de Chile.

Miguel Nussbaum is a full professor of computer science at the School of Engineering of Pontificia Universidad Católica de Chile. His research areas are technology-enhanced learning and knowledge management. He received his Ph.D. in Technical Sciences from the Eidgenössische Technische Hochschule (ETH), Zurich, Switzerland.