UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA DE POSTGRADO

Basis for a Methodology to Define, Validate and Apply Best Practices in a Computer-integrated Classroom

Submitted to the Departamento de Ciencias de la Computación of the Universidad de Chile in fulfillment of the thesis requirement to obtain the degree of Ph.D. in Computer Science

JENS HARDINGS PERL

Advisors:	Nelson Baloian Tataryan
	Ulrich Hoppe
	José A. Pino Urtubia
Examiners:	Luis A. Guerrero Blanco
	Sergio Ochoa Delorenzi
	Julita Vassileva

Santiago, Chile

March 2006

Dedicatoria

A todos los que me acompañaron en esta aventura. Especialmente a Carolina, quien estuvo junto a mi en cada paso.

Agradecimientos

Este es un proyecto que no es posible emprender solo. Durante el desarrollo de mi tesis recibí ayuda de más fuentes de las que podía imaginar. No me es posible agradecer en estas líneas a todos, así que pido disculpas a quienes involuntariamente he dejado fuera.

Agradezco a mis profesores guía, Nelson Baloian, Ulrich Hoppe y José Pino, con quienes pude contar no sólo en lo profesional sino también en lo personal. Al Servicio Alemán de Intercambio Académico (DAAD) que permitió dedicación exclusiva a mi trabajo durante un período importante, a través de Maria Hartmann que siempre estuvo dispuesta a ayudar. Al grupo Collide de la Universität Duisburg-Essen que me acogió en un ambiente muy propicio, especialmente a Ulrich Hoppe, Katrin Gassner, Marc Jansen, Andreas Lingnau, Niels Pinkwart, Lars Bollen y Andreas Harrer. A la Escuela de Ingeniería y particularmente al Departamento de Ciencia de la Computación de la Pontificia Universidad Católica de Chile, en especial a los profesores Rosa Alarcón, Marcelo Arenas, Felipe Csaszar, Jorge Díaz, Yadran Eterovic, David Fuller, Domingo Mery, Jaime Navón, Miguel Nussbaum, Marcos Sepúlveda y Álvaro Soto, por ayudarme a finalizar la tesis en un grato ambiente. También a José Miguel Piquer, quien despertó mi inquietud de realizar el doctorado.

Por sobre todo agradezco a mis padres que siempre me apoyaron de todas las formas que pudieron. Y a Carolina, por su paciencia, involucramiento y ser fuente de motivación.

Resumen

El uso de Tecnologías de Información (TI) dentro de salas de clases se está haciendo cada día más difundido. Sin embargo, que el uso de computadores y tecnologías de información en general tenga un impacto positivo en la educación está lejos de ser directo. Es necesario utilizar la tecnología de manera que realmente mejore el proceso de aprendizaje de alguna forma.

Variadas iniciativas apuntan a proveer una forma de saber qué metodologías y aplicaciones de TI en la educación entregan mejores resultados, presentados en "Mejores Prácticas". Sin embargo, estas mejores prácticas, específicamente en el área de la educación, están generalmente basadas en investigación anecdótica y descriptiva. La consecuencia es que esa experiencia es reunida e interpretada en un proceso opaco que no es susceptible de mejoras secuenciales. Cualquier mejora debe venir de la misma fuente o partir reuniendo la experiencia desde el principio.

Salas de clase equipadas con computadores e instalaciones similares son ambientes de enseñanza y aprendizaje donde las TI son usadas principalmente como herramienta de modelamiento y presentación. Como un efecto secundario, estos sistemas generan un alto volumen de información que puede ser capturado con el objetivo de proveer retroalimentación al profesor y apoyar la tarea de seguir una metodología pedagógica determinada. El desafío es encontrar correlación entre propiedades medibles dentro de esa información y los resultados de aprendizaje, definiendo mejores prácticas que pueden ser aplicadas a situaciones específicas o generales.

De acuerdo a este trabajo, es posible utilizar los datos disponibles dentro de un típico ambiente de sala de clases esquipada con computadores con la finalidad de obtener información objetiva y comparable que permita al profesor determinar cómo la tecnología está siendo empleada. El uso de un sistema de consultas durante este proceso le permite al profesor realizar medidas y monitoreo durante una sesión en la sala de clases, posibilitando que aplique y valide mejores prácticas que han sido definidas previamente.

Abstract

Usage of Information Technology (IT) in Classrooms is becoming every day more widespread. However, a positive impact of the usage of computers and information technology in general on the learning process is far from straightforward. It is necessary to apply the technology in a way that will improve the learning process in some way.

Several initiatives aim at providing a way to know which methodologies and applications of IT in education deliver best results, embodied in "Best Practices". However, these best practices, specifically in the educational arena, are generally based on anecdotal and descriptive research. The consequence is that the experience is gathered and interpreted in an opaque process that is not susceptible of sequential improvement. Any improvement must either come from the same source or start gathering the experience from the beginning.

Computer-enabled classrooms and similar settings are teaching and learning environments where Information Technologies are used mostly as modeling and presentation tools. As a side effect, these systems generate a fair amount of information that can be seized in order to give feedback to the teacher and support the task of following a predefined pedagogical methodology. The challenge is to find correlations between measurable properties in that information and the learning outcomes, defining Best Practices that can be applied to specific or general situations.

According to this work, it is possible to use the data available inside a typical computer-enabled classroom in order to obtain objective and comparable information that can allow a teacher to determine how the technology is being employed. The usage of a querying system during this process can enable the teacher to perform measurements and monitoring during a classroom session, being able to apply and validate Best Practices that have previously been defined.

Contents

List of Figures

Chapter 1

Introduction

This thesis presents methodology and tools for the monitoring and evaluation of face to face classroom environments with the objective of defining, applying and validating Best Practices in those environments. This chapter starts with the motivation for this work, followed by the hypothesis, specific and global objectives, and describing the main contributions.

1.1 Motivation

Improving learning has been a goal of many efforts throughout history. This goal has been pursued using a number of methodologies sustained by diverse theories. Today people are capable of using technology as a tool to continue improving education and thus, learning, in concordance to widely accepted pedagogical methodologies. Technology can be applied in different scenarios, such as improving distance learning, or supporting face-to-face activities inside a classroom.

While e-learning can improve the distance learning scenario, it also has to face tremendous challenges. In a distance learning scenario there is no need for gathering at the same time and

	Co-located (same place)	**Remote** (different places)
Synchronous (same time)	Face-to-face conversation, Electronic Meeting Rooms	Telephone, Instant Messaging
Asynchronous (different time)	Post-it note, Shared filesystem	Letter, E-mail

Figure 1.1: The Space-Time Matrix

place, and asynchronous activities can be carried out where every person involved can participate at a different place and time frame than all others (Hiltz and Wellman, 1997). But it is necessary to overcome the absence of a very rich face-to-face interaction by some information that can be transmitted and reproduced for each participant. With the possibility to perform learning activities in a co-located, synchronous (see table 1.1) scenario like a face-to-face classroom, the role of technology is not focused on restoring the channels that are unavailable in the remote situation (Mulder et al., 1997; Zurita and Nussbaum, 2004). This allows us to focus on other aspects where technology can contribute to make the learning experiences an ever richer and effective resource.

Using the Space-Time Matrix (figure 1.1) defined by Dix et al. (1998), the activities inside a classroom, whether they include the use of computing devices or not, can be classified into the Co-located, Synchronous category, but it is possible to find activities in any other of the four categories. For instance, when students are doing homework, the space is remote and the time is asynchronous.

In all cases, Courseware (Sparks et al., 1999; Retalis et al., 1997; Ochoa Delorenzi, 2002) is an important resource that can have significant impact on the learning outcome. Therefore it should be a main objective to enable teachers and students to have access to courseware that helps them not only to reach the learning goals, but to improve in the methodology they use to reach them.

1.1.1 Benefits of using IT in classrooms

The major benefit of using Information Technologies, and thus digital media, inside classrooms is that both, teachers and students, can take advantage of its ease of manipulation (reproduction, reuse, editing, versioning, etc.). Teachers can prepare relatively complex material in advance, reuse them in other courses and contexts with or without modification, and give copies to students for them to edit and learn through experimentation. The digital media is alive in the sense that it encourages the users to modify and use it in creative ways, in contrast to the use of digital content as a one-way information carrier. Media that has been used commonly, like ink on paper, does not support interactive communication as well, due to its lack of flexibility and reuse. When using digital media, it is a matter of choice and not a restriction inherent to the nature of the materials.

The manipulability of the digital media is a powerful tool that can be used to improve the in-classroom learning experience in many ways. The most direct is the time that can be saved in some activities, either by automating actions or allowing the sharing of data among students. Other improvements originate facilitating interactions that are otherwise cumbersome or poorly scalable in larger groups. In this context, the experience of the Computer-integrated Classroom (CiC) implementation in the NIMIS project (Lingnau and Hoppe, 2002; Tewissen et al., 2001; Hoppe et al., 2000) is a good example, in which the "reading through writing" methodology (Reichen, 1991) of teaching to read and write can be applied with the computers taking over part of the burden that normally is left to the teacher.

Additionally, it is possible to exploit natural advantages of digital media, by saving a complete session to use later-on, change sessions or contexts in a glimpse or modeling complex systems easily. A teacher can save the contents of a whiteboard at any time, including several "pages" of annotations, drawings and modeling, keeping this way a record of the whiteboard state at that precise moment. That state can then be restored at any later instant, as if no time had passed between the moment the state was saved and the instant it was restored. A teacher might use this functionality to prepare a complex state in advance, replacing the time-consuming task of re-building the content on the whiteboard with a simple instruction to load the prepared state.

The potential of using advanced technologies to fulfill routine tasks in the classroom is very large and would allow the teacher to concentrate more on pedagogical aspects, knowledge management and support for special needs. However, the change of focus demands teachers to acquire new skills and methodologies that allow them to perform their task in an efficient way, foresee problems and guide discussion threads. Taking these changes into account, the tools available in the classroom should allow the teacher to perform the activities in the best possible way, without restrictions on pedagogical methodology or learning styles. The computer support should be perceived as a tool, not as a guidance of how to structure the session content. This idea is summarized by Norman in (Norman, 2002, page 193): *In general, I welcome any technological advance that reduces my need for mental work but still gives me the control and enjoyment of the task. That way I can exert my mental efforts on the core of the task, the thing to be remembered, the purpose of the arithmetic or the music.* **I want to use my mental powers for the important things, not fritter them away in the mechanics**. (emphasis added).

Limits to usage of Technology

As useful as the use of technology might be inside a classroom, there are several reasons in which it might be desired to limit the use made of technological aids. A teaching - learning setting that incorporates technology should be as simple, in the best case even simpler, as the traditional setting. Otherwise, the usefulness of the technology will compete with the extra burden of learning to use it or managing the available features. While it is possible to organize the features in a way that trivial operations are trivial and complex operations are possible, a user might feel intimidated by a system that shows so many possible interactions.

Also, a system that does not allow students to make mistakes will not necessarily improve the students understanding of the subject. Faced with such a system, a student will have a high chance of ending up solving mechanically the tasks proposed to her, just trying to perform the next possible step(s), since it will always be right. This way the student will solve every proposed problem, but probably without understanding the reason why a problem cannot be solved in some other way. Therefore, the student has to be able to make mistakes along the problem resolution, effectively learning why a proposed solution is not viable in that particular problem. It should be possible to find a balance between the students freedom to elaborate their own reasoning and the help they get from technological or human resources, looking for maximizing the benefit of the learning experiences during the available time frame.

Role of the Teacher in the Classroom

Pedagogy should not be implicitly redefined by technology, but rather the technology should adapt to the various roles a teacher may play in a classroom according to several pedagogical methodologies. In the words of Chizmar and Williams, *The pedagogy must drive the choices of instructional technology, not the other way around.* (Chizmar and Williams, 1997). In this way, it is possible for major changes to take place in the pedagogical arena by providing tools that create new possibilities for innovation. It is important that the technological improvements do not force to embrace a certain learning theory, and instead open the possibility to innovate on the way learning takes place (c.f. Scardamalia and Bereiter, 1996). In Computer-integrated Classrooms and similar environments, the role of the teachers changes dramatically as they move away from being active information transmitters to become “classroom information managers”. It has already been seen that, specially in primary school, teachers act as managers of distributed activities and a variety of resources, as *facilitators* (Rogers, 1969). The tools provided to these teachers have to adapt to the various needs each teaching style will demand.

In a setting that includes Information Technology inside a classroom, a lot of interactions take place and useful information about those interactions is gathered routinely. However, that data on its own has little value if it cannot be used constructively. Since each person has a limited capability of attention, it is not possible to make use of all the potential of the available information. So, in order to improve the human ability in managing such face-to-face situations, using specialized tools will help handle a larger amount of information in a more efficient way. This will help the involved persons to be better connected to the reality than would be possible without external help. Teachers could be made aware of more details to determine if students are unchallenged or overstrained.

1.1.2 Challenges to widespread use of Computers in Classrooms

Although several facilities using computers in classrooms exist, a generalized deployment is not yet accomplished. Several challenges regarding this issue can be identified, with several interrelations, of which the most significant are the following ones:

Cost

Each classroom implementation requires a number of computers proportional to the number of students per class. Generally it is a one-to-one relation, but sharing one computer by two or sometimes more students is a common solution, seeking to keep costs affordable. Each computer can be equipped with a tablet for freehand input, in the best case a tablet integrated into the video display. An electronic white-board completes the infrastructure, as well as a network connection at least within the classroom and to the server. Since the costs of equipment tend to become more affordable as time passes, it is reasonable to consider this a temporary issue that will be solved in the next few years, specially considering the major worldwide efforts underway to provide access to technology in schools and universities.

Courseware

A framework is useful if it can be deployed widely, in order to justify the fixed costs inherent to the installation and maintenance of such settings. One of the main aspects that influences usage in this particular case is availability of courseware that takes advantage of the system's features. This aspect has been receiving great attention during the last years (see e.g. Sparks et al., 1999; Retalis et al., 1997; Ochoa Delorenzi, 2002), so improvements in this area can be expected and have already been seen. Most notably, a vast amount of content has been created using standards such as SCORM and LOM (IEEE, 2002), that can be used in several settings.

Appropriation

It is essential, for the proposed activities to be carried out in the best possible way, that teachers get involved and actively participate using the system and learning how to best achieve the desired results. In most cases, the burden of having to learn how to interact with the system has to be overcome, and a period of adaptation seems unavoidable.

On the other hand, the system must also be designed to be as simple as possible to use, minimizing the burden on the teachers to learn features that they will hardly ever use or need. If a simpler classroom helps to increase the interest in teachers to use it, the simplicity should be evaluated as a positive feature. If the usage requires long preparations before users rip benefits or even start using it, the incentives to adopt this technology will decline considerably. Making the system easy to use taking advantage of existing skills and methodologies is an important part in allowing a direct use without requiring the acquisition of new abilities.

Using top quality courseware designed specifically to make the best of the classroom environment, including proposed activities and a great amount of basic tools that can be used without much preparation certainly helps to mitigate the entry barrier. But still, it is necessary to have the teacher's attention and interest to use the system throughly, and work together to achieve the ultimate goal: make the best out of the educational process.

1.1.3 Facing the Challenges

As already seen, most of the challenges regarding usage of technology inside classrooms in the mainstream are not technical. Rather, it is necessary to know how to apply these technologies in efficient ways to improve the value obtained from the educational systems.

These problems will not be solved with new systems that integrate better technology, however beneficial they may prove. What is needed is to use the technology in the best possible way, by means of applying existing experiences in several remote locations. In this regard, a plan of action or set of *Best Practices* can provide just the needed link to make a meaningful difference in the usage of these technologies.

Best practices apply to several aspects, above all to the development of materials (Courseware) to be used in the sessions, as well as to the sessions themselves. In the first issue, the best practices may suggest what information to include into the metadata, how to structure the sessions and what additional information to give to the teachers that will use the material, and a long list of other possibilities. The second issue implies giving advice to the teachers so they can use the system most efficiently, including recommendations on how to make best use of the available courseware.

Supporting the Teacher's Task

It has been noted that one of the main responsible characters for the classroom interactions is the teacher (Cortez et al., 2005), her support or lack thereof will have a significant impact on the learning outcome. This is the reason why helping the teachers do their work is the primary goal in Computer-integrated Classrooms (CiC) (Baloian et al., 2002a; Hardings et al., 2003), in which the face-to-face and computer-supported aspects of *Collaborative Learning* are combined.

While the role a teacher plays in a classroom is changing over time (see 1.1.1 on page 5), and the CiC is no exception, the importance of her role is as important as ever. Focusing on the needs a teacher has for comparing the results to other experiences and using the best of them as guidelines will have an important impact on the learning experiences. Therefore, this work will address the definition, validation and application of Best Practices in the context of computer usage inside face to face classrooms.

Collaborative Learning Flow Patterns (CLFPs) (Hernández-Leo, 2005) have been proposed as a formal way to model these experiences, as a way of collecting Best Practices in collaborative learning. CLFPs represent broadly accepted techniques that are used by collaborative learning practitioners. According to Hernández-Leo (2005), a learning session can be structured like a script as a flow of activities that may be collaborative of not. This can be called a *Learning Flow* or *learnflow* similar to the *workflow* in Computer-Supported Cooperative Work (CSCW). Thus, a situation can be understood as the context, including the current state of the session in the sequence defined by the *Learning Flow*, as well as the readiness or convenience to move forward to the next activity. The *Learning Flow* can even be dynamically adjusted by deciding the next activity based on this information.

1.2 Hypothesis

In this thesis we are going to enable a systematic way to define, validate and apply best practices in a computer-integrated classroom scenario. To achieve this, the information contained within the data of these settings needs to be accessed. We will assume that the particular environment is a system that provides access to the data.

The processes related to defining, validating and applying best practices are long-term activities, with the objective of creating best practices as a general form of a *Learning Design*. According to (Hernández-Leo, 2005), a *Learning Design* is a description of a method enabling learners to attain particular objectives by performing learning activities in a certain order in the context of a learning environment. In the process of defining the *Learning Design*, the task of accessing the information in the system has to be performed repeatedly. Having a description of the best practices that are being considered, a system that helps to obtain the needed information in an objective, repeatable and efficient way can be developed. With this information it should be possible to answer the questions defined in 3.3.4. The following hypothesis is proposed:

Hypothesis: It is possible to identify situations in a classroom, based on the information available within the system. These situations can be identified automatically by using a tool and that reduces the effort required for the identification process.

If it is possible to identify situations based on some objective criteria regarding the information available in the environment, this will help in a number of ways. First, the criteria can be shared among several persons, creating a potential for collaborating closely during the processes of definition, validation and application of best practices. Since a best practice is supposed to be applied in many different situations, being able to compare the experiences should be a fundamental requirement.

Second, best practices can be validated using objective indicators or metrics. These indicators can be calibrated using a vast amount of successful and unsuccessful experiences, getting closer to a mature set of recommendations by each new validation. In case the validation results in an unexpected result, it can be checked if the best practices were well-defined, correctly applied and if the objective indicators effectively represent the desired information.

Third, it will be possible to aid teachers during their classroom sessions by providing the information needed. This information will be interpreted by the teacher to verify the accomplishment of expected situations, note differing aspects and possibly take actions based on them.

1.3 Global Objectives

The global objective of this thesis is to develop the basis, including tools and concepts, for a methodology to define, validate and apply best practices in a Computer-integrated Classroom (CiC) environment.

1.4 Specific Objectives

The specific objectives can be enumerated as:

1. Developing a working Computer-integrated Classroom (CiC) environment that can be applied in several domain-specific educational scenarios, not being specific to any particular education level.
2. Developing tools embedded in said CiC environment that allow a teacher or tutor to define, validate and apply best practices in that scenario in the process of steering towards an effective learning design.
3. Performing a series of experiments in order to verify if the hypothesis expressed above do uphold in a real setting.
4. Evaluating the results of the experiments, proposing other applications of the methodologies used to define, validate and apply best practices.

1.5 Contributions

Several challenges regarding widespread adoption of technology inside classrooms have been analyzed. Several of these challenges are not of technical nature related to the equipment, but rather have important social components. Mature technology that proves helpful in several environments is available, and what is missing is the necessary knowledge on how to apply this technology in day to day situations.

Generally, experience regarding the usage of technologies or the application of processes eventually gets codified into *Best Practices*. This is a good way to transfer experiences from one place to another, and the best practices can also be further improved in a sequential innovation. However, no systematic approach exists in education that allows to express the experience in a way that can be taken advantage of in other settings. The work proposed here aims at providing a methodology that enables the description the best practices in a measurable way, making use of the available technology in several classroom settings. This description can then be validated in other settings, used as a base in order to improve the experiences, and shared among peers in order to apply the collaboration not only inside the classrooms, but among teachers as well.

Chapter 2

Use of Computers in Classrooms

In this chapter we review necessary concepts related to the work of the thesis, starting from general to specific topics. CSCW and Groupware (2.1) is described first, followed by Computer Supported Teaching and Learning (2.2) and Distant Learning (2.3). In Computer Supported Collaborative Learning (CSCL) (2.4) we refer to the field that supports the collaborative learning methodology, including Face to Face CSCL that is the focus in this thesis. The concepts of Context and Awareness are highlighted next (2.5), and then we introduce some systems that make use of technology in a face to face classroom scenario (2.6). The general concept of a CiC is explained (2.7) and finally we propose a system that will be built and used specifically for this thesis: CiCv2 (5).

2.1 CSCW and Groupware

Applications have been developed for solitary computer users for a long time. Even when designing multi-user systems, the most important aspect was to avoid users disturbing each other, creating isolated environments for each of them. This way it was possible to achieve transparency, eliminating any difference between sharing the system and using it alone. As an evolution of "office automation" taking place circa 1984, the aspects of information sharing and group coordination began to form part of some applications that are grouped under the terms *CSCW* (Computer Supported Cooperative Work) (Bannon and Schmidt, 1991) and *Groupware* (Grudin, 1994). The term CSCW stands for the research area while Groupware has been defined as *computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment* (Ellis et al., 1991). Within CSCW, group learning has become a niche in itself, with a rather independent community around research in Computer-Supported Collaborative Learning (CSCL).

Working with groups as well as computer systems and applications implies to focus attention not only on the technical issues involved in the applications, but also on social, motivational and political aspects, making the research area necessarily multidisciplinary (Malone and Crowston, 1994). This is particularly true in CSCL because of additional educational issues. In such an environment it is sometimes difficult to reach agreement on apparently basic premises, but it is also a rich environment to foster new concepts.

Within groupware, one of the most important and still not quite solved challenge has been to provide multi-user interfaces that generalize the graphical user interfaces that are commonplace on today's desktops. According to Grudin (1990a), multi-user interfaces[1] form the fifth stage in the evolution of interface research and design. The concepts of *context* and *awareness* (Dourish and Bellotti, 1992; Gutwin et al., 1996; Borges et al., 2004) and how to achieve them is part of the ongoing research in both HCI and CSCW communities (see 2.5).

2.2 Computer-supported Teaching and Learning

Computer support can be applied to the whole range of learning activities. These activities involve a lot more than just the sessions in which students and teachers interact. It is required to plan the sessions, execute them according to that plan and generally have some type of post-production in

[1]Although in Grudin (1990b) he criticizes the term *user interface* and states that instead *computer interface* would be more suitable.

which results are analyzed, grades are assigned, documentation is archived and maybe the next learning activity is planned, based on the results that can be obtained. Not all systems and methodologies fit exactly into this scheme, but all show at least some resemblance with the three phases that are presented here.

Several tools exist, which have been grouped under different names like *Courseware*, *Course Management Systems (CMS)* or *Learning Management Systems (LMS)*. Examples of commercial systems include WebCT (http://www.webct.com), eCollege (http://www.ecollege.com/) and Blackboard (http://www.blackboard.com/). These systems generally include communication facilities and content management systems, aimed to provide distance learning solutions.

2.2.1 Course Authoring

There are several tools available to support the learning-activity planning phase, mainly as authoring tools (Baloian et al., 1995) and specifications like IMS (Instructional Management System) content packaging (IMS Global Learning Consortium, 2003a), IMS metadata (IMS Global Learning Consortium, 2004), IEEE LOM (IEEE, 2002), PALO (Rodriguez Artacho et al., 1999) and the SCORM framework (Initiative, 2004). One of the main differentiating features between authoring tools is whether they consider content as well as activities or only one of them. If they do consider both, another distinguishing feature is if they present a rigid structure in which content is strongly tied to the activities.

It is possible to modularize the didactic components sufficiently as to let the actual planning of the activities happen in real-time, during the classroom session, in a smooth way and without relying entirely on the skills of the presenter. It is possible to let the presenter choose what material to use, what activities will be most suitable for the content and the group, as well as the order in which to present them. But at the same time, the systems that allow this type of adaptive behavior (Baloian et al., 2002b, 2001; Graf and Schnaider, 1998; Eklund et al., 1997; Murray, 1998; Macías and Castells, 2001) require a model which has to be defined previously. This model is typically based on links and references between didactic components and therefore requires considerable work before any classroom session can begin.

Other initiatives have tried to make the work involved in authoring of course material almost negligible, by recording what happens during the classroom sessions (Abowd, 1999). This approach relies on the normal preparation of the session by the teacher, as well as a post-processing of the session, in which some indexing work is done to allow later access to specific parts of lectures.

2.2.2 Course Presentation

The presentation of material within a computer-enabled classroom is an important part of the system, and dynamic management of the learning path is desirable. A number of projects have tried to provide adaptive courseware, such as IDEALS MTS (Graf and Schnaider, 1998) and Flex-eL (Lin et al., 2002), or adaptive hyper-textbooks like Interbook (Eklund et al., 1997) and Multibook (Steinacker et al., 1999).

In Flex-eL (Lin et al., 2002) the authors use workflow technologies to keep track of the student's learning progress, determining probable collaboration opportunities and defining a suitable learning path in a flexible curriculum. In order to accomplish these tasks, metadata has to be collected that allows further analysis.

Another possibility is to use agents that provide realtime feedback and predict or detect certain situations, providing intelligent support as in Mühlenbrock et al. (1997), merging the presentation and analysis phases and thus enabling a more powerful support tool for teachers.

2.2.3 Course Outcome Analysis

After the course sessions have taken place, it is possible for the teacher or other educators to review the outcomes. The focus as well as results of these analyses lies within a wide range of possibilities, as well as various research areas. It is possible to analyze group behavior during the activities, try to measure individual or group knowledge, analyze the outcome of specific pedagogical methodology or identify student's special needs.

2.3 Distance learning

At present, in distance learning scenarios, face-to-face teaching with a strong teacher-student relation is being replaced by the interaction among learners (peers) and learning materials. The interaction between learner and teacher, and among student peers is performed through various channels, trying also to include non-verbal communication (Guye-Vuillème et al., 1999). This type of communication is crucial for a good understanding (Bos et al., 2001; Buckingham Shum et al., 2001), since it provides a large amount of information that is not received otherwise. One of the main challenges in distance education is to mimic (or otherwise compensate for the loss of) the social context within a traditional classroom environment, so that the learners feel and act like insiders rather than outsiders of the course and get a feeling of community (Wegerif, 1998).

It is also argued that technology can enable improvements on the educational models (Pea, 1993) (tele-mentors, tele-apprentices) that would not be possible otherwise because of technical or economic constraints. Approaches to mimic the face-to-face scenario generally include audio and video channels to provide some support for nonverbal communication and awareness widgets such as telepointers, users lists and other widgets (see for example Wang and Chee (2001)) which help to create a feeling of participation within the group activities.

However, the most common distance learning environments are web-based (for a comparison, refer to University (1999) or Jackson (2000)). This enables a wide audience to participate, since the requirements are lower because web-browsers are available almost everywhere without further installations or other technical details. On the other hand, using a standard web-browser means that the possibilities of interaction are severely limited by the functionalities that the browsers allow.

2.4 Computer-Supported Collaborative Learning (CSCL)

Although information technologies (IT) can be used in a variety of situations inside a classroom, a good example of such usage is the discipline of Computer-Supported Collaborative Learning (CSCL) (Koschmann, 1996a; Lipponen, 2002). It is possible to exchange contents and work on shared workspaces, manage the learning materials, turn-taking and other relevant aspects of the interactions that would not be easily managed without the technology. In some sense, CSCL can be viewed as a paradigm shift within the evolution of Instruction Technology, among Computer Assisted Instruction (CAI), Intelligent Tutoring Systems (ITS) and Logo-as-Latin (Koschmann, 1996b, 2001).

Collaborative learning has been one of the main ways to use constructivist theory to improve learning. In a collaborative learning environment, better results are obtained (Johnson and Johnson, 1998) in terms of more learning, longer retention of the learned subjects, development of superior reasoning and critical thought (Cortez et al., 2005). This is an area that can benefit through computer support by creating innovative scenarios in which students can collaborate. The possibility of showing one part of a whole model to a student and other techniques can be used easily in a computer-enabled classroom to naturally force the students to interact in some way to solve a common problem. This way, a collaboration among peers can result and this is the base for a constructivist approach. According to Roschelle and Teasley (1995), *Collaboration is a coordinated, synchronous activity that is the result of a continued attempt to construct and maintain a shared conception of a problem*, which leads to the notion of a *Joint Problem Space* as an enlargement of the concept of *common ground* (Clark and Schaefer, 1989).

Distance learning is being the focus of several research initiatives in the CSCL arena. However, most of the learning activities are currently held in face-to-face sessions in traditional classrooms.

It is not clear whether distance learning environments will replace the current learning activities or if both methodologies will enhance each other. Whatever the case, classroom activities as well as distant learning methodologies can benefit from improvement in computer-based support, and a classroom setting allows combining the face-to-face interaction along the computer-based channel.

CSCL is basically a tool that intends to support Collaborative Learning. This translates to defining the focus of CSCL, according to Lipponen (2001), on how collaborative learning supported by technology can enhance peer interaction and work in groups, and how collaboration and technology facilitate sharing and distributing of knowledge and expertise among community members. The acronym CSCL has been in itself not without controversy, and several authors (Koschmann, 1996a; Lipponen, 2002) even have avoided to refer to the expansion of the acronym, allowing it to be interpreted as each reader desires and focus the discussion away from the name of the field. It is now accepted to refer to CSCL as *Computer Supported Collaborative Learning*, since the term "Cooperative Learning" has a special meaning in the field of education, but some authors have expressed their wishes to substitute the word *Collaborative* for *Collective*, *Cooperative* or *Coordinated* (Koschmann, 1994). However, most authors agree that the area has a great importance, and it has been argued by Koschmann (1996b, 2001) that CSCL can be viewed as a paradigm shift within the evolution of Instruction Technology, among Computer Assisted Instruction (CAI), Intelligent Tutoring Systems (ITS) and Logo-as-Latin.

While CAI reflected the ideas of behaviorism and instructional efficacy, tailored to specific learners with specific needs, the ITS applied methods of artificial intelligence to understand tutoring in complex domains. Both paradigms rely on the transmission model of instruction (Koschmann, 1996b), and were categorized together by Crook (1994). Papert stated that students could construct and discover new understanding when engaged in programming activities, and developed the Turtle Logo Microworld (Papert, 1980) to explore this area, that is now considered a third paradigm (Logo-as-Latin).

According to Roschelle and Teasley (1995), *Collaboration is a coordinated, synchronous activity that is the result of a continued attempt to construct and maintain a shared conception of a problem*, which leads to the notion of a *Joint Problem Space* as an enlargement of the concept of *common ground* (Clark and Schaefer, 1989). The memorization of information and gain of capabilities is not the only and probably neither the most important aspect of learning, as illustrated by Minsky in what he calls *Papert's Principle*: "Some of the most crucial steps in mental growth are based not simply on acquiring new skills, but on acquiring new administrative ways to use what one already knows." (Minsky, 1986). This is also stressed by Piaget as the "shock of our thought coming into conflict with others" (Piaget, 1969), which results in construction of new conceptual structures and understanding, either because the co-construction takes place in individual minds as an effect of the *socio-cognitive conflict*, or because of the increasing ability to take into account other people's perspective.

Lipponen notes that in the history of psychology, many writers have stressed the idea that collaboration is a basic form of human activity and essential for cultural development (Lipponen, 2002) (c.f. Bruner, 1996; Engeström, 1987; Hutchins, 1995; Mead, 1934; Tomasello, 1999; Vygotsky, 1962, 1978; Wundt, 1921), while Paavola et al. note that *knowledge is an aspect of participation in cultural practices rather than something that exists in a world of its own or in individual minds* (Paavola et al., 2002) (c.f. Brown et al., 1989; Lave, 1988; Lave and Wenger, 1991), so that the concept of learning only makes sense within a social context, a *community*. According to several authors in the CSCL community, *learning is a social process of collaborative knowledge building* (Brown and Campione, 1994; Lave, 1991; Pea, 1993; Scardamalia and Bereiter, 1996). The importance of knowledge construction has motivated the creation of tools to allow and enhance awareness of this process in computer-mediated interactions (Collazos et al., 2003).

It is important to note that neither collaboration nor computer support provide a learning experience on their own. To say otherwise would be equivalent to argue the same about "learning from being alone" (Dillenbourg, 1999) or "learning using books, paper and pencil". There has to be a didactic model or theory of learning that is the central aspect of a learning scenario (Koschmann, 1994; Stahl, 2000).

2.4.1 Face to Face CSCL

Face to Face collaboration not only involves the human nature of communication, but also the tasks that are involved (Fussell et al., 2002). Communication involves much more than speech, such as gestures, eye gaze, and other nonverbal cues. And all of these elements are present in a Face to Face scenario, without the need to either substitute them with other indicators or cope with the lack of them, as is the case in distant learning. The ability to perceive and interpret these forms of information and their relation to the group task is important for negotiation, agreement interruptions, and acknowledgement (or lack) of understanding. The different knowledge and backgrounds of group members requires a shared understanding that is facilitated by the face-to-face relations (O'Neill et al., 1999).

It is necessary to distinguish two categories of work that is performed when carrying out activities in the shared workspace of a face-to-face collaboration. The team work refers to the work of working together, whereas the task work is the part of the work that focuses on carrying out the task (Pinelle et al., 2003). Both the team and the task work have to be understood as interconnecting elements with the actors of the activity. Task Analysis provides a basis to investigate all aspects of the collaborative task work. It allows decomposing in basic building blocks the team and the task work, which can later be used in the design of a face–to-face computer supported collaborative learning activity.

The mechanics of collaboration cover two general types of activities: communication and coordination. Communication is broken into two categories: explicit communication and information gathering, and coordination is broken into two categories: shared access and transfer. There are seven major activities that are covered by the mechanics of collaboration (Pinelle et al., 2003, 2004; Johnson and Hyde, 2003)

- **Explicit communication** relates to group members explicitly providing each other with information by verbal, written, and gesture means.
- **Implicit communication** concerns people's ability to pick up information implicitly pro-

duced by others in the undertaking of activities from their individual actions and their manipulation of artifacts.

- **Coordination of action**, considers that people organize their actions in a shared workspace so that they do not conflict with others.
- **Planning activities** carried out in a shared work space are task division among group members, reserving areas of the workspace for future use, and plotting courses of action by simulating them in the workspace.
- **Monitoring**, involves people keeping track of other people with respect to where they are and what they are doing; it can be associated with giving assistance.
- **Help**. Help may be opportunistic and informal between group members, when the situation makes it possible for one person to help another, or alternatively it may be explicitly requested.
- **Protection**. Group members have to be able to protect their individual work from internal (other group members) and external interference.

The collaborative activity to be undertaken and the skills necessary to engage in the activity are also important features. There are two sets of skills and expertise needed (Johnson and Hyde, 2003):

- To communicate and collaborate with other group members
- To harness the necessary individual and group skills to accomplish the task(s).

Group problem-solving, and planning the substance and structure of a group activity, requires successful negotiation, inference, and comprehension of expectations between group members. Also effectiveness is affected by the participant interests, attitude, motivation, and the role(s) that they must play in the group activity.

An interesting example that integrates personal and shared spaces using augmented reality to support face-to-face collaboration is Caretta (Sugimoto et al., 2004). A multiple-input sensing board with augmented-reality technologies is used for the shared space, while PDAs for each user's personal space. Based on the arrangement of objects in the multiple-input sensing board, the system executes computer simulations and users visualizes them in their PDAs. Users discuss and negotiate with each other in the shared space by manipulating physical objects, while they individually examine their tasks in their own personal spaces. The system not only increases the level of visibility of the users' actions, but also creates an immersive environment for collaboration.

2.5 Context and Awareness

As mentioned before, awareness is a critical aspect to successful collaboration and is commonly emphasized in CSCW and CSCL applications. It has been defined by Dourish and Bellotti (1992) as "an *understanding of the activities of others*, which provides a *context for your own activity*" (emphasis in original). This is particularly important in collaborative creative work like writing (Beck, 1994; Galegher and Kraut, 1990), and applies not only to synchronous activity but semi-synchronous and asynchronous work as well (Dourish and Bellotti, 1992; Borges and Pino, 1999; David and Borges, 2001).

The usefulness of the right information at the right time is clear and is not questioned. However, adding all available information does not help and may cause indifference, improper interruptions and diminish attention due to information overload (Maes, 1994). Some alternatives have been proposed to select the right information to be delivered to the user. One alternative is to summarize or aggregate the gathered information to deliver less detail to the user, thus changing the granularity of the information that has to be processed directly by the user (Borges and Pino, 1999). Another possibility that has been proposed is to filter information based on a dynamic and constantly updated profile, so that the user receives only the information that is relevant (David and Borges, 2001).

Context has no unique definition, among other reasons because the concept is perceived differently according to the domain (Brézillon et al., 2004). We will use the definition of context proposed by Brézillon et al. (2004): "*whatever does not intervene explicitly in a problem solving but constrains it*".

Context is relevant in this thesis in two ways: the first is any context that affects the information available in a system. We will use information and when the result depends on information that is indirectly available, the result will depend on the context. The second issue for context is regarding elements that do not reflect on the information inside the system but are important in the pedagogical outcome. These elements are mostly in the scope of Social – familiarity with the system, cultural heterogeneity, readiness to use the application, demographics, etc. – and Physical Context – space, physical conditions, privacy, relative physical position of students – as defined in Alarcón et al. (2005). These aspects of context have to be taken into account by the teacher, because the system to be used in this thesis does not provide any means to identify them.

2.6 Computer-enabled classrooms

In this section we present a selection of existing computer-enabled classrooms. They all use Information Technology (IT) to accomplish differing goals. We end with a discussion on the similarities and differences of these systems.

2.6.1 Colab

Colab (Stefik et al., 1987) is an experimental meeting room for groups of two to six people, each one working on a personal computer and a central, large, touch-sensitive screen and stand-up keyboard. The meetings are divided in several phases (e.g. brainstorming, organizing and evaluation), and the available tools have different behaviors depending on the meeting phase. This is done as an effort to emphasize particular meeting processes and styles of behavior, but without forcing their adoption.

- cognoter: collaborative tool used to create outlines for talks and documents. The meeting phases are: brainstorming, organizing and evaluation.
- boardnoter: meeting tool which provides an area for freestyle sketching, similar to a chalkboard.
- argnoter: a tool supporting arguing, designed for meetings in which participants have done previous work, which generally means that disagreements and disputes have to be settled. The meeting phases are: proposing, arguing and evaluation.

2.6.2 eClass (formerly Classroom2000)

Approaches like eClass, (formerly Classroom 2000) aim to capture a lecture integrating pen-based technology, audio services, etc. for use in short-term (e.g. lesson playback, exam preparation) and long-term situations (e.g. comparing courses taught in several periods, evolution over time, searching for relations among various courses) (Abowd et al., 1996; Abowd, 1999; Abowd et al., 2000). The lectures are captured trying to minimize any additional preparation and using processes that allow the systematic post-processing and archiving of a high number of lectures during a longer time frame. The result is a library containing a considerable amount of lectures that can be analyzed and compared themselves as well as their usage.

Figure 2.1 shows the eClass environment, indicating the following elements: the information presented on the electronic whiteboard (A), captured by a video camera in the back of the room (B) and by microphones embedded in the ceiling (C), as well as web browsing projected (D). Information from previous slides is presented simultaneously in other projected screen (E).

This approach is similar to other projects such as the Digital Lecture Hall (DLH) (Mühlhäuser and Trompler, 2002). In both cases, both teacher and students have to engage in almost no behavioral change at all, allowing for a smooth transition to the use of technology in the classroom. DLH concentrates on supporting large numbers of students, on integrating a spectrum of learner-owned devices, and on minimal distraction, while eClass has a focus on allowing to capture a large

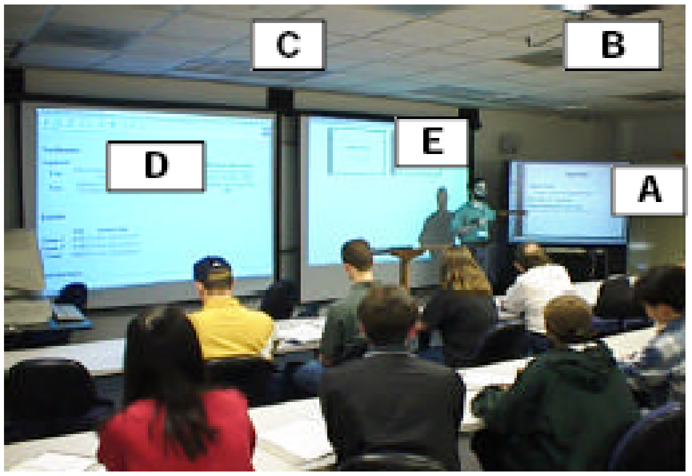


Figure 2.1: An overview of eClass.

number of classroom sessions.

2.6.3 CoVis (http://www.covis.nwu.edu/info/)

The Learning Through Collaborative Visualization (CoVis) project (Pea, 1993; Edelson et al., 1996) has as its goal to improve science education in middle and high schools, providing students with a range of collaboration and communication tools.

It is argued that most educational settings tend to focus on "learning before doing", whereas a model of "learning by doing" could have several benefits. New technologies can be used to facilitate the return of successful learning models that existed prior to formal schooling, such as apprenticeship and long-term mentoring.

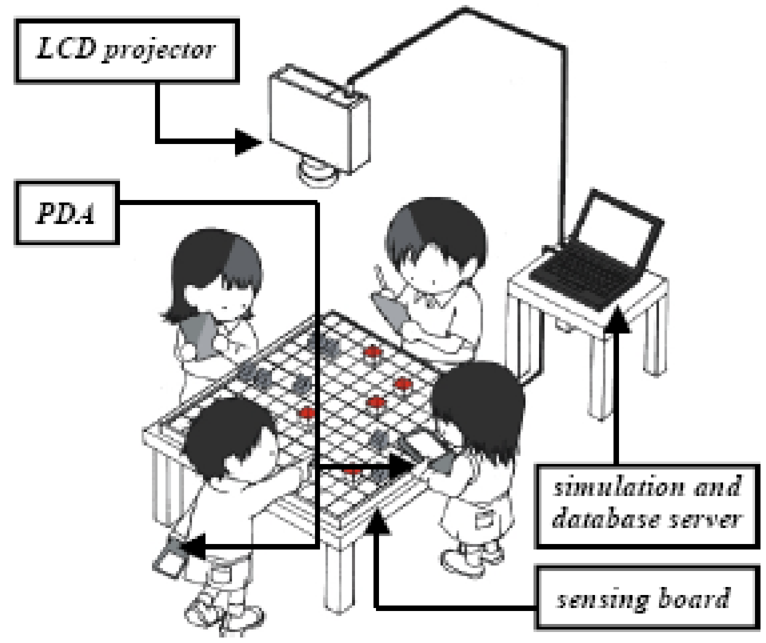


Figure 2.2: An overview of Caretta.

2.6.4 Caretta

Caretta (Sugimoto et al., 2004) integrates personal and shared spaces to support face-to-face collaboration. PDAs and a multiple-input sensing board (see figure 2.2) provide personal and shared spaces, respectively. Users of Caretta can discuss and negotiate with each other in the shared space by manipulating physical objects, while they individually examine their ideas in their own personal spaces. Caretta allows users to participate in group activities interchangeably and seamlessly using both these spaces.

2.6.5 Analysis and Discussion

We have seen several different approaches to introduce Information Technology (IT) into the classrooms. In the case of Colab (2.6.1), computer-mediated activities guide the development in a rather strict way, with predefined phases and concrete objectives or outcomes. The target group for Colab is small and not necessarily directed to teaching and learning, so its use as a classroom is not practical unless the class is divided into small groups. However, it is interesting to compare the nature of Colab, defining how the sessions need to progress, to the rather free environment of eClass (2.6.2). In the latter, teacher and students need not even be aware that the sessions are being captured for later use, including audio and video streams as well as auxiliary material. The latter is seen as being a less intrusive application of the technology to a classroom scenario, and thus a preferred way as the move towards computer support in learning should be smooth rather than progress in big leaps (Mühlhäuser and Trompler, 2002).

CoVis (2.6.3) is a tool specific to science in high-schools, being used as one of several possible ways to present a given subject as a (virtual) hands-on experience, in a similar way as Caretta (2.6.4) does. The latter is even more specific to a certain domain, but has the feature of presenting a form of "augmented reality" in which the students manipulate physical-world objects in order to manipulate the data in the virtual model.

In this thesis we are interested in using technology as a support tool throughout all of the classroom activities. To achieve this, we intend to not impose a specific learning methodology, allow dealing with any domain or subject matter and support the whole range of activities rather than provide isolated use of technology in some areas. As of this time, audio and video processing will be left out, but it could be considered as a rich information source in the future.

In the next sections we present the concept of a Computer-integrated Classroom (CiC) that intends to achieve the objective mentioned above. Next, we will describe the specific CiC environment that was implemented for the work in this thesis.

2.7 Computer-integrated Classroom (CiC)

The idea of a Computer-integrated Classroom was first introduced in the Cosoft project (Hoppe et al., 1993), and has since been adapted to an early learning scenario in the NIMIS project (Lingnau and Hoppe, 2002; Tewissen et al., 2001; Hoppe et al., 2000). The main objectives in a CiC are:

- offer a more interactive learning experience than a traditional classroom,in which the one-way flow of information and mostly the one-way initiative have been criticized.
- offer multimedia technologies during a lecture without introducing interruptions that are unrelated to the session.
- improve student-student and student-teacher collaboration within the group.
- offer an integrated learning environment with access to hypermedial databases, communication and simulations.

In a Computer-integrated Classroom (CiC) framework, the teacher has access to an electronic blackboard on which the learning material may be displayed and manipulated during the session (Elrod et al., 1992; Hoppe et al., 1999; Streitz et al., 1999). The students have access to the material using notebooks or PDAs, probably through a wireless LAN. The electronic blackboard and the student's applications are able to interact in order to share contents and work over shared workplaces. A CiC also includes systems which help their users to manage the learning materials, turn-taking and other relevant aspects of the interactions. This helps to avoid disruptive activities like typing filenames or navigating to find information within the sessions. It would be helpful to have available a "classroom management system" which allows not only to manage the learning materials, but also analyze and learn from classroom sessions. The teacher would be able to recognize when some student or group needs attention without having to interrupt their work. The activities can be divided in:

- individual exercises

- joint exercises
- peer to peer help

One of the benefits of having digital media inside the classrooms is both, teachers and students, can take advantage of the ease of manipulation (reproduction, re-use, editing, versioning, etc.) that are inherent part of these media. Teachers can prepare relatively complex material in advance, reuse them in other courses and contexts with or without modification, and give copies to students for them to edit and learn through experimentation. The digital media, in the way it is used in the CiC, is *alive* in the sense that it encourages the users to modify and use it in creative ways, in contrast to the use of digital content as a one-way information carrier.

One of the interesting aspects of these environments is that the role of technology is not the central point but rather a tool that moves into the background (c.f. Hoppe, 2002). This enables the pedagogical (and one would argue, the really important) methods and activities to develop independently of the technological rush. The use of technology in ways that do not resemble computers (Ishii and Ullmer, 1997) is another step in the same direction.

2.7.1 Cosoft

The Cosoft project (Hoppe et al., 1993; Baloian et al., 1995; Baloian, 1997) tries to avoid the use of Intelligent Tutoring Systems (ITS) and focus instead on teacher-centered instruction, which is the dominating classroom situation nowadays. The use of socratic dialogues, prepared exercises and reflection is encouraged in order to avoid the unidirectional effects generally associated to this method. The project proposes to use technical support mechanisms which may minimize the main disadvantages of traditional (non-computerized) classroom interaction, such as:

- (teacher) presentation
- individual (student) presentation

- group (student) presentation
- open discussion
- guided discussion

2.7.2 NIMIS

The Networked Interactive Media in Schools (NIMIS) (Hoppe et al., 2000; Tewissen et al., 2001; Lingnau and Hoppe, 2002) project started in 1998 and has as its goal to support classroom information management for children being 4 to 8 years old. Its central application is The Talking Typewriter (T^3) (Tewissen et al., 2000b), which allows children to learn reading and writing according to the well known method "Lesen durch Schreiben" (reading through writing) (Reichen, 1991), used in Switzerland and Germany. The computer uses a speech synthesizer to speak out what the child has written, a task that has to be done by the teacher in other settings.

In the NIMIS setting, the computer is not in the foreground, as the child only interacts with sensitive LC displays using direct manipulation (Hutchins et al., 1986) of visual objects. The idea behind "the disappearing computer" (FET, 2001; Norman, 1998) is to center the child's focus of attention on the real work, avoiding distractions due to the use of computers. It is also tightly bound to the *ubiquitous computing* (Weiser, 1991) or *pervasive computing* terms.

2.7.3 SEED

The SEED ("seeding cultural change in the school system through the generation of communities engaged in integrated educational and technological innovation") research project (Seed, 2005), supported by the European Union, lasted from 2001 until 2004. An important aim of this project was the generation of integrated communities, consisting of teachers, researchers, and software developers, combining their educational and technological expertise in designing, developing and implementing innovative activities at small scale.

During this project, two Plug-Ins used in the development of this thesis were developed: the System Dynamics implementation (Bollen et al., 2002) and a Stochastics environment (Lingnau et al., 2003). The application of these plug-ins elaborates on the concept of "seamless media integration in the classroom", and also exemplifies the approach of computer based tools being integrated parts of learning environments as presented in Hoppe (2002).

2.7.4 COLDEX

COLDEX ("Collaborative Learning and Distributed Experimentation") is a research project funded by the European Union from 2001 to 2005 (Coldex, 2005). Its aim is to use new IT approaches and computational tools to foster scientific experimentation, modelling and simulation in distributed collaborative settings in an inter-cultural (European-Latin American) community of learners. In this context, a focus of the COLDEX project is the study of visual and other perceptual phenomena, including astronomical and seismic measurements, from both a scientific and a subjective experiential perspective. In this project, the connection of the system to real physical devices (including data transfer or exchange) was an essential factor, presenting "mixed reality" technologies which allow for a smooth transition between the physical and the digital worlds.

2.8 Domain-independent CiC Implementation

The existing CiC implementations have had specific audiences and objectives in mind. However, in this thesis the use of CiC is intended to cover a wider variety of scenarios, requiring to adapt to more general needs, for which the existing implementations are less suitable. Therefore, it is necessary to create a flexible CiC implementation that can be used in various educational domains and classroom situations. This implementation is called CiCv2 and is described in detail in Chapter 5. It has been developed by the author of this thesis, using an early version of the FreeStyler tool (described in 5.5) that has also been widely adapted for the purpose of this work.

In order to prove the hypothesis stated in 1.2, we need to use a general-purpose environment that can be tested in different scenarios and pedagogical domains. In the existing CiC environments there is no possibility of incorporating new visual elements, rules nor functionalities without modifying the application substantially. As in the CoolModes environment (Pinkwart, 2003), the CiCv2 environment will gain flexibility with the introduction of *reference frames* which provide visual languages, rules that govern them and functionalities related to these languages.

Also, the central goal of developing a specific environment is to enable the creation of a tool that can access all relevant information generated and used in the classroom sessions using a single interface. Since no such interface exists in other implementations of CiC nor in any of the reviewed Computer enabled Classrooms, the need justifies the creation of a new environment. By providing a single interface to access the data, the teacher can have access to all the needed information as we will see in the next chapters.

Another important aspect to consider is the extra-classroom activities that take place and can provide valuable information to the process. The work carried out by students at home is asynchronous, and can be seen as a black box from the collaboration point of view, since the environment is not under a centralized control and it is unknown whether the student is for example interacting face-to-face with peers or is working alone. Also, inside the CiC, there is no intention to force or disallow any particular interaction, since that is a pedagogical decision and should be freely made by the teacher. There have been some experiences (Goodman et al., 2001) trying to provide both, synchronous and asynchronous collaboration using methods for "replaying" sessions and interactions. However, the users have an important responsibility to provide useful information when working asynchronously in these scenarios.

Other unavailable information is how long the student really has been working on the document, or what activities take place in parallel to the visible work (from the system's standpoint). By allowing the use of the documents and material available in the CiCv2 environment from the outside, it is possible to greatly expand the scope of the pedagogical methodologies used, without tying all of the activities to a physical classroom.

Chapter 3

Best Practices

Many technical problems regarding usage of courseware in classrooms have been solved, most of them have also been tested in real settings. However, for a widespread use of these kinds of settings, it is necessary to have a solid background on the usage of these systems for achieving the best results. This background can be expressed as a series of *Best Practices* that can be used in a given context.

Best Practices, along with Benchmarking, are concepts used widely in areas dealing with strategic management and business processes (Crosby, 1980; Deming, 1982; Juran and Gryna, 1988), but has also been embraced in other contexts such as medicine (Mold, 2003), government (Hayman, 2002), public affairs (Altshuler, 1992), mental health (Martin et al., 2004) and others. In the case of Object-Oriented Programming, best practices have been described successfully in the form of *design patterns* (Gamma et al., 1994; Beck et al., 1996). Best Practices are the result of interpretation of a wide number of experiences, according to Ruchti (2002) usually developed through "anecdotal and descriptive research". To overcome this reality, it is necessary to develop some aids that allow to share the wide number of experiences that are the basis of best practices and avoid differing interpretations of descriptions and recommendations.

Bretschneider et al. (2005) define three important characteristics associated with a best practice in the context of public administration:

1. a comparative process
2. an action
3. a linkage between the action and some outcome or goal

These characteristics allow to determine which practices are better than others, eventually finding the best practices by comparison. Moreover, best practices should really be considered best *currently known* practices, assuming that the process to define them is of sequential, cumulative and adaptational nature. It is necessary to start comparing experiences at some point, and move from that on to the next best known practice. This is only possible when being able to compare both the actions and the outcomes or goals that motivate them.

The concept of "best practices research" or "best practice research" has been used to refer to an organized attempt to learn from the experience of others. It aims at identifying the best possible set of solutions for a given problem (Dawes et al., 1997). It has been called the latest version of the inductive practice-to-principle approach (Overman, 1994). A similar definition, oriented towards best practices in a clinical (medicine) environment is given by Mold (2003) as *a systematic process used to identify, describe, combine, and disseminate effective and efficient strategies developed and refined by practicing professionals*.

Myers et al. (2004) identify two major approaches to conduct best practices research in the context of public affairs: the *quantitative / microeconomic* approach and the *qualitative / case study* approach. The quantitative approach is said to be harder, due to the difficulty of finding comparable data for a set of exemplars or candidate practices. This is less of a problem in the educational scenario being proposed, and moreover it should be used together with a qualitative approach as a means of verifying proposed best practices.

3.1 Desired Characteristics of Best Practices

According to Bendixsen (2003), Best Practices is about accumulating and applying knowledge of what is working and not working in different situations and contexts. In this way, it is possible to take the knowledge and increase the chances of improving the effectiveness in similar situations and contexts. Bendixsen goes on to consider that the practices can be considered *best*, *good*, *sustainable*, *successful* or *promising* in an almost equivalent way. Since in a process it is not possible to definitively conclude that no better practices than the ones being currently considered do exist, it should be assumed that the practices being considered are generally currently known best practices, better than many comparable alternatives, but not necessarily better than absolutely any other possibility. Having this in mind, when looking for best – or *good* – practices some of the following characteristics should be kept in mind according to Bendixsen:

- be innovative
- make a difference (impact)
- have a sustainable effect
- have potential for replication

3.1.1 Innovation

When looking for better ways to solve a problem or to perform a task, it makes sense to look at new and creative solutions to common problems. In some cases, innovation is in itself a contribution, for instance when trying to avoid boredom or simply to make a product or process different from everything else.

Considering aspects like improving some indicator, optimizing a set of variables or speeding up a process, it also makes sense to search for new ways of doing things. Keeping looking at already known solutions that have already been optimized can have an impact, but a radically different approach can lead to improved results. But in this case, innovation is not considered to be a benefit by itself, rather it is seen as a possibility to improve the real objectives.

3.1.2 Impact

When a particular aspect or characteristic of candidate best practices has a high influence on the result, that aspect or characteristic will get a high level of importance. When that aspect conflicts with another that has not as high a influence on the result, the latter would probably tend to be ignored. This applies to both kinds of aspects: the ones that, when followed, improve the final result, as well as the aspects that indicate something should be avoided in order to gain the desired results.

3.1.3 Sustainable Effect

Some actions or recommendations can be applied more than once, or even permanently over time. These actions or recommendations have a sustainable effect when the continued application has a positive effect, or at least does not diminish its effectiveness in time.

In some cases, actions can be applied only once, for example ordering some elements in a particular order. Applying that same action again will have no effect whatsoever, because the elements were already ordered. However, the same action can have a sustainable effect if the elements tend to change their order in the normal course of their usage. So, the sustainable effect characteristic is not only tied to a particular action, but to the context in which the action is being applied.

3.1.4 Replication

Of course, the point of defining best practices is to replicate them in similar scenarios, or at a different time frame. So it makes no sense to define best practices that can only be applied once. But the replication, as already mentioned, can be local (applying the practices at a different time) or remote (applying the practices at the same or different time at a distant location).

Best practices can and generally will depend in some way on the context in which they are applied. This context has to be clear in order to predict if a set of best practices will be successfully applicable in some other context or not, before running into context-dependent issues.

3.2 Best Practices in Education

When applying the characteristics, defined in (Bendixsen, 2003) and mentioned above, to an educational environment, some characteristics should be given more importance than others, depending on the particular situation. For example, being innovative is generally not an important issue for a best practice, as innovation is not seen as a goal itself in the learning process. Rather, it might be useful to try out innovative ways to teach and learn, in the hope that the result of such innovations has a positive impact on the learning process. Therefore, the impact should receive most importance, since the goal in any educational process is to achieve a positive and tangible impact on the learning outcome.

A sustainable effect can be important when dealing with general or coarse-grained best practices (see 3.3.3), but should be less an issue when considering very specific (fine-grained) best practices. In the former case, a principle or recommendation can be applied several times in the course of a learning process that may last several years like school or university degrees. In the latter case, it is likely that the students will be subject to the practice only once, during them learning the particular subject, so a sustainable effect does not make as much sense. However, when considering coarse grained, more general, practices, it may be of utmost importance that the effect does not diminish when applying the same practice over and over. For example, when a child is told to perform exactly the same activities, like a game, and the only changes are the contents to learn through the game, the positive effect will probably not be sustainable in the long run. The game might be interesting in the first few instances, but later the student will become used to it and the game will loose the appeal that made it successful in the first place.

Regarding the characteristics expressed by (Bretschneider et al., 2005), they can be applied directly in an educational setting. Comparing outcomes or achievement of goals is a generally well understood, even if far from perfect, activity within educational activities. Although assessment, specially in CSCL, is an active research area, it is possible to rely on widely known and applied methods to evaluate learners progress. However, comparison of actions is not easy, since the same criteria need to be shared among the people evaluating whether a defined practice is being followed or not. This activity would be much easier when extracting some comparable characteristics out of the interactions that take place in the classroom, and that way being able to determine whether a certain practice is being followed.

3.2.1 Existing Best Practices in Education

Within educational settings, it is possible to find a lot of recommendations aiming towards the improvement of learning activities. In general, they are of a very high abstraction level, and they generally are not comparable among different criteria. This means that it is difficult to determine whether a practice is being used as intended. For example, Chickering and Gamson state in (Chickering and Gamson, 1987) that good practice in undergraduate education should:

1. encourage contact between students and faculty
2. develop reciprocity and cooperation among students
3. encourage active learning
4. give prompt feedback

5. emphasize time on task
6. communicate high expectations
7. respect diverse talents and ways of learning.

It is difficult to try and compare how far these aspects have been implemented in two classrooms that are thousands of kilometers apart and do not at least share a teacher, even at a different time frame. All of the recommendations sound pretty much right and make a lot of sense, and are effectively the result of years of experiences, including success stories and failures. But is it possible to know if high enough expectations are being communicated, whether the students engage actively enough in the learning process, and so with the rest of the recommendations? It is hard to measure the characteristics of the teaching and learning processes in order to compare them to others.

The same problem exists with other sources of best practices (c.f. Zemelman et al., 2005; Walker Tileston, 2005; Marzano et al., 2001), that are of a high level of abstraction and it is up to the teacher to manage to translate those recommendations into everyday actions. A teacher would love to have some quick way to determine what specific action she could take to improve the classroom interaction in a particular moment. That is, how to map the experience coded in the recommendations into something that is needed on behalf of her.

Odon et al. (2005) argue that the impetus for the current evidence-based movement in education reflects *a concern that effective educational practices, as proven by research, are not being used in schools*. That is, currently several effective practices exist, many are even evidence-based, but the real issue is how to take these practices and translate them into day to day concrete actions. Moreover, it is necessary to verify that the practices are being applied as intended.

3.2.2 Modeling and Specification of Best Practices

The formalization of representative and broadly accepted structuring techniques in collaborative learning is proposed in Hernández-Leo (2005), in such a way that CSCL systems could reuse, particularize and customize these best practices according to the requirements of a concrete learning situation. The formalization proposed is useful as a formal description of Collaborative Learning Flow Patterns (CLFPs) and can be enhanced by providing additional information to the teacher regarding the current context of a session. The context can be represented as the position or state in the learning flow, including the relevant information used to determine the convenience of moving on to the next state, or even deciding which of several possible next states is the best choice.

The description of CLFPs is based on natural language, as a *way of collecting "best practices" in collaborative learning* (Hernández-Leo, 2005). The IMS Learning Design specification (IMS Global Learning Consortium, 2003b) can be used to formalize CLFPs into a language that can be shared and processed by applications. A Learning Design is a description of a method enabling learners to attain particular objectives by performing learning activities in a certain order in the context of a learning environment. Although the primary focus of the IMS Learning Design specification is the creation of online content for e-learning scenarios, it can be used to model best practices in face to face environments as well. However, in the face to face context it can be desirable to dynamically adapt the particular order and even consider several alternatives that will be arranged by the teacher according to the particular development of the session.

3.3 Best Practices in the context of this thesis

Best Practices can be defined as a set of recommendations regarding a task, process or product to achieve effectiveness in the proposed goal. In a classroom environment, all these kinds of recommendations apply: identification of several *tasks* that are carried out by teachers and students, *processes* that are broader in scope, as well as *products* such as courseware used within the classroom, or homeworks delivered by the students.

3.3.1 Definition: What are Best Practices?

In this thesis, **Best Practices** will be the best among a variety of candidates defined as *a set of recommendations regarding processes, tasks or products, which are measurable and can thus be compared to other uses of the same best practices*. The Best Practice is thus the best known solution appropriate to solve a stated problem in a specified context. In order to be measurable, the recommendations need to be either based on easy to monitor measures, or they have to provide some means to measure. Some easy to monitor measures include e.g. time, since a teacher is used to assign time frames to each activity inside a classroom. But in case of other measures, like e.g. the size of the model the students are developing using their computers, it is necessary to know how the teacher is supposed to get that measure. In this context, the need of a tool for the teacher to find out about the classroom situation becomes evident.

As a set, best practices can include several kinds of recommendations. Normally it would be expected that best practices include recommendations of what features or activities should be included, and which should be avoided. When a teacher starts some pedagogical activity, the set of recommendations to follow is chosen. It can be assumed that each defined activity will have a set of recommendations that form a Best Practice for that particular activity. Of course, several activities can share some common recommendations, but it is interesting to be able to refer to the currently significant Best Practices according to the activities to be carried out.

The recommendations contained within the set can be **conditional** or **mandatory**. If they are conditional, the teacher can apply them based on her own criteria, generally in response to the context of the educational scenario. The context is defined by some external factor, that may be left to the criteria of the teacher. As an example, the depth in which some subject matter is to be treated in the learning scenario can be considered when defining the best practices. The teacher might want to adapt the usage of the learning material to the students, depending on their previous knowledge, how they are going to apply the learned lessons in their educational curricula, the motivation that made them take a particular course or the time available for a particular section. This way, the same practices can be applied in a wider range of situations.

Also, in many cases the order in which these recommendations are followed does matter, in that case the information about order should be part of the practices. It will be common, for example, to define what activities should be performed before others. As a trivial example, it obviously does not make sense to ask students to solve some problem before they have had a chance to understand the problem they are facing. But it is possible that a best practice includes the confrontation of an impossible problem as a way to motivate students to go through the activities that will make the problem seem easy to solve.

For the scope of this thesis, it can be assumed that the best practices are associated to each courseware material that contains not only the description of an activity, but also which the best practices for that particular activity are. There is no doubt that several elements of the best practices are relevant for a number of different scenarios, so it makes sense to keep them available in some library. This has several advantages, some technical and other related to usability. On the technical side, it will not be necessary to build the same measurement more than once, when the same or a related recommendation is used in several best practices. It suffices to have a unique reference for a recommendation, and given the case, define the parameters for that recommendation. For example, when a recommendation states that each student should be participating in a collaborative group activity, a parameter could include the maximum and minimum group sizes. Additionally, when teachers use the same measurements when following several best practices, defined in different courseware materials, they will feel familiar with the use and need lesser efforts to use them.

3.3.2 Teaching Strategies

Some work has been conducted by Van Marcke through the authoring tool GTE (Van Marcke, 1992), in which teaching expertise is represented using tasks, methods that carry out those tasks and objects being employed by the method. The objects can be understood as units of learning materials that are used to represent already selected contents in the best way, for example by arranging a sequence of explanations, exercises, tests and exploration.

Best practices can be specified as specific teaching strategies, but not exclusively so. Teaching

Strategies generally refer to generic strategies that can be applied to several concrete learning scenarios, whereas some best practices can be specific as we will see in the next section.

McCalla proposes an *ecological approach* in which the courseware available to students is grouped according to the teaching strategy. Learners interactions with learning objects are captured by way of attaching models of learners to the objects (McCalla, 2004). Once the information is captured, data mining techniques are used to obtain information that allow the improvement of the material and teaching strategies.

The idea of using the information to improve the strategies and materials is similar to the concept of developing best practices by sequential improvements. However, the approach used in both projects is different since in the approach presented in this thesis the teacher is monitoring information during the development of a session. This implies that the session being monitored is one at a time, while the approach of McCalla combines multiple experiences and makes efforts to find similarities across them.

3.3.3 Granularity of Best Practices

The granularity of the recommendations is an important issue, as it has been discussed above. Recommendations are needed in several levels of abstraction, but it is possible to have overly broad or too specific recommendations, which do not always contribute to improve the learning results.

If recommendations are too broad, it becomes difficult to find accurate measures that make the recommendations comparable across several scenarios. In that case, the recommendations may be helpful on another level, but it is difficult to provide computer support on the level being discussed in this thesis.

On the other hand, if the recommendations are overly specific, two problems will be evident. The first is that the recommendations will make sense only within the scenario in which they were defined. The measurements related to that recommendation will also be specific and it makes the process less practical, since it is not possible to apply the same recommendation (or a parameterized version) in another setting. Any change in the context will make the measurement provided for that particular recommendation lack any real meaning. The second, and arguably the most serious problem is that as ever more specific recommendations are made, the number also increases, as each recommendation embraces only a tiny part of the overall activities. The teacher will need to dedicate much time to the management of recommendations when the set of recommendations is too big. As a consequence, te teacher will not be able to dedicate more time to important tasks, thus failing to achieve the main objective.

A trade-off exists between both possibilities: providing too much detail and number of recommendations versus the difficulty in mapping the recommendations into actual actions to be done. In the case of recommendations for the classroom sessions, they can be classified into general recommendations, such as guidance by a known pedagogical methodology, or more specific, including aspects particular to a given subject matter or domain, or even specific to a particular activity. Therefore, the following granularity for recommendations is defined:

General

A General recommendation does not contain any domain-specific aspects, or alternatively, all domain-specific aspects of a recommendation can be defined as parameters. This kind of recommendation can be applied to a wide variety of situations, and should be included in a basic library of recommendations. Using an identifier, a Best Practice can include a recommendation by indicating said identifier and optionally some parameters to further define its application.

Recommendations that would fit into this category are the following examples, where the underlined words are parameters for the recommendation:

- set a defined order on the planned activities contained on several files on the repository, first file param1, next file param2, next ...

- define groups of <u>min</u> to <u>max</u> number of people per group for collaborative sessions.
- create a snapshot of the current document for all students, for later use.
- leave the students working on the modelling of a solution to the problem until the model reaches <u>numberN</u> nodes and <u>numberE</u> edges.

Domain-specific

The recommendations may include some information that is specific to the domain being treated in the classroom activities. Since the recommendations are tied to the domain, it makes sense to provide these recommendations either as a library associated to that specific domain, or even included in the courseware for the activities that make use of them. This way, teachers who are not interested in that particular domain do not see the recommendations and thus they do not get confused on the existence of additional parameterized measurements.

For example, following recommendations are domain specific:

- (using the Java Reference Frame (see 7.3.2 on page 124)) find one solution that includes the usage of the class `java.util.Enumeration` and ask the students to discuss it.
- (using the Stochastics Reference Frame (see 7.4.2 on page 130)) find one solution to the Stochastic Model that is not making use of the `Lottery coupon` filter.

Whereas "find the most active student and ask her to explain her work" is not domain-specific. It is possible to apply the same recommendation in several situations where different topics are treated. In particular, it is possible to apply this query in both scenarios mentioned above in the domain specific examples.

Scenario-specific

The recommendations that are adapted to a specific task or classroom session are considered scenario-specific recommendations. It is possible to define a set of classes that group scenarios sharing common characteristics together. By grouping the scenarios, it is not necessary to redefine the recommendations for each scenario, but rather to classify it into one of the existing classes, or to define a new class when none of the existing ones is applicable.

When considering recommendations for specific scenarios, it is possible to consider characteristics of the participants, which are not known in general. Aspects such as age, previous knowledge, background and motivation can influence that one set of recommendations is better suited. The recommendations considering these aspects would be both, general and scenario-specific.

Another way to classify recommendations as scenario-specific is considering what specific knowledge of the activities is necessary to specify the recommendation, being this knowledge not applicable to other activities, even in the same domain. For instance, when a stochastic model allows several different solutions, a recommendation can suggest that the students are divided into groups in a way that all of the possible solutions are present in each group, for discussing the differences and delivering one solution considered to be the best. As the identification of the different solutions needs information about the details of the task, it does not make sense in any other scenario. This kind of scenario-specific recommendation is also a domain-specific recommendation.

3.3.4 Definition and Validation of Best Practices

It can be seen that the creation of Best Practices is not an easy task, and it is not the only issue. At least as important is to know if the same Best Practices can be applied in other contexts, if they can be improved or even whether they are properly defined. Also, when the recommendations defined as best practices are followed, it becomes necessary to know whether they were implemented effectively, that is, if they have been followed accordingly. These activities are part of the validation of best practices, a requirement before considering wide adoption.

Defining the best practices for a given classroom situation is a complex task that requires the coordinated work of a multidisciplinary team including experts in pedagogy, teachers, experts in the domain that is taught, and possibly others. This team uses previous experiences to find patterns that may indicate a good way to reach a certain set of goals. At this point, it is interesting to have information about the experiences that indicate if certain characteristics repeat in successful ones, or if there appears to be no relationship about one characteristic and the result of the experience. Also, rebating candidates can be essential: when the application of a good practice fails to deliver the expected results, and the recommendations were followed correctly, then the recommendations need to be changed. This change might modify the recommendations to fit the newly discovered case, or they can be dismissed as best practices candidate.

The questions to be answered for widely approved Best Practices are the following ones:

1. Does a specific characteristic repeat among successful experiences? If it does, it is a good candidate to be included as a recommendation into Best Practices.
2. Were the best practices effectively implemented, or are there mistakes following the recommendations? If more information about the intent of each recommendation within the best practices and intermediate results to be expected is available, the revision at this level is easier.
3. Are the results as expected? If yes, this would add to the validity of the defined practices, but a negative answer would indicate that something is missing. Maybe the practices did not include all of the needed parameters, or some other problem surfaced.
4. Do the same Best Practices defined for one situation apply to a slightly different scenario?

3.3.5 Assumptions

Regarding Best Practices, three related phases need to be supported:

1. Finding patterns in successful experiences that can be used to define Best Practices.
2. Validate or refute proposed best practices, as well as applying them in different contexts to extend or limit their scope of application.
3. Determine whether best practices were correctly applied in a given context, possibly giving feedback on how to improve the definitions.

In order to aid educators in each of these phases, all information available within the classroom environment or even a distant-learning scenario can be used. Several tasks can be automated and in this way provide valuable information for a teacher or tutor to interpret.

Assumption 1: a person with knowledge about pedagogical issues is in charge of interpreting the information being provided.

To do this, a set of assumptions should be defined for the Best Practices to fulfill, so that each of the recommendations be measurable. Only when this assumption is correct, it is possible to to determine whether the recommendations have been followed.

Assumption 2: each recommendation forming part of the Best Practices has a measurable objective.

That is, it is possible to establish a quantitative value that determines whether the objective has been met after following the recommendation. For example, let us consider a recommendation that indicates to perform group work in groups of 3 to 5 students, for 10 to 15 minutes in order to solve a specific problem. We can actually split that recommendation into three parts for which each has its own measurable objective:

1. Split students in groups of 3 to 5 each. Measure: number of groups having more than 5 or less than 3 students.
2. Perform work during 10 to 15 minutes. Measure: time elapsed since begin of work for each group.
3. Solve a specific problem. Measure: progress in percentage of each group solving the problem.

Since each measure can be compared, it is also possible to establish some referential values and a range that is considered normal. When the measure falls outside of that range, some action needs to be executed, like applying a new recommendation, repeating some previous recommendation or require other teacher interventions. In the example, the teacher might decide to take some students from one group to another when the number of students in each group is no in the desired range. When the students finished sooner than expected, the teacher might want to skip a section that is not necessary since the students have presumably understood the explanations well enough to solve the problems under the planned time.

3.4 Relevant Information

In order to provide recommendations that consider the current events inside a classroom, it is necessary to consider relevant information. The relevant information enables us to issue different recommendations based on the situation.

- Documents used by students:

  The documents currently in use by the students will probably be the most used information source, since they contain the direct interactions of the students during the sessions. It will be possible to determine the progress of all students in some task, analyze their work and more. It is clear that the recommendations will vary according to the progress (or lack thereof) of the students during a classroom session. This information is needed for any recommendation that is based on the modelling of the users, including whether the students are taking the right path or making common mistakes, which may require teacher intervention or used as a practical example.

- Metadata: this includes all information that is not explicitly created by the users, and is not part of their modelling. Using this information, it is possible to identify if students are exchanging documents as part of the collaborative work. Also, it can determined whether the documents provided by the teachers along with the course material are really being used as a base for the personal or collaborative work, precise date when parts of documents have been changed, and other useful information. This category includes information about the courses, students in each session, etc. In any recommendation that is based on the state of several students, this information is used to get aggregate information from each student's data inside the classroom session.

- Logfiles: in a Computer-integrated Classroom, it is possible to keep a log of each interaction among students, between a student and the repository and even the actions inside a single student application. This information is useful to determine whether students are following the instructions, or using available resources and interacting among their peers.

  In the case of the CiCv2 (5), it is possible for the applications to interact using a server called MatchMaker (Jansen et al., 2001; Jansen, 2003). This application enables the underlying model of documents to be shared in real-time among several participants, enabling users to see current modifications as they occur. This application generates its own log of events in several XML files which can be used to obtain valuable information in this system. It is possible to analyse the interactions into very specific details using these logs. Recommendations can make use of the information provided in the logfiles to, for example, determine the activity of students, as well as verifying the correct submission of the assignments.

- External information: some queries may get information from outside of the system, be it external XML files or even information that is typed in as a parameter for that particular query. The teacher may indicate the precise phase inside the session plan that the group is currently in, or adjust any parameter to make the recommendations suitable for the specific situation.

Based on the need for this information, it is evident that a knowledge acquisition tool, in the form of a querying system that can access the data available in an environment like a CiC, is a good starting point for determining, validating and using Best Practices. It is necessary to provide all of the presented information inside a CiC, and to develop a query infrastructure that can be used by the teacher in that environment.

Chapter 4

Measurable Recommendations

In an educational setting, particularly considering the evermore possible reuse of educational material, courses and materials are constantly evolving, expecting to improve the overall learning experience. This gives birth to a sequential improvement, trying to reach better and ever more complete ways to learn and teach several subjects.

As courses and materials keep improving, it is necessary to evaluate the results. A good way to assure good initial quality is keeping close to currently known best practices applicable to general contexts. On the other hand, it is also possible to define or refine best practices to fit specifically into the subject being treated. In both cases, the burden of definition/adaptation, validation and application of best practices is high, and no systematic process to perform these activities does exist. According to Ruchti (2002), *"best practices" are usually developed through anecdotal and descriptive research only*.

This thesis addresses the problems related to the definition, validation and application of best practices using courseware in a classroom. The system to be used in this context is a specially developed Computer-integrated Classroom (CiC), called CiCv2 (see 5 on page 74). This environment is based on the general idea of a CiC (see 2.7 on page 28) in which the students work, performing collaborative activities and being guided by a teacher.

Using the information available in this environment, it should be possible to identify common patterns in both successful and unsuccessful experiences and verify the correct application of current best practices in any session. The teacher will have access to a set of queries that can deliver information about the current status of several aspects of the classroom situation. Using the results delivered by these queries, the teacher will be able to compare the results to the expected or needed ones according to the recommendations given by the Best Practices directives, and thus interpret whether the guidelines are being followed according to the plan.

4.1 Relevant information

A first step towards definition and validation of best practices is to identify which information is relevant for the task. What is needed to know is whether the proposed best practices are being followed in a particular situation, and whether the result of that situation was successful. The latter will be evaluated by the teacher through some kind of assessment. The former is where access to the information generated by the students is needed.

Since the students are the subjects on which the learning process should evidence some effect, it should be expected to find out the current progress by looking at what they are doing and have done before. Additional information about the context in which the activity takes place has also to be considered, but the context is much less likely to change. Instead, the information produced by the students is expected to show improvements towards a defined goal, so it will change accordingly.

Therefore, it is important to be able to analyze the information produced by the students in their normal classroom activity, not once, but several times during the sessions. This way it can be established what recommendations of the currently applied best practices were effectively followed, what recommendations were not implemented in this session and confront this information with the outcome of that particular session.

4.1.1 Document Differences

In this thesis, differences among documents play an essential role in obtaining the needed information. Students will modify documents, and the relevant information is almost always in those modifications. Therefore, it is more valuable to have access to the changes among documents than only the complete documents.

A teacher may want to be able to look in an efficient way at comments a student adds to a document, whether in machine-readable text or in handwriting. Using document comparison the teacher can be able to see if there are any differences at all, in which places they appear and she could even be able to group annotations made by different students to the same source document. It is important to distinguish differences that are substantial from differences that are relatively simple. For example, when a student moves a page or a paragraph from one place in the structure to another, it should not be considered as the elimination of the element and the inclusion of a new one but as a move operation. Another possible usage of differences is to improve performance of repeated document queries, avoiding to recalculate queries on documents or portions of documents that have not been changed since the last query.

4.2 Example

For example, the following practice that suggests a specific order in a classroom session:

1. presentation of the main points by the teacher
2. individual work
3. collaborative work
4. homework

This is a very general example, and it is possible to find practices related to more specific aspects of the matter treated in the classroom sessions. A teacher will have access to the information in the classroom environment through the use of queries. These queries are presented to the teacher using a visual language, and are packaged to specifically fit the information needed for the recommendations in the particular session. Therefore, the teacher only needs to make use of the results of the prepared queries to take the decisions suggested by the recommendations as part of the best practices. In the example presented in this section, following aspects can be considered:

4.2.1 Presentation of the main points by the teacher

In this scenario, the teacher will probably be using the electronic whiteboard to explain some fundamentals that will be used in the following activities, as well as some advice on the activities themselves. The objectives of this activity can be diverse, so it is necessary to have a definition of what the precise objectives are in this particular case. The objectives are therefore not equal to all sessions in which the best practices of this example are applied.

Suppose the teacher is using material that is also available to the students in the form of an electronic document. In that case, it should be asked whether the students are using that material, if they are making personal annotations on the documents and whether they are consulting additional sources (glossaries or dictionaries for example). The access to the material is evident from the logfiles of the CiC environment, and it is possible to calculate the differences between the original file and the one opened in the student applications for changes.

This information is valuable not only to verify the correct application of the best practices in this situation and to validate the practices accordingly, but also to verify best practices in the creation of the material. If the students are making a especially high number of annotations on a particular page of the material, it is a signal that this page might be specially interesting for the students. Or rather, that page might contain some errors that students are trying to correct, or they need to make annotations to better understand some concept that is not being explained the best way.

4.2.2 Individual work

For the individual work, the teacher might send some prepared material to the students using the CiC system. This would be used during this phase, and also would mark the change of activities inside the classroom. If no such material is used, the teacher might signal the change in some explicit way to the system.

During individual work, students are expected to work on their material, creating some outcome. The teacher can ask for what material the students are currently accessing, and for example conclude that none of the students is considering some part of the material. Whether that fact is a consequence of not properly emphasizing the material in the previous exposition, lack of interest on behalf of the students, bad design of the documents or sheer coincidence has still to be determined, probably using some other information provided by the system as a way to get to it.

The teacher may decide to intervene in order to accomplish the objectives. For instance, the objectives of this activity might be to spread the whole contents across the students in order to cover all of it and work in groups discussing all the aspects in the next activity. In that case, not covering all of the material would be a case requiring action on behalf of the teacher to fulfill the objectives stated in the best practices.

Additionally, the teacher may see the progress of the students, for example evaluating how many students have started to work on a specific task that consists of filling a part of a document. This way, the teacher can monitor progress and determine if the 15 minutes of work assigned to this activity are appropriate or if the students need more time or some have already finished after half the time.

If it is possible to classify the solutions developed by the students into a set of classes, it is possible to use that information in some pedagogically meaningful way. For example, by assigning groups in which each of the solutions are being represented by at least one student, and comparing them with other possible solutions. Or by having one representative of each class of solutions explain his or her reasoning in developing it to the rest of the class. In the case of answers in natural language it might be difficult to reach a classification like the one proposed and in the case of multiple-selection answers the results might be too obvious, but there are really interesting cases in which the results can be quite useful. For example, in the modeling of petri nets or other form of mathematical model, solutions can be classified into a meaningful set of cases.

4.2.3 Collaborative work

Generally, the first step of this activity is to create a number of groups in which students will work together. The decision on what groups to create can be made by looking at how the work in the previous phase was carried out. The best practices can state a recommended way to group the students, based on what material they decided to work on, as a way to suggest groups in which each student has worked on different subjects, hopefully covering all of them with each group of students.

Also, a teacher might want to see if the groups are interacting, if there is collaboration among the groups rather than only inside them and what material is being used (for example, if the contributions of each group member are too asymmetric). The same information about progress that is used in the individual work activities can also be applied in this phase.

4.2.4 Homework

The teacher delivers a document containing the description of a homework using the system. The students are expected to deliver the results in form of a new (or a modified version of the same) document within a certain period of time.

This phase is different from the previous one in a very important aspect: the teacher has no additional information than the documents submitted by the students. In the case of the classroom scenario, the teacher can interpret the data provided by the system by complementing it with her own observations inside the classroom. This phase is more like the e-learning scenario, in which the communication channels rely only on the digital media, and in this case the communication also is asynchronous. The documents can be used to perform some automatic evaluation of the results and provide statistics of various kinds.

4.3 Sample Information Needs

In several different situations, the same information is valuable for correctly determining, validating and applying best practices. The information presented below is useful and thus a requirement to be provided for any knowledge acquisition tool.

4.3.1 Classroom Snapshot

This information will be used for the purpose of providing, as its name suggests, a snapshot of the classroom situation in an instant. Every student application generates a serialized version of the current document, sending the result to the repository where it is saved indicating the current date and time.

When repeating the process regularly, it is possible to simulate any particular instant of the classroom session, allowing the teacher to re-create all data for any moment in the classroom session.

4.3.2 List Students

Much of the information needed include the status of every student. Having a complete listing of every student present in the session is a neccesary condition for that. Since every student normally identifies herself at the beginning, this information is present in the system and can be used directly, without any need for teacher intervention.

4.3.3 Specified Page from Student

Once it is possible to identify all students in a session, it is useful to get details of each student's achievements during the session. The access to any page enables the teacher to see the progress of each student, retrieve a page in order to use it as an example of a good solution of the proposed problem, or to show a common error. Also, this information can be used to further process it locally, although a better approach is to process most of the information on the student application, to make use of the distributed nature of a CiC environment.

4.3.4 Current Page from Student

The teacher can retrieve the currently opened page from a predefined student. The teacher may use this query to view the work a student is doing by directly reviewing the page he or she is working on. This is a rather slow process, specially when the teacher would access the current page for every student. Generally however, the teacher will use other queries to gather information and, based on this information, she will choose to take a closer look at the work of certain students.

4.3.5 Diff Count

This is the calculation of the differences between a specified file in the repository and the current document. The number of changes between both documents is counted, separated in categories: deletions, updates and inserts.

4.3.6 Opened File

A teacher can determine from the information in the CiCv2 which students have opened a specific file. The teacher can use the tools available in the CiCv2 to deliver assignments to the students. When the students receive the file, an icon appears in their Inbox indicating the availability of the document. The students can then click on the icon or open the file using the menu bar, and this triggers an entry in the student log. Using a StudentQuery, and indicating the name of the file the students should have opened, the teacher receives the list of students that have effectively opened the file. This information can be presented in several forms according to the teachers need: as a specific list of students, the inverse form indicating students that have not yet opened the file, or as a summarized information indicating the number of students having opened the document and the number of students pending.

4.3.7 Student Activity Indicator

Activities performed by each student during the last couple of minutes is useful information. It allows a teacher to identify several parameters, e.g. how long it takes for students to work on a given task and whether they start right away or need some time to evaluate how to start. Also, the kind of activity matters a lot, so it is useful that the activity is categorized into a set of possible groups. Just like the previous example, this information can be presented to the teacher in detailed form or as a summary.

4.3.8 Collaboration list

The collaboration in an environment like CiCv2 generally consists of several groups working together. Just as the list of students is needed by the teacher to get precise information, so is the list of collaboration groups that are active in the system. This information will generally be used to gather other information later on, including the activities that are taking place inside each of the groups, how those activities are divided among the participants and other possible information.

4.3.9 Model Complexity

For each active student, it should be possible to generate a row in a table, detailing for each page opened in the student application the title, number of nodes, number of edges and number of strokes. This information helps a teacher to compare the density of the models each student is managing, and having an approach of the average complexity by means of that density (number of nodes and strokes in a model). The resulting objective information can be used in several interpretations.

4.3.10 Summarized Information

Most of the information needs mentioned above are specific and can be summarized for the teacher to have an overview. In most cases, the teacher does not need nor want to have all of the details. The summarized information is enough to spot specific problems, for example that not all of the students are working on the test they are supposed to be evaluated on. Moreover, it is possible that the teacher can solve most of the problems without needing access to a more specific set of information, in the test example by reminding the class to work on the test file and asking whether anyone is having difficulties to access it.

In cases where the teacher needs to know what specific student is having difficulties solving the problems, it is possible to ask for the specific information. The teacher will then be able to identify which students to address and to offer advice. Likewise, the teacher might know in advance, e.g. from previous test results, what students need more attention to fulfill the goals. In that case, the teacher can specify to receive specific information for those students and compare them to the summarized information available for the rest of the students.

4.4 Queries

In section 3.3.1 (page 41) it was defined that, in the context of a CiC, Best Practices are understood as *a set of recommendations regarding processes, tasks or products, which are measurable and can thus be compared to other uses of the same best practices*. In some cases, the measurement(s) associated with each recommendation can be naturally managed by the teacher alone. However, there are plenty of other cases in which the teacher might need more information than she has access in normal circumstances, and the recommendations have to include some way to access that information.

A tool will be provided in the form of queries that can be used by a teacher to get some form or measure to be used when following the recommendations. These queries are defined at the same time the recommendations are written down, so they become part of the Best Practices. The teacher will find the recommendations accompanied by a set of queries that will help her to apply the best practices.

4.5 Querying Java Model and DOM

Let us consider a scenario in which the students are studying stochastics and use a CiCv2 (described in 5) environment to model and simulate stochastic processes. During the modeling, students will modify parameters, add and delete elements and relations between those elements. During a first session, the teacher makes a summary document available to the students and asks them to model two different processes (Process A and Process B, for which each student develops Model A and Model B, respectively). The students may then finish the rest of the models in a second session. For each document in the CiCv2, two different instances can be found, depending on its state:

1. Document stored in repository: the document is not being edited, or the fact that a copy of that instance might be in editing mode is unknown. For example, a student created Model A

in the first session, and is now modifying Model B, which is stored in a separate document. The document containing Model A is available as an XML Document Object Model (DOM) in serialized form. It has to be noted that the same document can be made available inside a Java application as the non-serialized version of the same Document Object Model in primary memory. In both cases, the data in this model will be referred to as **the DOM representation**.

2. Document instance opened by an application: the document is being edited within a classroom session. Using the same example as above, the student is currently modifying the document containing Model B. An outdated version of the document may be stored in the repository, lacking the changes made by the students since he last saved the document. As the interaction inside CiCv2 is synchronous, the relevant information to the teacher is probably only in the model of the application that is editing the document. This information is in an application-specific format in primary memory, and will be referred to as **the Java Model representation**.

It has to be noted that in both cases, the whole information about the current state of the document includes any modeling the student may have done or, in the latter case, is currently doing. This is due to the use of the Model View Controller pattern that is used to implement all of the modeling elements, as described in the next section. When the student for example changes some parameters, they are immediately reflected on the model, so there is never an inconsistency between what the students sees on the screen and the corresponding document model.

Of the two possible states of the documents that will help the teacher to analyze the outcomes and development of learning activities, it is clear that there exist two corresponding ways to extract information, one for each representation of the data as described above. For data in the DOM representation, the document files would be parsed into main memory and the DOM accessed directly, using XML-specific standards and tools. For data in the Java Model representation, application-specific methods and tools would be used to manipulate the data, either *in situ* or a copy thereof.

Given the need to perform queries on documents in both states mentioned above, the fact that both states are equivalent means it is not necessary to develop two different ways to process the data. It is possible to treat documents in any of both states using the same tools or methods, just transforming them from one state to the other. This allows us to save a huge amount of work and it simplifies all queries.

4.5.1 Equivalence of Java Model and DOM representations of the data

Since the DOM representation of the data is a transformation of the Java representation, it can be concluded that they are equivalent. It is possible to create the DOM representation starting from the Java Model, for example by loading the document containing Model A into the application for review or modification. The other way around, it is possible to restore a Java Model based solely on the DOM representation, such as saving Model B into a file on the repository. However, some portions of the Java Model may contain more information, that is discarded when transforming to the DOM representation, and redefined in some way by the application when transforming from the DOM to the Java Model.

The most significant difference between both representations is that inside the application, a Model View Controller (MVC) pattern (Krasner and Pope, 1988; Buschmann et al., 1996) is used to represent the objects. In the DOM representation, only the corresponding Model is present, and in the application, the View is also available. Moreover, an object Model can be accessed in the Java Model only through the reference present inside the object View. This view includes data about its size, which is a parameter that could potentially be used in queries. But one of the characteristics of the implementation of the MVC pattern used in CiCv2 is that the view can be changed. Therefore, the size that results when instantiating the model does not have a significant meaning, since it depends of the particular interpretation that the application makes. Two different application instances can assign different views to the same object model, so view-dependent elements such as size, color and others do not make much sense when performing remote queries.

It is important to consider that the model has to include all of the relevant information that will be used in any query. Once this has been achieved, a real equivalence between both representations is achieved. It will thus be possible to use any of these representations to process the queries with essentially the same results. The result of analyzing Model A by using the document stored in the repository is the same than analyzing the same Model A once it has been loaded into the application.

However, the performance of transforming from one representation to the other is not quite equivalent. Due to the processing necessary to set up the View and Controller part of each object inside the model, the transformation from the DOM representation into the Java Model representation is more complex and thus time consuming. In tests, the process of parsing an XML Document into an in-memory DOM representation and then transforming it to the Java Model representation took several orders of magnitude longer than generating the DOM representation from the Java Model representation and saving it into an XML Document. In the latter case, the processing is simpler because it is only needed to extract the relevant subset of the whole information that represents the model part of the Model View Controller pattern.

4.5.2 Querying on the Java Model representation

When using the Java Model representation, it is possible to take advantage of the objects defined inside the application. Let us consider Model B that is currently being edited by the student, as described above. The objects include helpful tools that provide access to aggregate information, execution of methods and some other specific properties. The same information is also available through the DOM representation as just presented, but possibly not as direct or easy to use.

On the downside, there are currently no existing tools that might be used to directly query the Java Model representation as present inside the application. In order to perform any query on Model B, all tools would have to be developed from scratch or be heavily adapted in order to be usable. When developed, these tools would not be suitable to use outside of the CiCv2 environment, and any interaction with tools from the outside requires some transformation for the communication of data to work. The alternative is to transform Model B into the DOM representation, achieving the same state that currently has Model A.

4.5.3 Querying on DOM representation

Using the DOM representation to perform queries has several advantages. As XML (Bray et al., 2004) becomes ever more standard and is being used in several educational environments, increasingly useful ways do exist to manipulate and transform XML content like XSLT (Clark, 1999) and extract information from XML repositories using XQuery (Boag et al., 2001).

This approach has advantages because it is the most generic way to tackle the problem and it can be adapted to other XML documents relatively easily. It is possible to interact with the "outside world" in both ways when using XML: by sending XML documents or fragments to an outside application for processing, and by receiving input from outside files or applications that can be integrated seamlessly into the querying infrastructure. Choosing the internal model as the way to access information would mean that the external data should be imported in some way.

Additionally, it will be necessary to transport the intermediate results to the final destination within a distributed scheme (see 6.1) such as is the case in the CiCv2. In order to transport the results, the data has to be serialized and de-serialized, which is trivial using the DOM representation. Would we be using the Java Model representation, the data would be serialized in some way, being the DOM representation the simplest way.

4.6 Processes to define Best Practices

The process of defining Best Practices in educational settings has not been studied deeply. Moreover, most of the approaches regarding best practices are related to software engineering (Paulk et al., 1995) and specially business processes in management (Myers et al., 2004; Bretschneider et al., 2005). Before thinking about defining, validating and adapting best practices inside computer-enabled classrooms, the process through which best practices can be reached and reevaluated needs to be defined.

Best Practices have to be applicable in a wide range of scenarios, in which they are used by several people in different circumstances. Thus, the feedback and adjustments made to Best Practices need to be integrated into a single set of recommendations. As a consequence, the process needs to be collaborative, with a common goal of creating a set of practices suitable to all of the considered scenarios. This is consistent with the main motivations of defining the Best Practices, ensuring that several independent applications of the Best Practices generate the expected results. When looking for other contexts where such a development takes place, some good examples are seen in the Free / Libre / Open Source Software (FLOSS) communities, and in the same principles applied to other contexts such as Arts, Entertainment and Education – commonly referred to as Open Content.

4.6.1 FLOSS: Bazaar-style Software Development

In the first approach to characterizing the software development style used in several well-known Free Software (Stallman, 2002) projects, Raymond states some principles that can be applied to software projects but also to other types of work (Raymond, 1998):

- “Release early, release often”. This premise is also part of another popular software development methodology: Extreme Programming (XP) (Beck, 1999). It is generally not necessary to provide a fully working application for potential users to get interested, and when the users are participating early on in the design process, the end result will generally be of more value to them when their issues are taken into account.
- “Every developer scratches its own itch”. The distributed and loosely coordinated manner in which FLOSS works is based on the autonomous actions of each involved individual. The software generally gets widely tested and in some cases used in ways and contexts the original author never had imagined. This has an impact on the design of the software, and specially on design changes or re-designs.
- “Given enough eyeballs, all bugs are shallow”. When sharing the source code of a program, more people will be looking for solutions to problems and hence the solutions will be

found quicker. This also means the need to provide fixes gets bigger, both because of more users needing the software working correctly and because of the bigger number of problems found, at least initially. At the same time, a big and interesting project tends to attract more developers (generally, most of them are users themselves), which help fixing the problems.

Nothing resembles any sort of central organization in these principles, but rather some chaotic and individual-oriented basic actions. However, just like in Conway's Cellular Automaton "Game of Life" (Gardner, 1970, 1983), these individual actions give place to the emergence of apparently coordinated entities. In the case of Free Software, said entities are projects that compete and in several cases overcome well-funded, centrally organized software developments (Scacchi, 2003).

The same way Free Software is able to generate products of indisputable quality, a development of Best Practices in education can be based on the collaboration of several interested parties. This contrasts with the traditional process for defining Best Practices in Education, which is both centralized and opaque, without the possibility of sequential improvements. But before we are able to start such a collaborative process, it is necessary to define and develop the tools that will support it. The development of measurable recommendations is one of the prerequisites, since it allows the interested parties to share objective information that allows them to compare results of proposed practices.

4.6.2 Collaborative Content Creation

Based on the success of various Free Software projects, several initiatives started to apply a similar methodology to other fields, particularly what has been called Open Content (Cedergren, 2003; Yue et al., 2004). The methodology is based on a license that allows any interested party to use the material (computer language source code, encyclopedia articles, books, chapters or articles) that is publicly available through web servers and other distribution forms. In some cases, the licenses require derivative works to be made available under the same conditions, these licenses are called to have a retribution or copyleft clause. The Creative Commons project (Lessig, 2004) is one example of how the legal framework for Open Content is laid.

The second, and arguably most important factor is a (technology-supported) collaboration network. This network communicates all participants and allows them to share, contribute and use the resources imposing as few constraints as possible. The result of the interactions of all participants is, in many cases, a high-quality product.

4.7 Identification of Situations

The problem of identifying situations within a classroom can be tackled in a number of ways, using both qualitative and quantitative approaches. In the qualitative approaches, the primary tool to gather data is observation, both in person and through devices such as cameras (Paterson et al., 2003; Harel, 1991). Ethnographic methods of observation have been applied to educational scenarios as well (Frank and Uy, 2004), being able to identify situations from an insider's perspective.

Using quantitative approaches it is possible to detect situations based on the data available in the system, and this data can be used in several situations, including complementary information in qualitative approaches. This information can be processed, filtered and aggregated, to be presented to the users as awareness information (see 2.5), who receive this information passively. The users may select what kind of information to receive, and this information is supposed to be relevant in most of the situations in which the system is normally used.

However, in the case of identifying best practices, teachers and tutors are probably interested in more specific information than the awareness information mentioned above. If the information needed by teachers and tutors is predictable, it would be possible to summarize or present the information in some useful way, defined in advance. But the situations in a classroom are not always predictable, and it is not possible to exclude information without knowing the outcome in advance. In order to provide all necessary information to a passive receiver, the amount would be too high, generating information overload (Maes, 1994) which does not contribute to the desired results. In this case, it is convenient to have a flexible way to select the information that the user can access. The user will not need to receive the information passively, but she will request the needed information each time it is needed, possibly in a slightly different way each time. The best way to assist the users in this case is to provide a system that allows them to perform queries on the classroom system.

The information accessed through a querying system can be obtained in each session, without any need to previously set up the classroom to perform observations, and without incurring in the high costs of the observation processes. Moreover, the information is available in an efficient way, and the process can be repeated both in the same session and in other locations.

Considering that best practices are the result of a vast experience, making incremental improvements over time, having a way to obtain relevant information directly from the classroom system is very useful. This information allows the teacher to evaluate the current situation, verify assumptions through the validation of the data and moreover share her improvements with other teachers in order to widen the experience considered in the definition of the best practices. Thus, a querying system is a very useful tool that can help to identify, validate and apply best practices in a variety of settings. Teachers have to know what and how to search inside of the available data, so the courseware should include suggestions and material appropriate to provide this ability.

4.8 Products

The system to be implemented builds on the concept of a Computer-integrated Classroom (CiCv2) as described in 5 on page 74, creating a querying system that allows access to available data in a usable form. One implicit assumption is that the teacher is the one who performs queries, although it will be possible to extend the system in order to allow other participants to do so. A querying framework will be incorporated into the CiCv2, allowing a teacher or tutor to answer the questions proposed in 3.3.4.

4.8.1 CiCv2

A CiC addresses the face-to-face scenario, providing a framework where the teacher uses an electronic blackboard to display and manipulate the learning material during the session. The students have access to the material using notebooks or PDAs through a Wireless LAN or another type of network. The electronic blackboard is controlled by the teacher's application, which is able to interact with the student's applications in order to share contents and use shared workspaces collaboratively.

This system also includes applications which help their users to manage the learning materials, turn-taking and other relevant aspects of the interactions. This helps avoiding disruptive activities like typing filenames or navigating to find information within the sessions. In this sense, it can be thought of as a "classroom management system" that allows not only to manage the learning materials, but also analyze and learn from classroom sessions. The teacher can be able to recognize when some student or group needs attention without having to interrupt their work.

4.8.2 Deliverables

During the development of this thesis, several items will be produced along with the theoretical aspects that will be contributed. The main part of the work will be delivered in the form of a functional CiC implementation named CiCv2, which has the ability to include domain-specific modeling and it is not tied to any teaching environment in particular. The CiCv2 will help the teacher to actively supervise the learning process and outcome, by collecting important information and making it available to the teacher whenever it is needed. The teacher can thus verify if the best practices are being followed effectively in each session.

The detailed list of deliverables is as follows:

1. A functional **CiCv2 environment**, comprising:

- **System applications**. Classroom Applications for Teacher and Student, home environment and Administration module, adapted to make use of the querying framework.
- **A Querying System**. The teacher will be able to use visual queries in order to obtain information necessary to define, validate and apply Best Practices in the CiCv2.
- **Documentation of the delivered system**, including also a separate up to date description of the FreeStyler application which is embedded in the CiCv2 environment.
- **Test data** in the form of simulated teacher, student and course accounts with files as well as logfiles with which test runs of the querying system can be performed.
- **A Query Library** of existing queries for the several usage scenarios exemplified in the test data.

2. A document containing the **methodology**, describing an approach to define, validate and apply best practices in a Computer-integrated Classroom scenario. This document will also include the test results of the experiments performed during this research.

The querying system will allow the identification of a variety of measurable characteristics inside a classroom session. This identification is based on a quantitative analysis of the available data, creating a repeatable assessment of the classroom situation. The analysis performed by the teacher does not have to limit itself to a quantitative situation, however. It is possible to use the information to validate a qualitative analysis done by the teacher or another person.

The methodology will allow a teacher to share experiences with others, based on the queries used to gather data and possible interpretations of the results. Other teachers may then apply these queries in their own environments, compare the results and analyze whether the suggested interpretation makes sense. If the results are not as expected, the interpretations will be enhanced to consider the new cases, or the queries will be improved to consider additional factors. This strategy will improve the proposed best practices until they reach a level of maturity.

It is expected to be able to compute several possible indicators that will be valuable for the process referred to above. They include aspects that could be presented as awareness information, such as progress monitoring, interaction level among peers, idle times between activities, etc. On the other hand, information such as classifying the answers to a problem into a defined set of classes would be much too specific for awareness, since they make sense in a reduced number of situations. The same applies for queries that produce some modifications to the documents, or queries producing a new document as their output.

Chapter 5

CiCv2: A Flexible CiC Implementation

Best practices are formed by grouping a set of related recommendations that should be followed according to the circumstances. In order to apply a given recommendation in a Face to Face CSCL scenario, it is necessary to identify specific information. On one hand, the teacher will be able to determine whether a given recommendation needs to be applied. In some cases it might be necessary to choose from a set of possible recommendations. On the other hand, the teacher can evaluate whether a given recommendation is having the desired effect and take a different action. In other words, the need is to access the information available inside the classroom in a specific way for each scenario. The solution being proposed in this thesis is a framework that allows a teacher to use visually constructed queries in order to get the right information at the right time. This information will help the teacher to determine several parameters in order to contrast them with the expected results. The expected results are defined in a set of recommendations, *Best Practices*, and help the teacher decide whether it is useful to continue with the current exercise, intervene to solve a troublesome situation or continue with the next step in the planned schedule. Examples of information useful in several contexts are presented in 4.3.

It would be desirable to imitate the way a teacher can use teaching aids such as books with proposed exercises and curricular suggestions. Educational material used in CiCv2 should include exercises, assessment recommendations and useful queries as well. These queries should focus on the subject- and domain-specific aspects in order to be useful. The expected output may help the teacher to understand what is going on in the classroom. This output should not be in a final format, but rather provide the essential information that the teacher can refine and adapt to her particular needs in every moment executing general-purpose (i.e. not domain-specific) queries. These general purpose queries would be used to display the information accordingly to the situation, extract aggregate data such as statistical indicators and other applications. Some of the general purpose queries will be included in the system as a library, and additionally a user may construct her own queries to be used in several contexts.

The queries are created using basic building blocks that can be composed, creating new queries (see 6.2). The resulting composed queries can then be used interchangeably as a basic building block in new composed queries. This way it is possible to create queries of arbitrary complexity. The composition is made visually, just as the rest of the interaction with CiCv2, using drag and drop and similar constructs, making the interaction with this framework familiar to all involved users.

The system to be implemented builds on the concept of a Computer-integrated Classroom described in 2.7 on page 28, developing a domain-independent implementation that includes a querying system. This querying system allows access to available data in a usable form. One implicit assumption is that the teacher is the person performing queries, although it is technically possible to extend the system in order to allow other participants to perform queries. Probably this would include access restrictions that block unwanted access to some information or tasks.

A new CiC implementation (CiCv2) has been developed, taking into account the need to include any topic into the teaching materials and the various situations in which the CiC implementation might be used. It is important to consider previous implementations of a CiC like NIMIS (Lingnau and Hoppe, 2002; Tewissen et al., 2001; Hoppe et al., 2000) and Cosoft (Hoppe et al., 1993), in order to adapt or generalize the useful concepts and ideas to a multi-purpose implementation.

The implemented CiC environment, CiCv2, consists of three modules that interact with each other. Figure 5.1 shows the basic principles and architecture. One module is the central repository, available from inside and outside the classroom sessions offering document archive and authentication services. The other modules are the ones used by teacher and students, respectively. Teacher and Student Modules can interact among themselves by sharing documents through the coupling mechanisms explained in 5.5.3, and they use the repository to store and exchange documents.

5.1 Repository

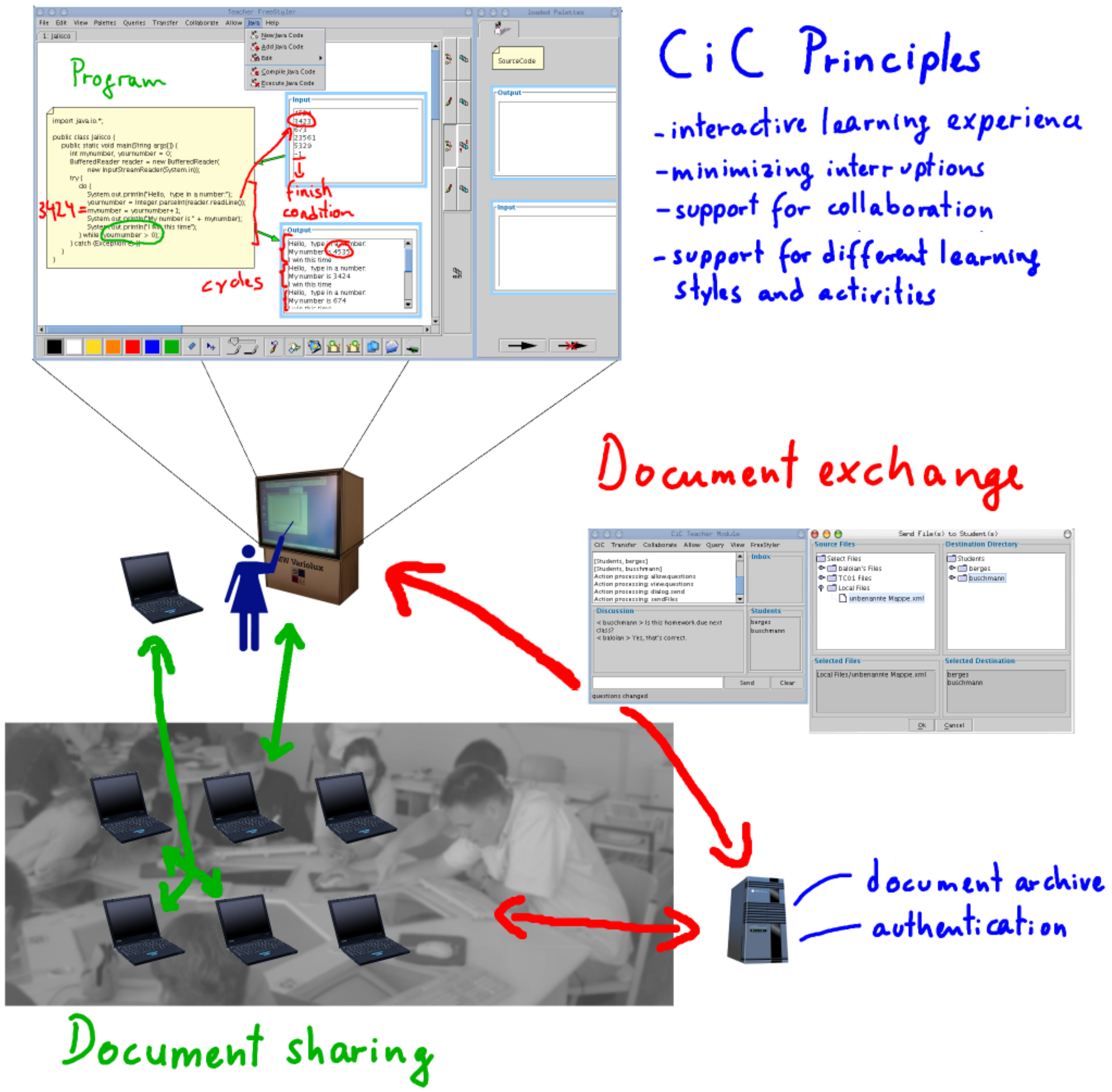
CiCv2 uses a repository that manages several tasks:

- User Authentication
- Session Directory
- File Repository and Archive

Each request is handled in a separate thread in order to avoid an application or network error to hold up other requests. In particular, file transfers can take a relatively long time to complete, and moreover, they tend to concentrate when all students are opening the file suggested by the teacher in the face-to-face classroom scenario.

5.1.1 Session Management

When starting a session, the teacher authenticates at the repository, and afterwards she creates a session for the corresponding course. This session identifies the teacher and includes a reference for the students to contact the teacher application directly. Each session receives an automatically generated ID, based on the current day and the course identifier.



```
import java.io.*;

public class Jalisco {
    public static void main(String args[]) {
        int mynumber, yournumber = 0;
        BufferedReader reader = new BufferedReader(
                new InputStreamReader(System.in));
        try {
            do {
                System.out.println("Hello,  type in a number:");
                yournumber = Integer.parseInt(reader.readLine());
                mynumber = yournumber+1;
                System.out.println("My number is " + mynumber);
                System.out.println("I win this time");
            } while (yournumber > 0);
        } catch (Exception e) {}
    }
}
```

```
3423
673
23561
5329
-1
```

```
Hello,  type in a number:
My number is 4535
I win this time
Hello,  type in a number:
My number is 3424
I win this time
Hello,  type in a number:
My number is 674
```

```
[Students, berges]
[Students, buschmann]
Action processing: allow.questions
Action processing: view.questions
Action processing: dialog.send
Action processing: sendFiles
```

```
< buschmann > Is this homework due next class?
< baloian > Yes, that's correct.
```

```
berges
buschmann
```

```
questions changed
```

```
Local Files/unbenannte Mappe.xml
```

```
berges
buschmann
```

Figure 5.1: Computer-integrated Classroom (CiC)

A log of all event related to the session are stored under the session ID, and this information can be used to extract information. For example, a teacher can determine who participated in the session after it has been closed.

5.1.2 Shared File Access

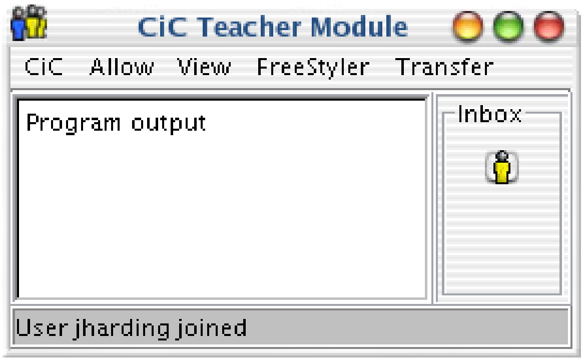
Each user can access the files available on the repository at any time, without the need for a session to exist. The user that requests a file transfer sends its authentication credentials, and the requested action. In the case of file access, the actions can be listing of a subtree, uploading of a file or downloading of a file. The applications request a subtree to enable the user see an overview of the available files and to select one for download or upload.

By using the files a teacher can process the data and obtain important information about the learning process. The applications the students use automatically upload the files to the repository, so even after the student has logged off the session and quit the application, the data continues to be available through the repository.

5.2 Student and Teacher Applications

Students and Teacher use applications that are specific for each role. Both applications share a common base class, `info.collide.cic.classroom.AbstractModule` as seen in figure A.2 on page 157. This class provides all the basic functionality that both applications share, including the connections to the repository for authentication and other requests, initialization of the customized FreeStyler version (described in next section) and other features.

The graphical user interface (GUI) presented to the user once the authentication and session creation/selection has been finished includes several elements as seen in figure 5.2:



User jharding joined

Figure 5.2: Teacher Application GUI.

- Menubar: several actions are available through this menu, most notably the Transfer, FreeStyler and View menus. The menu entitled ”Allow” is available only in the Teacher Application. Each of the actions present in the menus gets logged, so this information is also available for the teacher.
- Log: underneath the Menubar is the Log Panel, which keeps a record of the latest events inside the classroom session. The events are registered in a log file for the teacher to use.
- Inbox: to the right hand side of the Log Panel. This element, which is not always visible, indicates events that are important or need an action. For example, when a student receives a file from the teacher, an indication is shown and the Inbox element is made visible automatically. When the user clicks on the icon shown in the Inbox, the default action for the corresponding message is executed.
- Discussion board: visible in figure 5.3, it allows teacher and students to exchange text messages.
- Status bar: at the bottom of the GUI. It contains the last registered event or output. It gives feedback to the user, indicating that a selected task has been successfully executed or that some error has occurred.

The teacher application also maintains a MatchMaker server that is used to share content (see 5.5.3). The sharing of content also keeps a log for each shared workspace, enabling the teacher to use this information.

5.2.1 File Transfers

The Student and Teacher applications implement several forms of file transfer actions, according to the purpose of the transfer. Each action allows information to be shared, either directly or through the repository when necessary. The following actions can be identified:

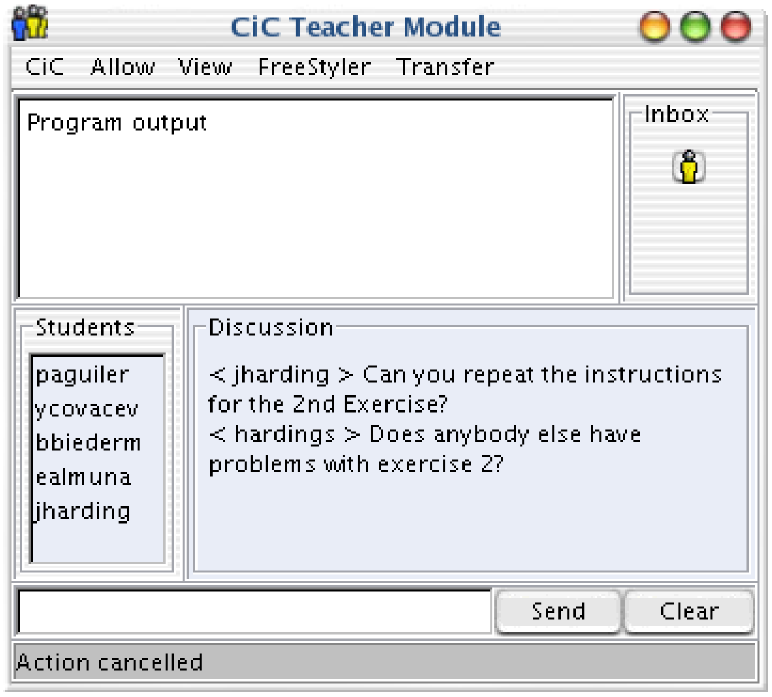
- **Distribute Assignment**: This action is only available for the teacher inside a classroom session. A file is selected from a location that may include the repository or the local working directory. This file is sent to each student participating in the active session, and they are notified of the availability of the newly sent file.
- **Collect Assignment**: This action is only available for the teacher inside a classroom session. A teacher has already distributed an assignment to the students. Afterwards, the students are asked to save their work, by selecting the Save item in the FreeStyler tool. The teacher then uses the ”Collect Assignment” action to save a copy of the file each student has been working on into a location in the repository. The students will not be able to modify those copies anymore.
- **Distribute Homework**: This action is only available for the teacher, not necessarily inside a classroom session. A file is selected from a location that may include the repository or the local working directory. This file is copied to a directory inside the student home directory on the repository, for each student assigned to the currently selected course. No notification is performed, since there is no session to which students could be connected.
- **Process Homework**: A student connects to the repository and searches for Homeworks distributed by the teacher using the ”Distribute Homework” action. The Homework gets copied to the local working directory and the student can begin to manipulate the file.
- **Commit Homework**: After finishing a homework, the student would use this action to send the file back to the repository where the teacher will collect them at a predefined date.

- **Collect Homework**: This transfer mode is only available for the teacher, not necessarily inside a classroom session. The teacher retrieves the files supposedly delivered by the students using the ”Commit Homework” action.
- **Place/Get file(s) in Archive**: It allows a teacher or a student to send a local file to the server or get a file from a server to use it locally. When using FreeStyler files, these actions are not necessary, since the retrieval and uploading of the files from/to the repository is automatically executed.
- **Send file(s)**: Similar to the ”Distribute Assignment” action. The teacher selects a source file and any number of students to whom the file will be sent.
- **Get file(s)**: It allows the teacher to retrieve files from the students inside a face-to-face classroom session.

For each case, a file transfer needs a selection of one or a group of sources and destinations. The file transfer is also used to automatically retrieve files from students and the repository when the teacher needs access to certain information.

5.3 Discussion Board

The CiCv2 GUI, as shown in figure 5.3, allows text-based teacher-student communication. The students can send messages to the teacher, and the teacher can send messages to an individual student or to a group of them. The interactions taken place through this tool are added to a logfile and can be accessed at any time to retrieve information.



paguiler
ycovacev
bbiederm
ealmuna
jharding

< jharding > Can you repeat the instructions for the 2nd Exercise?
< hardings > Does anybody else have problems with exercise 2?

Action cancelled

Figure 5.3: Teacher Application GUI showing Discussion Board.

5.4 Permissions (Allow)

A teacher can restrict some of the features in the CiCv2 sessions using this menu, selecting either "allowed" or "not allowed". Currently, the teacher can control two features:

- **Student Questions**: by disallowing student questions, the discussion panel keeps deactivated and students cannot make use of it.
- **Student Sessions**: normally, students do not start their own collaborative sessions (see 5.5.3). However, if the teacher permits it through the use of this feature, the students can start sessions on their own. This is specially useful when giving the students freedom to choose their own groups for some collaborative activity using the computers. It is not necessary for the teacher to define the groups, but the students can organize themselves.

The usage of this feature enables the teacher to define whether students are operating in a controlled or free environment, according to the particular pedagogical methodology chosen.

5.5 FreeStyler

FreeStyler (Hoppe and Gaßner, 2002) is a flexible application combining freehand annotations with visual language elements and can be used in various situations like modeling, whiteboard control or presentations. The documents generated and used by the FreeStyler application are divided into pages, and the storage and retrieval of the documents is based on an XML model.

Each of the pages used by FreeStyler includes several layers that are superimposed over each other. It is possible to visualize all layers at one time, defining the order (bottom to top) in any possible configuration. Also, it is possible to hide any of the available layers at any time. By default, each page includes four layers: two layers for freehand drawing (cf. 5.5.2), and two layers based on the JGraph (cf. 5.5.1) which allows the usage of visual languages to perform a varied range of modeling.

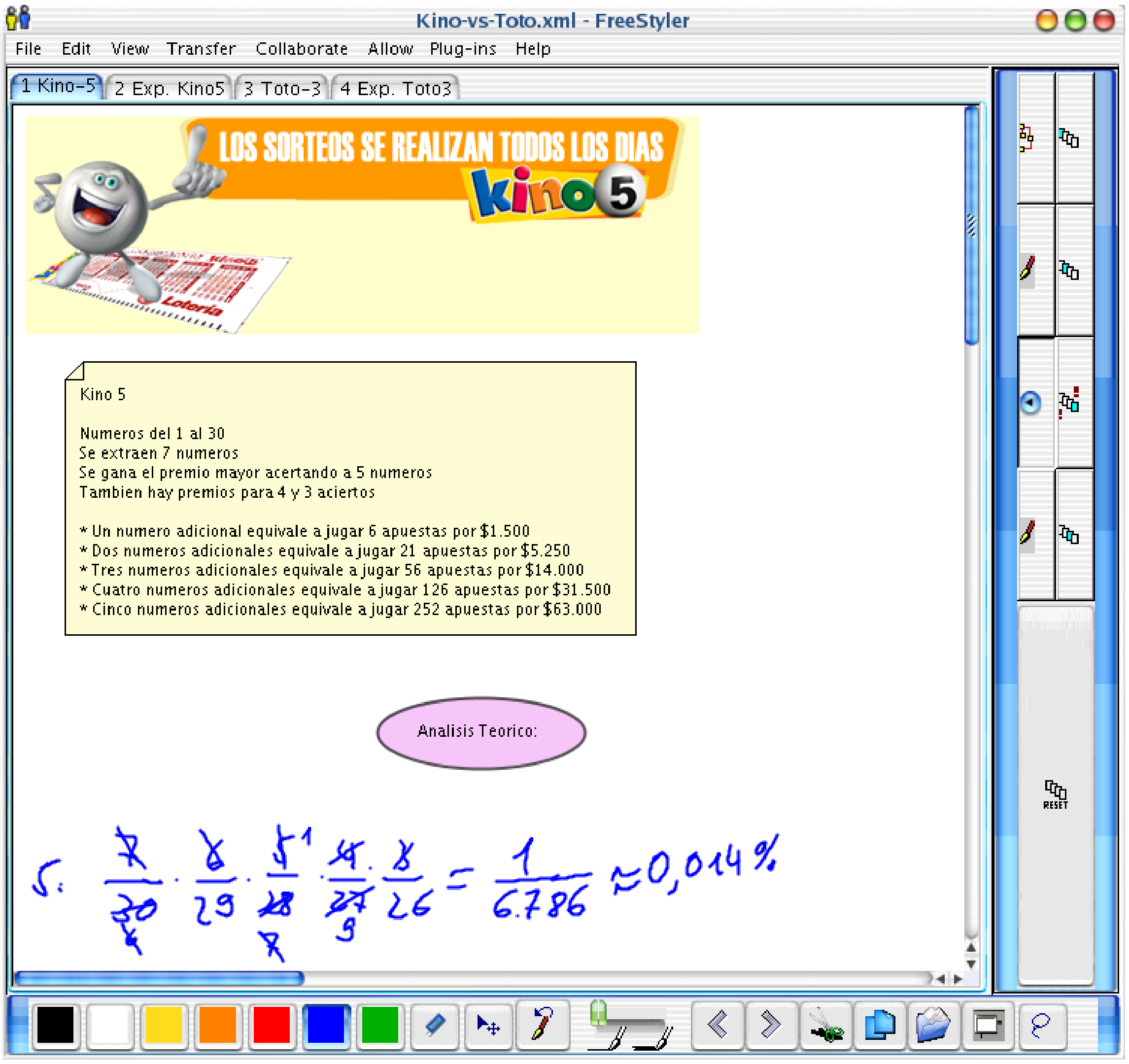
Figure 5.4 displays the FreeStyler application in use, showing a page in which images, nodes corresponding to a visual language and freehand drawing are used simultaneously. The freehand drawing is specially useful in a setting in which a freehand input device is available, as is the case with an electronic whiteboard, a tablet PC or many modern PDAs.

In the next sections we present a technical overview of the implementation. The objective of the FreeStyler application presented here is to allow the teacher to manage the classroom sessions as she best sees fit, including the possibility of using tools targeted at supporting Collaborative Learning. The availability of all relevant information for monitoring is a central part of the development and is described in more detail in Chapter 6.

5.5.1 JGraph

JGraph (Collide, 2003) was developed by the Collide group at the University Duisburg-Essen, and uses the Model-View-Controller (MVC) paradigm to provide the modelling of visual languages in a Graph. The implementation of the MVC in the Collide JGraph case has some limitations, such as not allowing the use of several different views for the same model at one time in the same application. This is the cost of being able to provide the synchronization or coupling (cf. 5.5.3). However, it is possible to display the same shared model using different views on different applications, which relaxes the restriction mentioned above.

JGraph allows two basic types of objects to be used, as expected for a graph: Nodes and Edges. Nodes can have any level of complexity, and are interconnected by Edges. It is possible to define rules that need to be followed to determine whether a particular connection of two nodes by way of a particular type of Edge is allowed or denied. Additionally, a Node can access its neighbours and thus it is possible to define complex simulations by following the particular characteristics of a specific domain. As examples, an implementations of a visual language for modeling petri nets


Kino-vs-Toto.xml - FreeStyler
File Edit View Transfer Collaborate Allow Plug-ins Help
1 Kino-5
2 Exp. Kino5
3 Toto-3
4 Exp. Toto3
LOS SORTEOS SE REALIZAN TODOS LOS DIAS
kino 5
Loteria
RESET

Kino 5

Numeros del 1 al 30
Se extraen 7 numeros
Se gana el premio mayor acertando a 5 numeros
Tambien hay premios para 4 y 3 aciertos

* Un numero adicional equivale a jugar 6 apuestas por $1.500
* Dos numeros adicionales equivale a jugar 21 apuestas por $5.250
* Tres numeros adicionales equivale a jugar 56 apuestas por $14.000
* Cuatro numeros adicionales equivale a jugar 126 apuestas por $31.500
* Cinco numeros adicionales equivale a jugar 252 apuestas por $63.000

Analisis Teorico:

5: $\frac{\cancel{7}}{\underset{6}{\cancel{30}}}\cdot\frac{6}{29}\cdot\frac{\cancel{5}^{1}}{\underset{7}{\cancel{28}}}\cdot\frac{\cancel{4}}{\underset{9}{\cancel{27}}}\cdot\frac{3}{26}=\frac{1}{6786}\approx 0{,}014\,\%$

Figure 5.4: FreeStyler application showing a combination of nodes and hand-written text.

and another one for robots escaping a maze have been implemented using this framework.

5.5.2 FreeHand Drawing

One important aspect inside a CiC is to use the available resources, without imposing any drastic changes. As a classroom is normally equipped with a whiteboard, it is most natural to have an electronic whiteboard in a CiC, which is used in much the same way as a traditional whiteboard. Certainly it is possible to take advantage of the digital nature of the electronic whiteboard, but the first requirement is to be as useful as the existing technology.

FreeStyler provides the possibility of using the pages as if they were a whiteboard, drawing strokes by hand using some freehand input, such as an electronic whiteboard, a tablet PC, a PDA or any other similar, generally pen-based, device. Each stroke is defined by a path as a sequence of points, but also has characteristics such as width and color. Using some simple buttons it is possible to easily control these characteristics at least as easily as it is to use a traditional whiteboard.

Optionally, it is possible to "compress" each stroke after the last point has been defined, which allows not only to minimize the storage space needed for the resulting document, but to obtain straighter lines or other effects as desired. Also, the undo mechanism allows to erase the last drawn stroke in just one action, as opposed to grabbing (in both the traditional and the electronic whiteboard) and carefully erasing the last stroke which may be on top of previously drawn strokes.

5.5.3 Coupling

Using a MatchMaker server (Tewissen et al., 2000a; Jansen, 2003), it is possible to share one FreeStyler page across several applications. When a page is coupled, a change made by any user in the same MatchMaker session is replicated to the other applications, so all connected users can see the changes almost instantly. This is an important resource for collaboration inside a classroom, by sharing all layers contained in the coupled page, including freehand drawing layers and JGraph based modeling layers.

Although it is also possible to enable collaboration inside a classroom by physically sharing the same device, that approach does not scale well when considering groups of more than two or three students. In many cases, a mix of these approaches is used, having more than one student use one device running FreeStyler that is in turn coupled with other applications.

An additional element to consider is that the MatchMaker server can store the interactions that take place in each session, allowing automated analysis either in real-time or afterwards. In CiCv2, each created session is logged in full detail and the information is available to the teacher from the time it is generated.

5.6 Plug-in Mechanisms in FreeStyler and CiCv2

Within CiCv2, each participant can have her own instance of a Classroom Module to access shared documents and perform local annotations, construct models and other related work. Annotating and modeling will take place using a tool called FreeStyler (Hoppe and Gaßner, 2002), which stores its documents in XML format. It has been integrated into the CiCv2 as the default tool to handle documents. FreeStyler previously had capability to include only a predefined set of visual languages with which the students could work, so it was necessary to generalize its use by adding the possibility to include special plug-ins. This way it is possible to use the tool for any learning domain and evaluate the proposed Best Practices in several differing environments.

The plug-in architecture used is described in Pinkwart (2003, 2005). It defines reference frames that define the visual language as well as its behavior. Since a primary goal for the existing reference frames is modelling of diverse real-world and theoretical constructs, the behavior is essential, leading to modifications of the state according to users' actions, rules regarding the specific domain and the current state of the model.

It is necessary to have access to the information as it is being produced on each separate Classroom Module in order to achieve useful results within the context of the CiCv2. In particular, the interesting data in a face-to-face classroom session is the one being manipulated in the very moment a query is performed or just slightly before. Otherwise there would be almost no difference between synchronous or asynchronous sessions. The data that is handled within each Classroom Module has to be made available to one central location where it is gathered, possibly taking advantage of the distributed nature to do parallel processing whenever possible. In this scenario it would be natural to manage the data internally in the application, where it is already in use and available. On the other hand, the documents that are not being edited at the time are stored either locally by each module or centrally on the Document Manager and contain equally important information, as well as do other information sources, detailed in the next section.

The queries are oriented to be a tool the teacher uses within and outside the classroom, and as such the kind of requested information will probably be different in both situations. In the first case, it is desirable to have fast and less detailed feedback, e.g. to act according to the identified situation or to suggest who needs help. The teacher probably can spend more time interpreting results and refining queries until she gets the desired information when she is outside of a session. Sometimes the same type of information can be required, but with a different detail level, for which a refining of already existing queries might be useful.

5.7 Logging

Logging is an important aspect in CiCv2, since it sums up to the information available to improve the learning experience. Several levels of logging can be identified inside the CiCFreeStyler, described below. The logfiles that are accumulated in each session are transferred to the repository prior to exiting the application. This way, all of the data is kept in the centralized repository, accessible for analysis at any time.

Logfiles are one of the central information resources for a teacher to evaluate and monitor the classroom sessions, both during the sessions and afterwards. All of the activities of students and

```
<meta><SessionID>cc52n-31_5_2005</SessionID>
<UserID>jgarrido</UserID>
</meta>
<LogEntry time="17:31:33"><message>OpenFSDoc</message></LogEntry>
<LogEntry><message>OpenFSDoc: File Kino-vs-Toto.xml loaded.</message></LogEntry>
<LogEntry time="17:32:13"><message>JoinSession</message></LogEntry>
<LogEntry><message>JoinSession: session_vherskov_0</message></LogEntry>
<LogEntry time="17:35:13"><message>QuitSessionAction</message></LogEntry>
```

Figure 5.5: Example of User Actions Log

teacher leave a trail in the logfiles. The details are shown in the next sessions.

5.7.1 User actions

In both applications, for students and teachers, each action is logged locally. The action defines what exactly is logged, and what the result of the executed action is. Figure 5.5 presents a simplified example of the data written into this type of log.

5.7.2 Standard and Error output

The application and any active Plug-in has access to Java standard output `java.lang.System.out` and standard error `java.lang.System.err` streams. To have access to this valuable information, each statement is transformed into an XML fragment that is printed using the normal output, and an additional copy is kept locally in a unique file identified by the session name and log type.

```
<?xml version="1.0" encoding="UTF-8"?>
<SyncActions>

<SyncAction action="objectCreated" label="//53/57/95" number="0"
    objectType="class info.collide.draw.Stroke" time="1117574866545"
    typeOfAction="not set" user="ycovacev">
  <Stroke created="1117574863402" modified="1117574863402">
    <Color blue="0" green="0" red="0"/>
    <Thickness>2.0</Thickness>
    <Point x="336" y="325"/>
  </Stroke>
</SyncAction>

<SyncAction action="actionExecuted" label="//53/57/95" number="0"
    objectType="class info.collide.xml.helpers.Point" time="1117574866614"
    typeOfAction="addPoint" user="ycovacev">
  <Point x="335" y="329"/>
</SyncAction>
```

Figure 5.6: Example of MatchMaker Log

5.7.3 MatchMaker logs

Using the Replay mechanism provided in MatchMaker (Jansen, 2003), a complete history of each collaborative work performed within the CiCv2 sessions is available during and after the session. This history contains all relevant actions and changes done on pages that are shared using MatchMaker collaboration. Figure 5.6 shows an example of the format in which the log is kept.

Chapter 6

Query Model and Implementation

A methodology to model existing best practices has been defined in order to prove the hypothesis stated in 1.2. This methodology follows the assumptions defined in 3.3.5. This will allow a teacher to be helped in evaluating the current situation in the classroom by providing objective information that can be interpreted accordingly, in an efficient way.

A new version of a Computer-integrated Classroom environment, CiCv2 5, has been developed to prove this concept. This CiCv2 includes a generic Querying system, adapted to fit into the FreeStyler application, the main application for managing documents inside the CiCv2. This querying system is used by the teacher to retrieve information in a predefined way, as part of the courseware to be used in each class. The teacher can then adapt the currently used practices in order to find the best way for the students to learn. The queries corresponding to the practices can then be modified by the teacher or courseware author to validate the improved practices.

Several best practices can be implemented and later used in other contexts using CiCv2 along with the querying system, in order to validate the system and the best practices. To do so, the sessions using the predefined queries will be evaluated, and the results contrasted with a careful analysis of the classroom sessions.

6.1 Relevant Information

As already implicitly stated, the information sources from which the queries will draw upon are various and of diverse nature. Information can be stored directly, e.g. through the user's modeling and note taking as well as indirectly, such as metadata that the application adds within the documents. Additionally, the CiCv2 system keeps track of other information, like access to files or interactions among participants. The system can access information from various sources, performing specific queries on distributed data and aggregating the parts to form a unique result as a new XML document, or a document fragment which may then be incorporated into an existing XML document. The sources to be used are the following ones:

- FreeStyler files: written in XML, the FreeStyler (Hoppe and Gaßner, 2002) documents represent the persistent serialized version of the application data. The explicit information of all users is to be found in these documents.

  These documents will probably be the most used information source, since they contain the direct interactions of the students during the sessions. It will be possible to determine the progress of all students in some task, analyze their work and obtain other useful information.

- Metadata: the system will keep metadata information that is not explicitly created by the users. In the case of FreeStyler, the meta-information will be managed internally. However, in order to allow the use of external tools whose file format is unknown, it is preferable to maintain the metadata in a separate file.

  It is possible to identify if students are exchanging documents as part of the collaborative work using this information. It can also be determined whether the documents provided by the teachers along with the course material are really being used as a base for the personal or collaborative work, etc.

- CiCv2 Logfiles: the Computer integrated Classroom (CiCv2) keeps information in XML-based logfiles, which can be accessed to extract useful data like session duration, participation, transfer of documents and other events as well as their chronological interrelation.

From this information it can easily be learned if students are interacting, which groups are being most active or how much information is exchanged among students and among groups.

- MatchMaker logs: during the classroom sessions, it is possible for the applications to interact using a server called MatchMaker (Jansen et al., 2001; Jansen, 2003). This application enables the underlying model of FreeStyler documents to be shared in real-time among several participants, enabling users to see current modifications as they occur. Unlike other such tools that only share the visualization, in this case each application maintains a copy of the model which can still be used when the session has finished. This application generates its own log of events in several XML files which can then be used to obtain valuable information in this system.

  It is possible to analyse the interactions into very specific details using these logs (Mühlenbrock, 2001). However, in this work the usage of this information will be limited to a broader scope, including what interactions do exist, starting and ending times, identification of the groups in which students are interacting and other information following that line.

- External information: some queries may get information from outside of the system, be it external XML files or even information that is typed in as a parameter for that particular query.

  When a teacher is changing the activities inside a classroom session, this may be obvious in some cases, for example when the teacher sends the students some files to use in an assignment. In other cases, the change of activities may not be easily detected using automated tools, and the teacher can signal these changes that will be available in the system as information coming from an external source.

The modelling that is possible in a distributed environment such as the CiCv2 resemble a purpose-based user modelling (Niu et al., 2003; McCalla, 2004), given the availability of the presented information and the type of queries presented in 4.3 on page 58. A purpose-based user modelling allows the decentralized modelling, without necessarily adhering to a common representation scheme (Niu et al., 2003). The CiCv2 is a rather centralized tool, in that the applications used are not expected to be modified and thus the abstract models are centrally managed. However, its nature of allowing the inclusion of previously unknown domains means that the user modelling

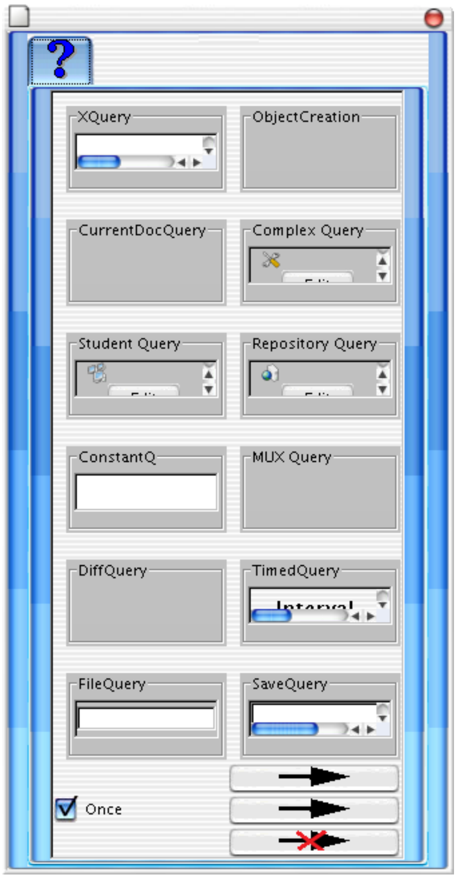


Figure 6.1: Query Palette

according to a specific domain is necessarily decentralized. Also, when considering the CiC in a wider scope, the concept of managing the information about learners as an ecological approach (McCalla, 2004, c.f.) makes sense.

6.2 Query Reference Frame

The queries developed for this thesis are available to the teacher through a visual representation for each element, grouped together in a Query Palette (see figure 6.1). In the Query Palette are the basic queries, building blocks that can be combined to create new queries. It is possible for example to obtain the difference between two documents by using one FileQuery for obtaining each file and connecting them to a DiffQuery. In order to hide the complexity, this three queries can be encapsulated inside a ComplexQuery, so the end user sees only one simple query that performs what is expected.

The Query Palette was developed specifically for this thesis and is the engine that provides the teacher with the information needed to define, apply and validate Best Practices. The visual presentation of both the queries and their results use the interface provided by the JGraph framework (cf. 5.5.1). Thus, the query engine could be separated from the JGraph and CiCv2 framework, provided the Java interface is used or a new graphical interface is used instead.

The implementation of the query functionality is separated from the visualization of such query. Following the Model-View-Controller pattern as described in 5.5.1, the query functionality implementation is used as the model part of the pattern. For each model, a specific graphical representation is created, which allows to manipulate any potential variable stored by the *Query* that should be user-defined. The output of one query is connected as input to another query by means of Edges in a JGraph (cf. 5.5.1), allowing the encapsulation of these sequences of queries into a single unit, a Composed Query.

6.2.1 Data Types

Although the data used in the system is represented in XML, there is no absolute compatibility among queries. Some queries do expect its input in a predefined format, and any input that is not in this precise format will generate either an error message or an unpredictable result.

A data type definition associated to each query, along with creation-time type checking could prevent these errors, opening the possibility for others to manipulate queries without the need to know the internal XML representation of each intermediate result. Definition of several data types is recommended, not all strict, in such a way that it is possible to classify one query into several categories. Using this scheme, it is possible to maximize the reuse of queries without adding unnecessary incompatibilities to queries.

However, each time a new item is added to the possible classification, it is necessary to redefine the classification of each potentially affected query.

Currently, the power of redefining queries at any time without the need to fit a strict classification is more valuable than the possibility of persons not familiar with XML to modify the queries. Once the system matures and gets used in everyday work inside educational facilities, it is expected that the benefits of defining such categories will outweigh the costs, specially when non-technical users start using the system daily.

6.2.2 Visual Representation

The visual representation of the queries is specially important because this is the only way in which the normal users of the system interact with the queries. All features of the queries have to be available through the graphical user interface, and the representation should be clear to define what actions will be performed by the queries.

The decision of using a visual language over a text-based query language has two reasons. On one hand, some studies suggest that the usage of visual languages to perform querying is easier for the users (Rontu, 2004; Catarci and Santucci, 1995). However, the main point is that teachers are not expected to modify the queries during classroom sessions, except for adjusting some predefined parameters. That is, the queries are prepared in advance, currently by the teacher with the assistance of an expert who is familiar with the system. In the end, the teacher will only see a query element with an “execute” button to trigger the query, and at most a few parameters that can be configured.

Assuming the availability of a really simple interface that encapsulates all of the complexity a query might entail, the visual representation has one big advantage: the teacher does not need to switch from one tool to another, or even from a visual interface to a textual interface. This helps to keep the focus by avoiding the need to adjust to a particular interface, which costs time and concentration.

Considering the flow of data between the queries, it seems intuitive to represent them as elements combined by arrows where the direction of the arrows represents the flow of information. However, there are two aspects that need to be analyzed: control flow in the execution of the queries and the context changes that take place in distributed queries.

The context changes are represented by encapsulating the queries that take place in the different context in a separated visual panel. The panel is part of the query that makes the context change happen. This allows the visual separation of the queries that are executed in various contexts, which avoids possible confusions of where each query gets executed. Some of the queries even can be executed several times in different contexts, with the results summarized by the parent query. Additionally, the separation of queries executed in different contexts simplifies the rest of the model, because the focus is on the flow of information assuming that all of the queries connected within one panel share the same context. Otherwise it would have been necessary to display the passing of context arguments from each query to its sub-queries.

The visualization of queries has some common characteristics, shared by all of the implemented queries. This includes the border of the graphical widget with a brief name for each query, a button for executing a particular query and an *internal panel* in which the peculiarities of each query are presented to the user.

In the case of complex queries, the name displayed in the border of the query visualization can be edited, in order to give each complex query a unique name that identifies the action to be carried out when executing it. By default, the *internal panel* of a complex query includes an icon that identifies each of the possible complex queries, an image and an optional text description. The icon is defined for the queries that simply group sub-queries together and also for the ones that change the context for the sub-queries *Student Query* and *Repository Query* (A.3.9). This way, the user is always aware of the context in which the sub-queries will be executed.

The image inside the *internal panel* for the complex queries is provided automatically by a miniature visualization of the JGraph that defines the composition of the sub-queries, unless a different image is provided. The image, along with the descriptive text should allow a user to identify the desired query in a glimpse, taking advantage of the visual nature of the query manipulation.

The button for executing each query is displayed as long as the query is the *last query* of a composition. When the query is connected so that its output is directed to another query, the button disappears since it is expected that only the last query will be used to initiate the execution. This avoids the cluttering of unnecessary buttons on the screen, making it clear for the user how she is expected to initiate the execution of a particular query. However, for debugging purposes, it is always possible to execute a query that is not the last, ignoring the outgoing connections by returning the result as a standard *QueryResult* node. This is done by using the pop-up menu of the corresponding query node in the JGraph.

6.3 Abstract Query Model

Before starting with the implementation of the queries in the CiCv2 scenario, it is necessary to take a look at the abstract model that has been implemented. As already discussed in 4.5, the information available to the queries is in XML format, and it makes sense to define the output to be XML as well. This way, the output from one query can be used as input to another one, favouring reuse of available material.

The queries will need access to certain resources, such as files stored on some filesystem or in a repository, access to the network to trigger queries in other locations, as well as access to all information sources described in 6.1. Considering these requirements, following concepts are defined: *Query*, *Query Resource Provider*, *Query Connection*, *Query Connection Link* and *SubQuery*.

6.3.1 Query

A Query is an object that optionally takes some input in XML format and has access to context-dependent information. This context-dependent information is available to the query in two ways: one part is concerned with the input and output of a query, as a means of communication with other queries that it is connected to. A query can be connected to other queries in a two-directional way as seen in figure 6.2. The other part consists of access to several context-dependent resources, as detailed in 6.4.1.

Atomic queries are those which are not composed of other queries. They take as input one XML stream, they perform some modifications to it, acting like a filter in the Unix sense, and finally they deliver another XML stream as the output. However, it can be the case an atomic query has more than one input and/or more than one output stream. In particular, it is possible to define multiplexer and demultiplexer queries so that all other queries use these instead of having multiple inputs and outputs. Having done this, almost all queries could be described in terms of a language like XSLT (Clark, 1999) or XQuery (Boag et al., 2001). Most of the processing is done using the XQuery language in the queries described in this thesis.

An atomic query can be as complex as wanted, as long as it does not involve other queries. Should it be possible, it is be preferable to separate a query into several sub-queries, since it allows more possibilities for reusing parts of that query in other ones. All queries are subclasses of `CiCQuery`, as is the case of `ComplexQuery` and `ConstantQuery` shown in figure A.3 on page 158, and have a visual counterpart (`ComplexQNode` and `ConstantQNode, respectively`).

6.3.2 Query Resource Provider

The context in which a query is executed, as already discussed, is relevant to the results of that execution. That is, a query that is executed in one context can and is expected to deliver different results than the same query executed in another context. This difference stems from the context-dependent data available from the outside for each query.

Any query may need information from the outside, i.e. additional data that is not provided as input coming from another query. For example, the content of a document stored in the repository, a list of the currently signed-on students or the name of the current session. All of this information is accessed through a *Query Resource Provider*.

6.3.3 Query Connection

A Query as such is only concerned about providing a result when given a specific input, output and context-dependent resources. The way in which a query is connected to others in order to form the input and output connection is handled by the *Query Connection*. The connection contains the information of the neighbours the *Query* has. This information includes both a link to a next or previous *SubQuery* and a reference to the connecting entity, in the form of a *Query Connection Link*.

The order of the *Queries* is an important part of a *Query Connection*, since it determines in which way the information will flow during the execution. That is, the order determines which *Query* will give its output to the “next” *Query*. However, a *Query Connection* represents a two-way relationship: one way is how the data flows, but the flow control also has to be considered.

When considering two connected queries, one of them will be called *first query* and the other will be called *last query*. The *first query* will be executed, and the output will be used as input to the next and last query. But in order for the first query to be executed, the last query has to connect to it and request the output. This way, The *last query* is the one that triggers the execution, initiating the control flow. When having more than two queries, the control flow propagates until it reaches the first query, and from then on the process is reversed as each intermediate query receives its input from the previous query. The data flows in one direction, from the first query (the one without incoming data) to the last one (the query without outgoing data). When executing this query sequence, called *query composition* in the context of this thesis, the query to be executed is the last one.

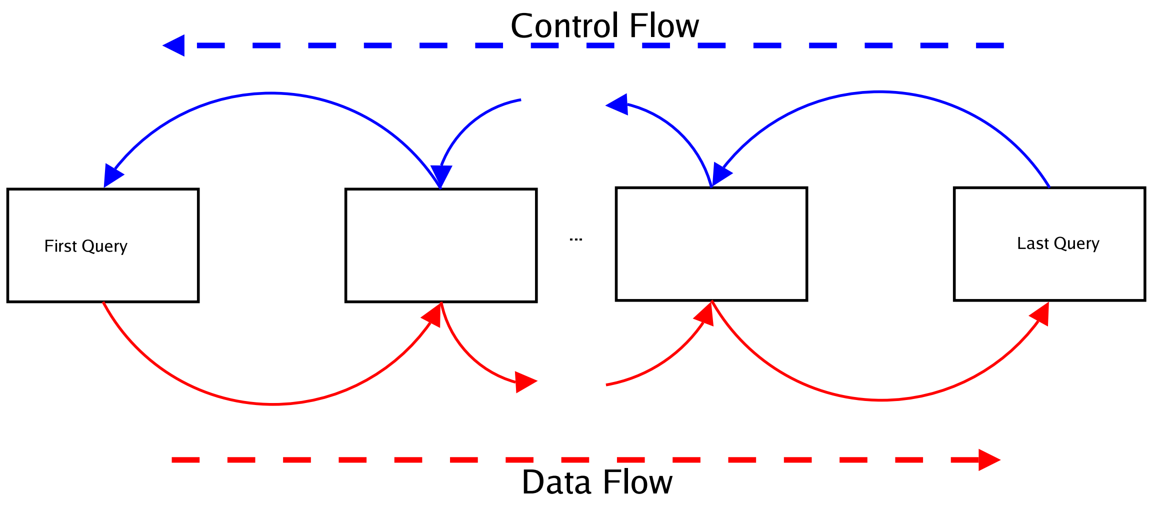


Figure 6.2: Control (red, from left to right) and Data (blue, from right to left) flow

6.3.4 Query Connection Link

The element that links two queries together may have some properties that depend on the implementation. It can be assumed that the link will provide a label, and this label enables a *Query* to keep one identifier for each of its inputs. In some cases this identifier may be ignored, or just used to order several inputs in some repeatable way. But other queries (see for example the Multiplexion Query in 6.5.6) may use this label as if it were part of the input, so the output will depend on its value.

6.3.5 SubQuery

A SubQuery is an ordered pair of a *Query* and a set containing all of its *Query Connection* elements. In other words, a *SubQuery* joins the basic implementation of a *Query* with the knowledge of its neighbours that is contained in the *Query Connection* elements.

It is possible to compose queries, forming a new query as the result, using the Composite Design Pattern as described in (Gamma et al., 1994). This query can also be used in further compositions, since it behaves the same way as any other query. This scheme allows queries to be processed at several locations in parallel, enabling the distribution of queries as mentioned in 5.6. A *SubQuery* is defined for this purpose, and essentially provides a way to execute a query just providing the *Query Resource Provider*

6.3.6 Visual representation of Abstract Model

As mentioned in 6.2, the connections available in the form of Edges in a JGraph (cf. 5.5.1) implement the *Query Connection Link*. Additionally to the standard Edge in a JGraph, the *QueryEdge* can optionally include a label in order to comply with the requirement of the *Query Connection Link*.

The abstract model is implemented in two stages. First, the functionality of the query itself was developed, as described in A.3. This corresponds to the *Query* definition above. In the CiCv2, the *Query Connection*, *Query Connection Link* and *SubQuery* constructs have been implemented using the concept of a *reference frame* (cf. 5), as a visual language for constructing queries.

In a JGraph, the information about neighbouring nodes is available to the View, so the *Query Connection* is implemented by the visual representation of each query, and the *SubQuery* is also implemented by the View, which always maintains a link to the Model. This way, the *SubQuery* requirement for giving access to the query implementation and to manage the connections to neighbour queries is fulfilled.

```
CiCQueryResult execute(QueryConnection conn, QueryResourceProvider prov);
```

Figure 6.3: Signature of the method provided by all subclasses of CiCQuery

6.4 Query Context in CiCv2

According to the definition in 6.3.1, all information a query may access is either available as XML input (cf. 6.3.3) or through requests to a Query Resource Provider (cf. 6.3.2). In the CiCv2 implementation, this maps to the definition of two interfaces: `QueryConnection` (see 6.4.2) and `QueryResourceProvider` (see 6.4.1). Using these interfaces, a query is executed using the `execute` method as defined in figure 6.3. This way, every execution of a query has access to all relevant context through these defined interfaces.

6.4.1 External data: QueryResourceProvider

A Java interface called `QueryResourceProvider` is defined to make the relation of a query to its context as clear as possible. The QueryResourceProvider interface gives access to all information a query needs to execute and that depends on the external context. The resource provider is the path by which a query can interact with its context. A query can be executed in different contexts according to the part of the distributed environment in which the query has to be evaluated. The queries access all context-dependent data through the resource provider, and this way the context can be encapsulated changing the resource provider for the query.

In the CiCv2 architecture, there are 3 possible contexts in which a query can be executed:

- Teacher application
- Student application
- Repository

In each context the query has an associated resource provider. This way the query has access to the current document, log files and archive. Some of these elements do not make sense in all context, in which case the returned result is null. The implementation details for the `QueryResourceProvider` are described in A.2.1 on page 159.

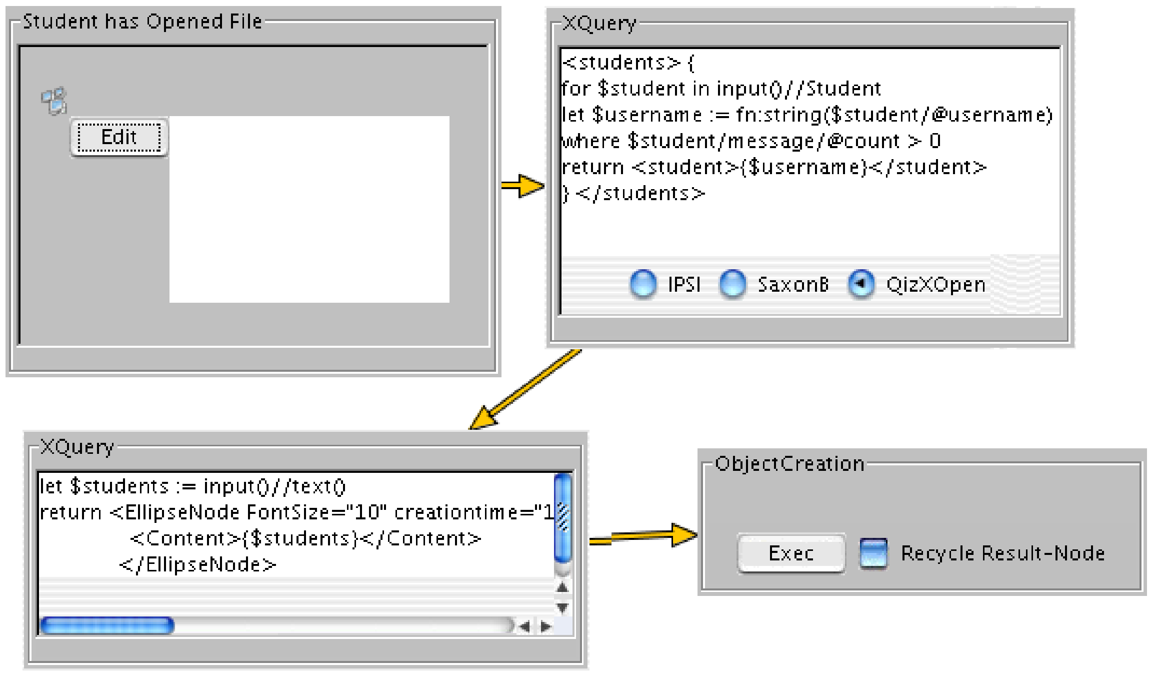
In several monitoring scenarios, the teacher first needs a list of students. This way the teacher can select one student to retrieve specific information, or she may send a query to each student and compare the results. This information is available in two possible ways: the first is directly in the case of the TeacherApplication, where the data is stored in memory. The second case is using the log files, which is also a permanent resource that will still be available even when the session has ended.

Access to documents is available for reading and writing. The teacher has the ability to access any file this way, including files in the repository as well as the files that students have currently open, whether these have been saved or not. Access to logfiles is granted as read-only, since there is no intention to modify the recorded information.

6.4.2 Query Composition: `QueryConnection` and `QueryConnectionLink`

The relationships among queries are defined using Java interfaces named `QueryConnection` and `QueryConnectionLink`. These relationships define the data flow: one query is executed first, sending its output to the next query, which uses that output as its input, and so on. This is a generic approach that can be managed in more than one way. In the CiCv2 implementation, a graphical connection between graphical representations of the queries is used to make the composition, as shown in figure 6.4.

The figure presents a composition of 4 queries. The first query is a custom query called "Student has Opened File", created by combination of basic building blocks. The output of this query is a raw XML output that is processed by the second query ("XQuery"). An XQuery processes the XML input it receives using the script written in XQuery language. In this case, it simplifies the



```
<students> {
for $student in input()//Student
let $username := fn:string($student/@username)
where $student/message/@count > 0
return <student>{$username}</student>
} </students>
```

```
let $students := input()//text()
return <EllipseNode FontSize="10" creationtime="1
        <Content>{$students}</Content>
       </EllipseNode>
```

Figure 6.4: Query Composition: 4 queries composed using visual connections

output leaving only a listing of the relevant username of students that have opened the file defined in the “Student has Opened File” query. As can be seen, the XQuery allows the selection of several XQuery engines called IPSI, SaxonB and QizXOpen (cf. 6.5.8).

The third query is also an XQuery that takes the list of students and generates an XML fragment that will be interpreted by the last query. The fourth and last query is an “Object Creation” query that takes the input and attempts to interpret it as a Node object or a FreeStyler page. When it succeeds, it adds the generated element to the document. As the “Object Creation” query is the last one, it displays the *Exec* button that triggers the whole pipeline of queries.

A connection between two queries, created by using a `QueryConnectionLink`, can optionally contain a label as a means of identification. This identification, in form of a `String`, allows the query that receives the input to distinguish among two or more inputs. In cases where the order of arguments matters, the result depends on being able to correctly identify the parameters.

Some queries do not use any input, in which case there should be no inbound data connection to avoid confusion. However, the existence of inbound data connections is not controlled at this point. See 6.2 for details about the visual control and development of queries.

6.5 Basic building blocks for queries

All queries are built by using the basic building blocks presented in this section. They are queries on their own, but the output is in raw XML format, so they are rarely used by themselves. Rather, a typical query fetches information, transforms it and then presents it to the user visually, for example by adding some element to the page where the query is.

The implementation of these atomic queries is described in section A.3 on page 161. These implementations are independent of the method used to link several sub-queries together using a Query Connection (cf. 6.3.3) implementation. The same Query implementations can be used with different Connection implementations, which allows the queries to be used outside the scope of the CiCv2.

6.5.1 Constant Query

This is the most basic form a query can take. It stores a predefined value that is returned each time the query is executed, with no variation. It is mainly used for allowing the teacher to specify values for parameters.

```
<student username="pedro">
    Student has opened file
</student>
<student username="juan">
    Student has not opened file
</student>
```

Figure 6.5: XML Fragment (not well-formed XML)

```
<students>
    <student username="pedro">
        Student has opened file
    </student>
    <student username="juan">
        Student has opened file
    </student>
<students>
```

Figure 6.6: XML Fragment (well-formed XML)

6.5.2 Query Result

The result of a query is always an XML fragment. This means the result has to be well-formed XML, but not necessarily have a single root element. For example, a listing of several items one after another as seen in figure 6.5 does not qualify as well-formed XML, but the same fragment wrapped in a single root element does, as in figure 6.6.

The XML fragment can be stored either as a string of characters or as a tree-based Document Object Model (DOM), however both forms of storing the result are equivalent. That is, the string form is the serialized DOM. In order to avoid unnecessary computations, the result can be stored in any of both forms, and transformed as necessary to the other form. Most queries will use the DOM form, but when transferring a result from one context into another, the string form is used.

Also, some queries (e.g. the Constant Query or File Query) return their result codified as a string.

It is possible to use any QueryResult as the input to any query as if it were a Constant Query with a fixed content, as a means to support the query creation process. The only difference is that the content is not typed in by the user but generated as output from a previous query.

6.5.3 File Query

This query allows a File to be read. The filename argument can be provided either as a predefined filename typed in by the user or fixed by the creator of the query, or as input from another query. This allows to define the filename to be read dynamically, for example by determining the previously saved file of a student and then reading that file to be processed.

The File Query also gives access to logfiles, which are intended to be used by the people crafting complex queries, but not by the end user (teachers) directly. Access to these logs is granted by way of special file names that cannot be mapped to actual files on the repository.

6.5.4 Current Document Query

This query enables access to the currently opened document. The document is returned as a String element that can be parsed into a Document Object Model as mentioned in 6.5.2. This is one of the most important queries for the system, as it is the only way to have access to the information the students are currently generating or modifying.

6.5.5 Complex Query

A sequence of queries linked together by `QueryConnectionLink`s can become excessively complex. To overcome this problem, and to hide this complexity from the teacher, it is possible to group the whole sequence into one single query called Complex Query. Using this mechanism an arbitrarily complex query, composed of any number of sub-queries, can be encapsulated and it will look as simple as any other basic query. In particular, this complex query can then be used in a composition, replicated and sent to a remote location, and any other operation that can be performed on a simple query. Figure 6.7 shows an example of a Complex Query, where the sub-queries corresponding to the complex query marked as (a) are being shown on a separate window (b) for editing. In normal use, that window is hidden and the user only needs to see the graphical representation of the complex query, which features its particular description and icon, or a miniature representation of the sub-queries when the former have not been provided.

When a Complex Query is executed, the first and last query are identified inside the grouped sub-queries. The input that is connected to the Complex Query has to be connected to the first query, and the control flow needs to continue to the last query, from where it will propagate until it reaches the first query. The result of the last query is then received in the `ComplexQuery` and handed over according to the `QueryConnection`.

6.5.6 Multiplexion Query

The query model allows a query to have multiple inputs, and even to have multiple outputs. Actually, when multiple outputs exist, they are not used at the same time. As seen in 6.5.5, it is the last query that triggers the execution, so only the output connected towards that query is used. It is thus equivalent to create a copy of the query with more than one output and consider both as absolutely independent queries. The advantage is the reuse of a portion that can be used in more than one particular query.

The existence of more than one input to a query is not that straightforward. Some queries do

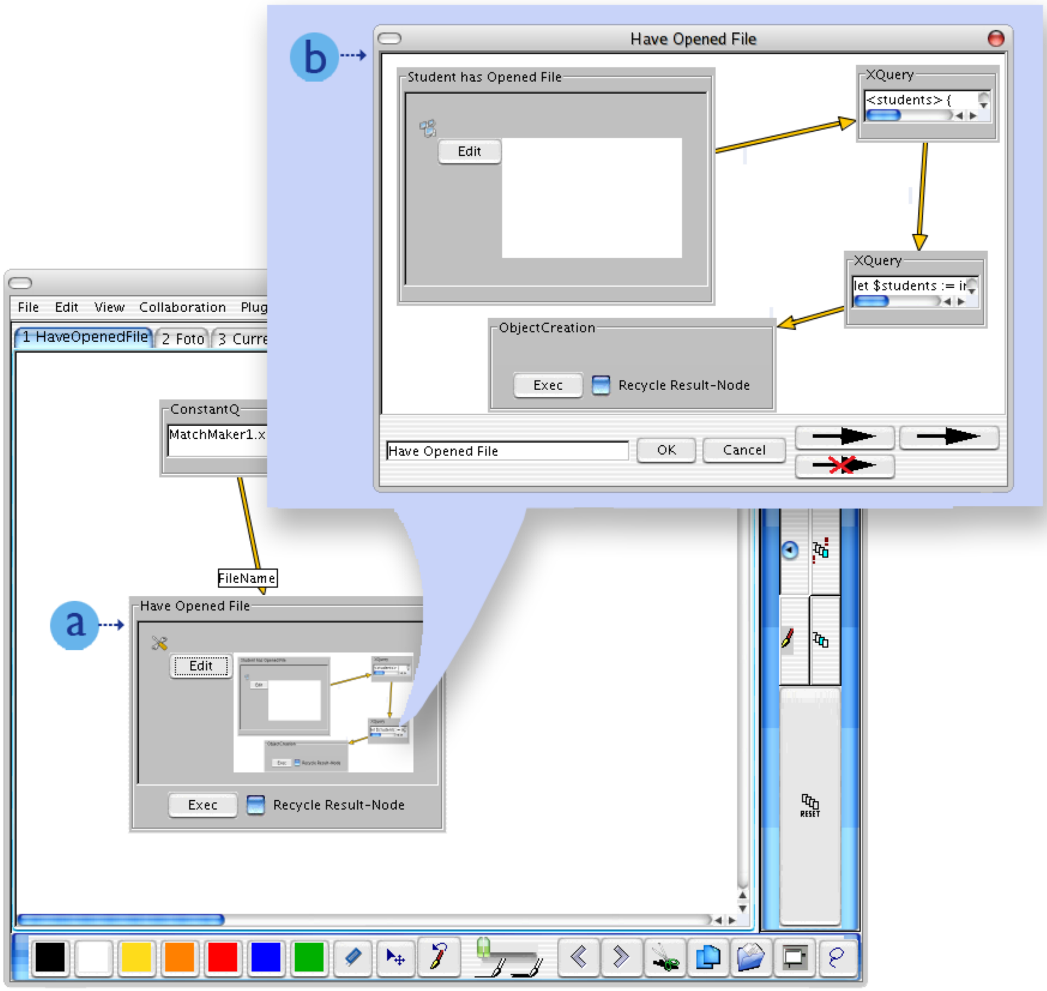


Figure 6.7: Complex Query.

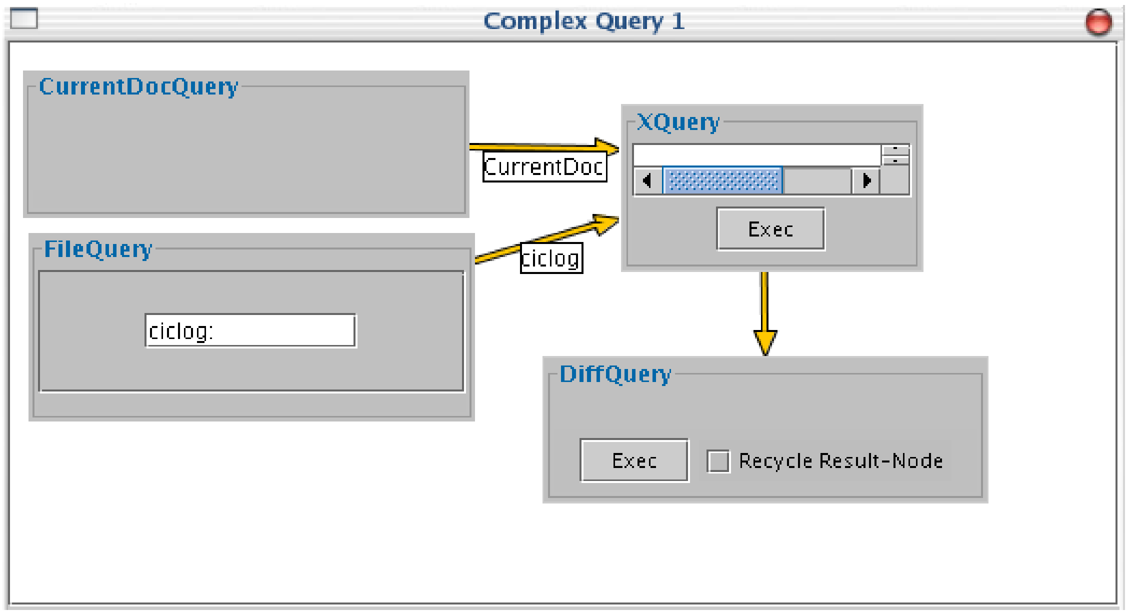


Figure 6.8: Sketch of a query showing the number of changes a student made in the last X minutes to a document

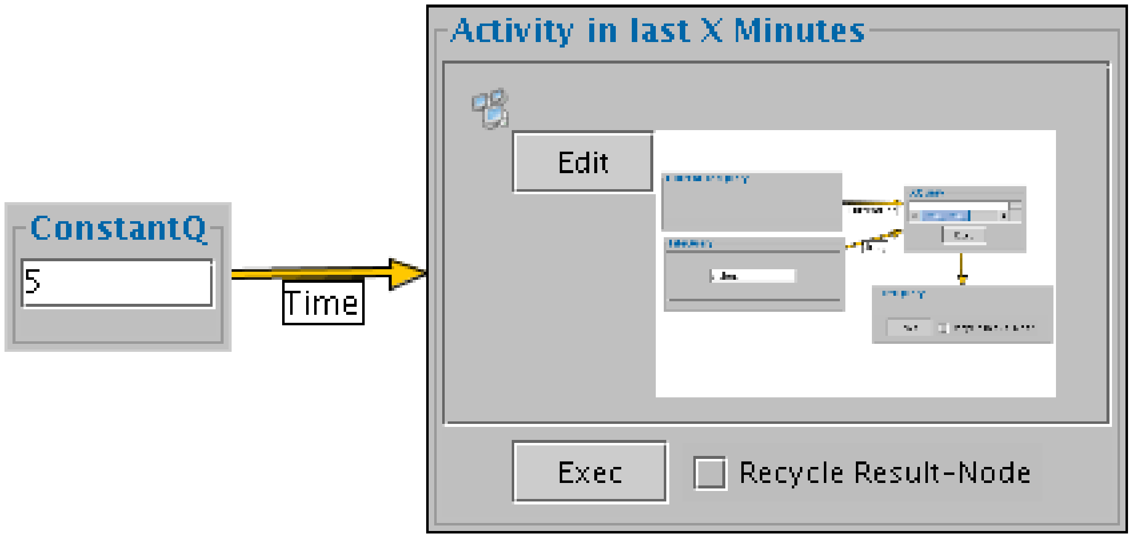


Figure 6.9: Complex query encapsulating query of figure 6.8 and adding a parameter

not make use of any input, such as the Constant Query mentioned previously, and others may use at most one input, such as the File Query. In several cases, queries need information from more than one source. For example, a diff query always needs two inputs. Likewise, a query that returns the number of changes a student made in the last X minutes to a document needs following input:

- Filename of the original document
- Number of minutes to consider (parameter X or Time)
- The current document for that Student

However, not all queries are able to handle several input sources even when it makes sense to have them. The XQuery behaves this way, because of limits and different APIs of the underlying engines. The Multiplexion query is used for solving this problem, by providing multiple input sources encapsulated inside only one. For each input that is connected to a Multiplexion Query, a subtree in the XML output is created with the output of that particular query, and it is identified by a tag containing the label of the connecting QueryEdge (see 6.3.6 for details). Each input source is easily accessible inside the encapsulation by using the label, so no capability is lost by using this multiplexing. It is even possible to use the output of one Multiplexion Query as the input to a Diff query and the latter is able to identify both sources and act as if they were accessible separately.

For the example mentioned above, figure 6.8 shows the sketch of a query that receives the three mentioned inputs. The filename of the original document is extracted from the ciclog, and the complexity of the query is encapsulated into a complex query so that the user sees only what is shown in figure 6.9. The only parameter the user should be able to see and modify is the Time, which is thus kept outside of the encapsulation. When we use the logic described in 6.5.5 to the query, the first query in figure 6.8 could be either the CurrentDocQuery or the FileQuery. So the output of the ConstantQuery containing the Time parameter would be sent to one of these. What is really intended is that the three parameters deliver their output to the XQuery. The solution is to use a Mulitplexion Query that will by default act as the first query and thus receive all of the three inputs, group them together and deliver them to the next query.

6.5.7 Diff Query

One of the objectives of monitoring and querying is to deliver the least necessary amount of information to the user. The user, in this case the teacher, is already overwhelmed with information and the tools should provide the relevant information. One good way to discriminate between useful and redundant information is to look at differences between two states. The Diff Query provides a way for the teacher to know what already was there and what is new when considering two versions of the same file. It is an essential part of several useful queries.

The context where this is applied is mostly the history of a document. However, it is necessary to consider all copies of a document, and in the CiCv2 environment the exchange of documents plays an important role of everyday interaction. For example, the teacher will deliver a document to all students for them to solve the proposed problems. In order to see what has been done by the students, the teacher is interested in seeing not the whole document but the work the students have

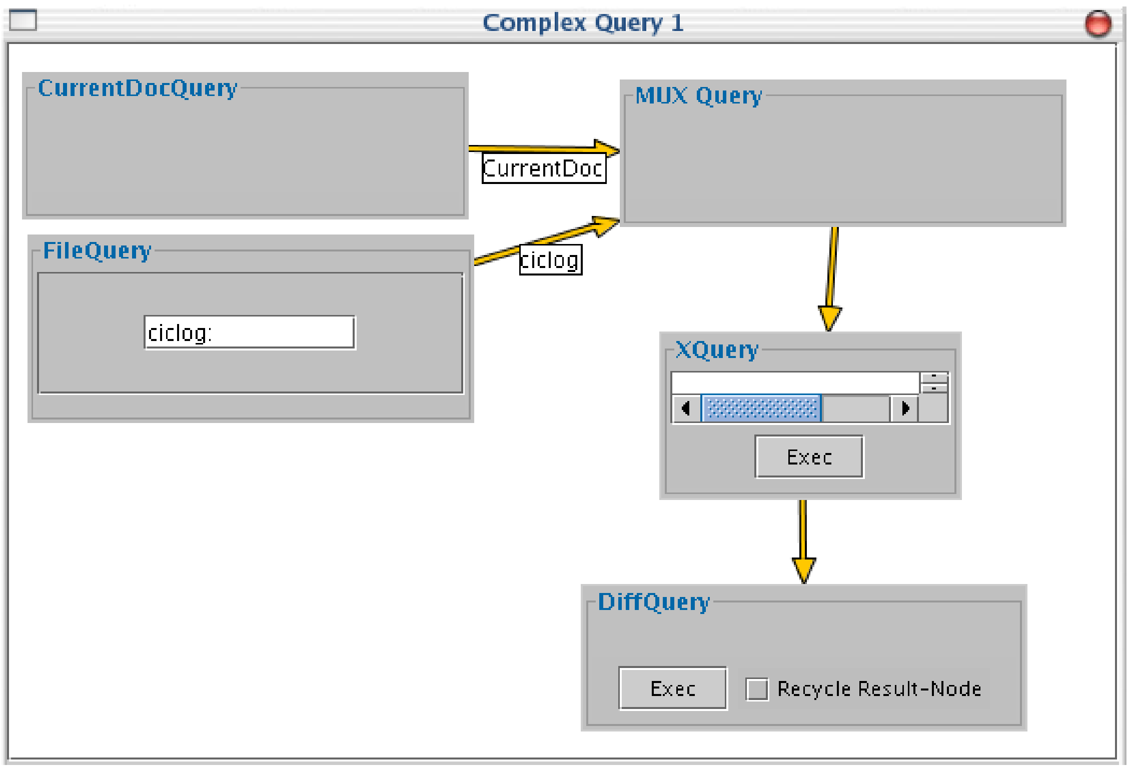


Figure 6.10: Using MUXQuery as first query

done, either by looking at the details of one document or by viewing the summarized information for the group.

6.5.8 XQuery

The main document type within CiCv2 is in XML format (Bray et al., 2004), and one of the available tools to perform queries, like XSLT (Clark, 1999) or XQuery (Boag et al., 2001) will be used whenever possible. The results of the queries should include XPath (Clark and DeRose, 1999) expressions in order to address specific parts of documents.

XQuery allows the arbitrary manipulation of the input, making it a powerful tool to provide to the developers of queries. XSLT is also a powerful tool, but the nature of the queries make the XQuery language more straightforward for a query environment. Teachers will not need to use these languages directly, but use a carefully drafted XQuery as a new building block.

6.5.9 Save Query

In several cases the teacher will want the result of a query to be saved locally or to the repository. The Save Query was created for this purpose, making it possible to use the saved information at any later time, either by another query or by opening it as a document. This allows a teacher to create new documents automatically based on the documents the students have opened, for example to gather all of the solutions to a single problem in one document for later assessment, or saving the current document of every student for later detailed analysis.

6.5.10 Object Creation Query

The *Object Creation Query* allows the content to be presented to the user. This implementation is closely related to the environment in which the queries will be used, in this case the CiCv2. If the query framework is to be applied in another classroom environment, it would be necessary to rewrite or adapt this query. The *Object Creation Query* interprets its input as a fragment of a FreeStyler document. This fragment can contain single elements or even several pages of data. The data is inserted into the document according to its type. This way, the teacher receives output in a familiar format and we avoid the use of a separate interaction environment for the queries.

A parameter defines whether the created object will replace a previous result or generate a new object without replacing any old one. When the teacher does not want to have several old results keeping around, she can define the query to replace the old outcome every time it is executed again. This way we eliminate the noise of providing the teacher with too much information and leave just the new information available.

6.5.11 Timed Query

For monitoring the student activity it is useful to have a history on which to draw from to compare the current state. To construct the history of any particular indicator or data result, the teacher can attach a Timed Query as the last query to any existing one. This triggers the query at periodical intervals or at a specified moment.

By combining this query with the Save Query, the teacher keeps a record that will be available in the repository for immediate or future use. The same way, a teacher might want to monitor only the current state of an indicator, so she would use an Object Creation Query configured to replace the previous result.

Chapter 7

Query Usage Scenarios

In order to validate or refute the hypothesis stated in 1.2, several case studies were carried out. The case studies were conducted at the Universidad de Chile, with students of an undergraduate course called “Computación para el Trabajo Grupal” (Group-Work Computing). The subjects were engineering undergraduate students (Ingeniería Civil en Computación) as well as some graduate students.

Before starting the described sessions, a preliminary session was conducted to introduce necessary concepts and basic knowledge regarding FreeStyler (cf. 5.5) and CiCv2 (cf. 5). The classroom sessions were performed in a regular classroom where every student (about 10 participants in each session) had a notebook with wireless network access and the teacher had a projector to display the material from a desktop computer. The teacher was running a version of the Teacher Application, and each student executed the Student Application. Each session was recorded using a video camera. The logfiles of teacher, students and repository applications were also stored. Additionally, using a query described in the next section, a periodical snapshot of the state in the system was taken, including all student’s currently opened documents.

7.1 Implementation of Common Queries

Some of the queries used in the classroom case studies were used in several sessions. These queries are not domain-specific, so they could be applied in several settings with other subject domains.

7.1.1 Classroom Snapshot

This query was used for the purpose of providing, as its name suggests, a snapshot of the classroom situation at an instant of time. A Student Query is executed, so that every student application generates a serialized version of the current document in XML format, he saves it locally and afterwards, he sends it to the repository. For implementation details, please refer to C.1 on page C.1.

The name of the file into which the document is saved has the form of `currentDoc-YYYY-MM-DD-HH-MM-SS.x` where `YYYY`, `MM`, `DD`, `HH`, `MM` and `SS` are filled with the respective values of the current year, month, day, hour of day, minute and second. The query is executed periodically with a resolution of seconds, resulting in each file having a different name and thus without any data being overwritten. As the logfiles contain a timestamp for each log entry, it is possible to simulate any particular instant of the classroom session by ignoring any entry included after the date reflected on the filename. This way it is possible to perform queries as if they were made in that particular instant.

The time query frequency will define the accuracy or resolution of the simulation. The data gathered during the three case studies described further on in this chapter was used to determine a reasonable frequency. 30 to 90 seconds is a frequency that allows an accurate description of the relevant information, without accumulating an excessive number of redundant data clogging the repository.

7.1.2 List Students

Even when the teacher has a list of connected students directly available using the CiCv2 application, it makes sense to provide a query that lists the active students, for several reasons:

1. The query is available in the same format as the other queries, making it more natural for the teacher to use it. It is not necessary to use a separate window in order to get the information.
2. The intermediate result may be used by other queries for further processing.
3. It is possible that the result of this query is different from the list of students registered at the teacher application. When a Student Application does not respond at all, or it fails to respond within a determined interval, it will not be included in the list returned by this query. Only currently reachable and responding applications will appear in the returned list.

Implementation details for this query are available in C.2 on page 186.

7.1.3 Current Page from Student

The teacher can retrieve the currently opened page from a predefined student.

A Constant Query is used to identify the student from whose application the page will be requested. The Constant Query is connected with the Current Page query through a QueryEdge labeled “Student”.

The teacher may use this query to view the work a student is doing by directly reviewing the page he or she is working on. This is a rather slow process, moreover when the teacher would access the current page for every student. However, the teacher will generally use other queries to gather information and, based on this information, she may choose to take a closer look at the work of certain students.

Once the page has been retrieved, it is appended to the document opened in the Teacher Application. The teacher may then use the new page like she would use any other one, which allows her to modify it without the changes propagating to the student. The new page is a local copy, and it is possible to use it in any activity such as displaying it for the whole class and taking it as an example, etc. Implementation details for this query are available in C.3 on page 188.

7.1.4 Specified Page from Student

Like the previous query, this query retrieves a page from a specified student, but in this case it is not the "current page". The query has a second input, labeled "PageNumber", in which the teacher writes the number of the wanted page. Implementation details for this query are available in C.4 on page 190.

7.1.5 Diff Count

This query computes the differences found between a specified file in the repository and the current document for each student. Then the number of changes between both documents are counted, separated in deletions, updates and inserts. Implementation details are described in C.5 on page 192.

7.1.6 Opened File

The "Opened File" query shows a list of the students that have opened a specified file. The teacher can use the tools available in the CiCv2 to deliver assignments to the students. When the students receive the file, an icon appears in their Inbox indicating the availability of the document. The students can then click on the icon or open the file using the menu bar, and this triggers an entry in the student log. Using a StudentQuery, and indicating the name of the file the students should have opened, the teacher receives the list of students that have effectively opened the file.

Implementation details are described in C.6 on page 196.

The same information can be used to create a query that shows the complement: all the students that have not yet opened the document. It is also possible to show all of the information, e.g. in a table format, in which each student is listed along with a symbol or word indicating whether he has opened the document or not.

7.1.7 Student Activity Indicator

This query presents the activities performed by each student during the last couple of minutes. The considered time interval can be configured, and in these case studies it was set at 5 minutes. As a result, this query generates a table containing one row for each active student, and the number of creation or modification of nodes and strokes inside the current document, as well as number of actions (through the use of the menu bar and buttons) performed. Implementation details and sample output are available in C.7 on page 198.

7.1.8 MatchMaker Collaborations list

The teacher gets a list of all active MatchMaker collaborations within the current CiCv2 classroom session using this query. For details about the implementation and a sample output of this query, see C.8 on page 203.

7.1.9 MatchMaker Activity Indicator

This query creates a table that shows the students that are actively participating in the session by providing the name of a specific session. Also, a summary of the total number of interventions is available, as well as a breakdown by type in `ActionExecuted`, `ObjectCreated` and

`ObjectChanged`. For implementation details, see C.9 on page 205.

7.1.10 Model Complexity

For each active student, this query generates a row in a table, detailing for each page the title, number of nodes, number of edges and number of strokes. Details about the implementation and a sample output are available in C.10 on page 212.

7.2 Scenario Case Studies

The next three sections describe the specific case studies that involve three different scenarios in which the queries were applied during classroom sessions. The queries used in each of these sessions includes the ones described in 7.1 as well as queries specific to the session or domain.

As mentioned in the beginning of this chapter, the case studies were conducted at the Universidad de Chile, with students of an undergraduate course called "Computación para el Trabajo Grupal" (Group-Work Computing). We will see that in the three scenarios we are able to identify valuable information using the querying system. This information was easily available to the teacher, and obtaining it by other means would be time consuming and/or expensive in human resources, making it inviable.

The first scenario, Java Programming, is useful to show that using the queries the teacher can monitor the group progress as well as identifying particular students that are having problems with specific tasks. It is relatively easy to construct queries that give a good approximation on whether students are using the right path or forgetting some important issues.

The second scenario shows a domain that is specially useful in modeling using tools like CiCv2. In the real world it is effectively possible to simulate stochastic processes in a laboratory. However, to achieve the number of repetitions necessary to get results that are accurate enough to compare them to the theoretical outcome, the time is generally too short. Using the simulation every student can repeat the actions virtually hundreds of thousand times in a very short time. It can be seen that, by using the querying system, the teacher can be aware of how many students are making well-known errors in their modeling and take actions as needed. In addition, this scenario uses group work in which each member shares a common workspace with the rest of the group. Thus, the teacher has information not only of each individual student but also of each group and even the interactions within that group.

In the last scenario the students use System Dynamics to model the interaction of a relatively complex ecology. As the model can be more complex and, above all, less predictable than in the previous scenarios, the challenge in this case is to provide the teacher with good quality information without relying on the specific details of the System Dynamics domain. Most of the information presented to the teacher in this scenario is not specific to the System Dynamics domain, however it proved to be useful, and relatively cumbersome to get to by other means if we did not have the querying system at hand.

7.3 Java Programming

This is the the first case study of the series, whose the objective is to learn the inner workings of the MatchMaker client and server. For this purpose, the students were expected to learn the basics and be able to interact with the MatchMaker server running on the teacher's computer.

7.3.1 Session Plan

First, the teacher presents the Java Reference Frame, which is not previously known to the students. Next, the students are asked to pass through five steps using the MatchMaker client:

1. Create a MatchMaker session on the server, with the name of the user.
2. List all available sessions on the server.
3. Join the MatchMaker session that was previously created in step 1.
4. Modify the MatchMaker session tree by adding an element.
5. Display the content of a MatchMaker session.

7.3.2 Java Reference Frame

The Java Reference Frame allows small programs to be written, compiled and tested. To do this, the following elements are available on the Java Palette as shown in figure 7.1:

- Java Node: it contains Java source code and it has buttons that allow a user to compile and execute small programs that can make use of additional jar files.
- Input Node: this is a text node that is used to inject as standard input into the execution of a Java Node connected through an edge.
- Output Node: it is a text node that receives the output during the execution of a Java Node connected to it through an edge.
- Import Node: it is a node that modifies the classpath of the JVM executing the Java Node it is connected to.

7.3.3 Related Queries

Along with the common queries described in 7.1, a specific query has been designed for this particular session. The base document delivered to the students contains one page for each of the five

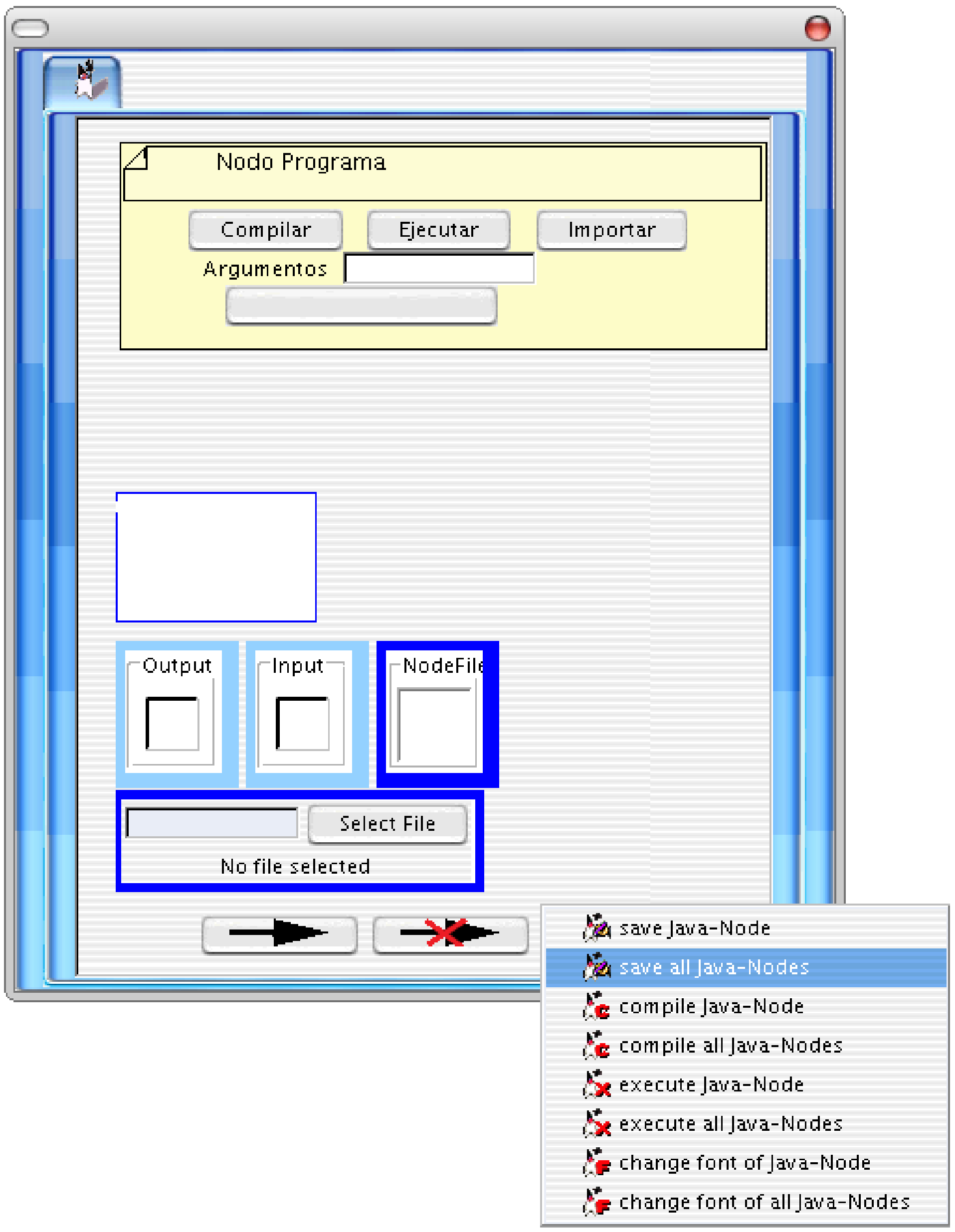


Figure 7.1: Java Palette.

steps mentioned above. For each of the pages, it is checked whether the Java code the students are asked to program effectively contains the expected methods.

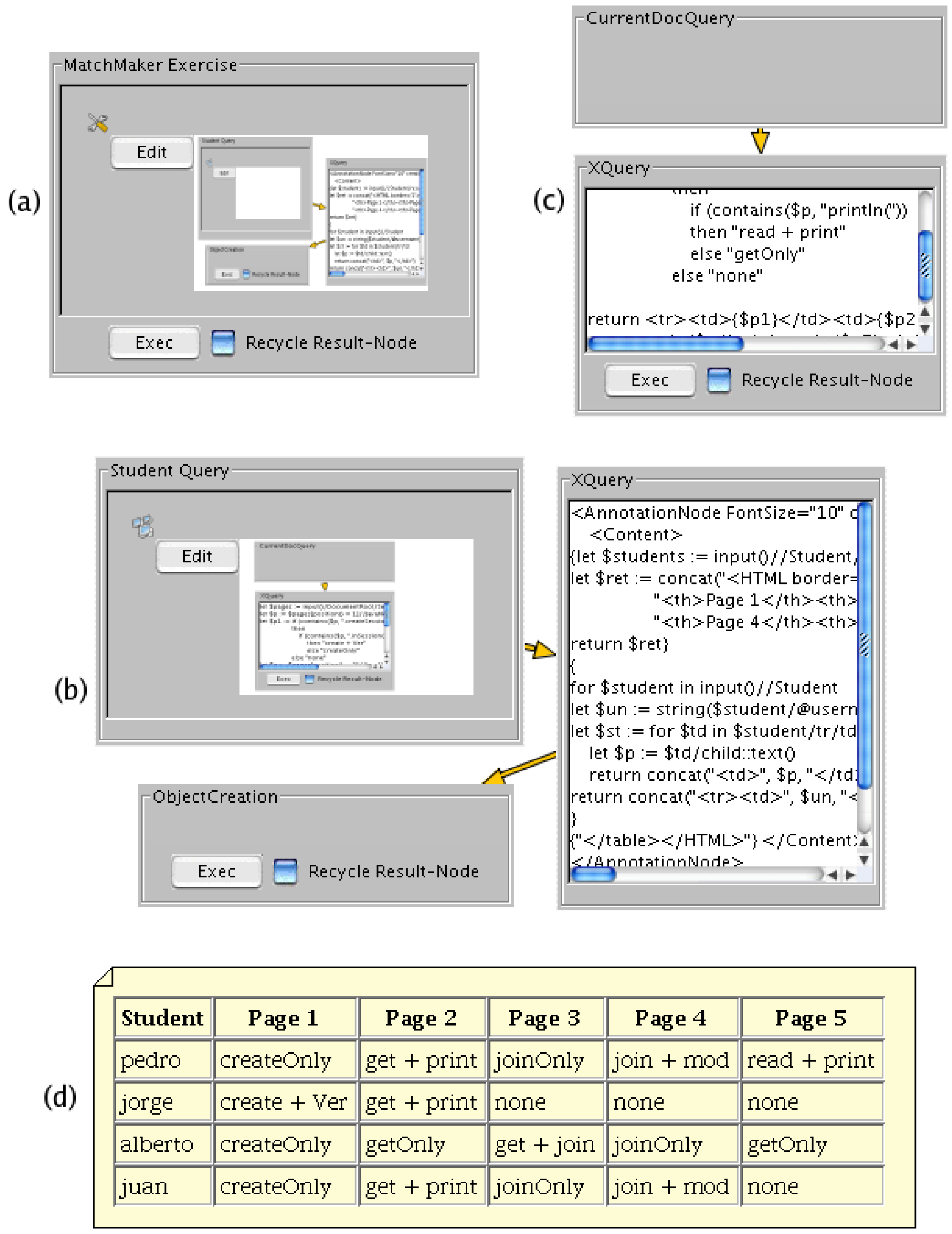
Based on the Java code found on each page, the student's progress is classified into several possible options. As it can be seen in the example in figure 7.2 part (d), the query creates a table in which each student gets his pages classified into one of several options for each page.

- Page 1: this page is classified into one of "none", "createOnly" or "create + Ver", indicating respectively that the implementation has not yet met any of the requirements, it has met the minimal requirement of creating the MatchMaker session, or additionally to creating the session, it also verifies that the session was successfully created.
- Page 2: similarly, the second page is classified into one of "none", "getOnly" or "get + print".
- Page 3 is classified into the possible values of "none", "joinOnly" or "get + join".
- Page 4 is classified into the possible values of "none", "joinOnly" or "join + mod".
- Page 5 is classified into the possible values of "none", "getOnly" or "read + print".

Alternatively, the teacher can request a summary of the previous information, thus treating the students as one single group that advances through several states or situations. At first none of the students will have completed the exercises correctly, and the state will change constantly. The desired final state is a situation in which all or most of the students have been able to complete the exercises successfully. Details of the implementation of the XQueries in figure 7.2 part (b) and figure 7.2 part (c) are available in section A.4, page 177.

7.3.4 Discussion and Results

It is possible to identify when the students are showing adequate progress on their tasks, since the teacher has an overview of value-added information. Students that are progressing at a different



```
      if (contains($p, "println("))
      then "read + print"
      else "getOnly"
   else "none"

return <tr><td>{$p1}</td><td>{$p2
```

```
<AnnotationNode FontSize="10" c
  <Content>
{let $students := input()//Student,
let $ret := concat("<HTML border=
          "<th>Page 1</th><th>
          "<th>Page 4</th><th>
return $ret}
{
for $student in input()//Student
let $un := string($student/@usern
let $st := for $td in $student/tr/td
  let $p := $td/child::text()
  return concat("<td>", $p, "</td
return concat("<tr><td>", $un, "<
}
{"</table></HTML>"}</Content>
</AnnotationNode>
```

Student	Page 1	Page 2	Page 3	Page 4	Page 5
pedro	createOnly	get + print	joinOnly	join + mod	read + print
jorge	create + Ver	get + print	none	none	none
alberto	createOnly	getOnly	get + join	joinOnly	getOnly
juan	createOnly	get + print	joinOnly	join + mod	none

Figure 7.2: Query specific for MatchMaker exercise

pace than expected will be noticed quickly by the teacher, making it possible to take a closer look either directly by approaching the student, or by using a query to fetch a specific page.

When a student is advancing faster than his peers, the teacher might want to provide additional feedback, ideas or assignments in order to keep stimulating the advanced students, or show the advance in order to discuss the solution with the other members of the class. On the other hand, when a student is having problems to solve the proposed exercises, the teacher can take measures to precisely identify and overcome the existing problems.

During the session, the system correctly identified the progress made by the students, allowing the teacher to be aware of the current state of the work performed by the students. The hints given by the teacher to the students were relevant, and they helped to guide the class in a natural way.

In particular, it was possible to identify that at least five students were not making any progress at all, despite the fact that they seemed to be very busy working on the exercises. Of these students, only two approached or asked the teacher for help. Upon a closer look it was clear that a technical problem was keeping the Java Palette from compiling properly, so the students had no way to verify the code they wrote would compile, much less verify the correct operation. Changing the laptop for a few of these students and having others working in groups with computers that were working correctly solved the problem, not only for the two students who proactively asked for help, but for all five of them.

As an alternative to using the queries presented here, it can be considered that the teacher walks through the classroom looking at the work done by the students. This has its own advantages, such as allowing the students to ask questions directly as the teacher passes by, or giving some hints personally at the moment a problem is detected. But by looking over the shoulder of the students, the teacher can only see the part of the model on which the students are currently working, both in the case of a CiC or when using pencil and paper to perform the modeling, programming or other activities. Reviewing a set of about 5 exercises for a group of 10 students (50 exercises) in total could be done in a setting without technological help. However, this review could only be afforded once or twice during a session, if at all, due to the time consumption of such a task. And even then, the review would not be deeper as the query presented above, the result of which can be updated every couple of seconds.

7.4 Stochastic Modelling

The stochastic reference frame (Lingnau et al., 2003) allows the modelling of random experiments.

7.4.1 Session Plan

First, the teacher presents the Stochastics Reference Frame, which is not previously known to the students. Afterwards, the students are asked to solve the Birthday problem: *find out how many students have to be in a course so that the probability that at least two were born on the same day is 50%.*

Once the students are used to the reference frame, they are asked to form groups of 2 or 3 people, and make a comparison of two currently promoted lottery games:

- Kino 5: seven balls are drawn from an urn containing 35 balls labeled 1 through 35. The first prize is won guessing 5 of the 7 numbers.
- Toto 3: one ball is drawn from an urn containing 1000 balls labeled 0 through 999. Equivalently, three balls are drawn from an urn containing balls labeled 0 through 9, and the drawn ball is returned into the urn each time. The first prize is won guessing the 3 digits in the correct order.

The question the students should respond is in which game it is most probable to win the first prize. The analysis should be done theoretically on one side and empirically by modelling one experiment for each lottery game.

7.4.2 Stochastic Reference Frame

The reference frame provides a series of elements that are classified into following categories, to allow the representation and execution of the random experiments. They are available through the Stochastic Palette that is shown in figure 7.4.2. The elements that this palette provides are the following:

- Urns: they are nodes that generate random elements. Examples of them include a Calendar that generates day/month combinations and urns filled with colored or numbered balls. These urns have a state, since they are initially filled and as they deliver elements, these elements will no longer be available unless returned into the urn or when the urn is reset. Other urns, like dice, coin or tetrahedron have no state, and always retain the elements, thus future draws are not influenced by the previous ones.
- Drawing Nodes: the urn is connected to a drawing node using a drawing edge, which is the way to fetch some value and put them into the urn. The values drawn are immediately returned back into the urn or not, depending on the type of the connecting edge. The single drawing node fetches a definable number of elements at a time. The multiple drawing node can repeat this experiment a number of times, refilling the urn between experiments.
- Containers: the elements or values that are drawn from urns are kept in containers. These containers keep a history of the results that can be visualized using result nodes.
- Filters: if not all drawn elements should be deposited into a container, the elements can be filtered using several possible filter nodes. Only the results matching the filters criteria will be passed on to the container.
- Result Nodes: a result node is connected to a container and it displays its content in a particular way.
- Drawing Edges: one edge draws elements from an urn node, without putting it back afterwards. The number of elements can be set by changing the number displayed on the edge. The other edge draws the specified number of elements from the urn, returning the drawn element each time.

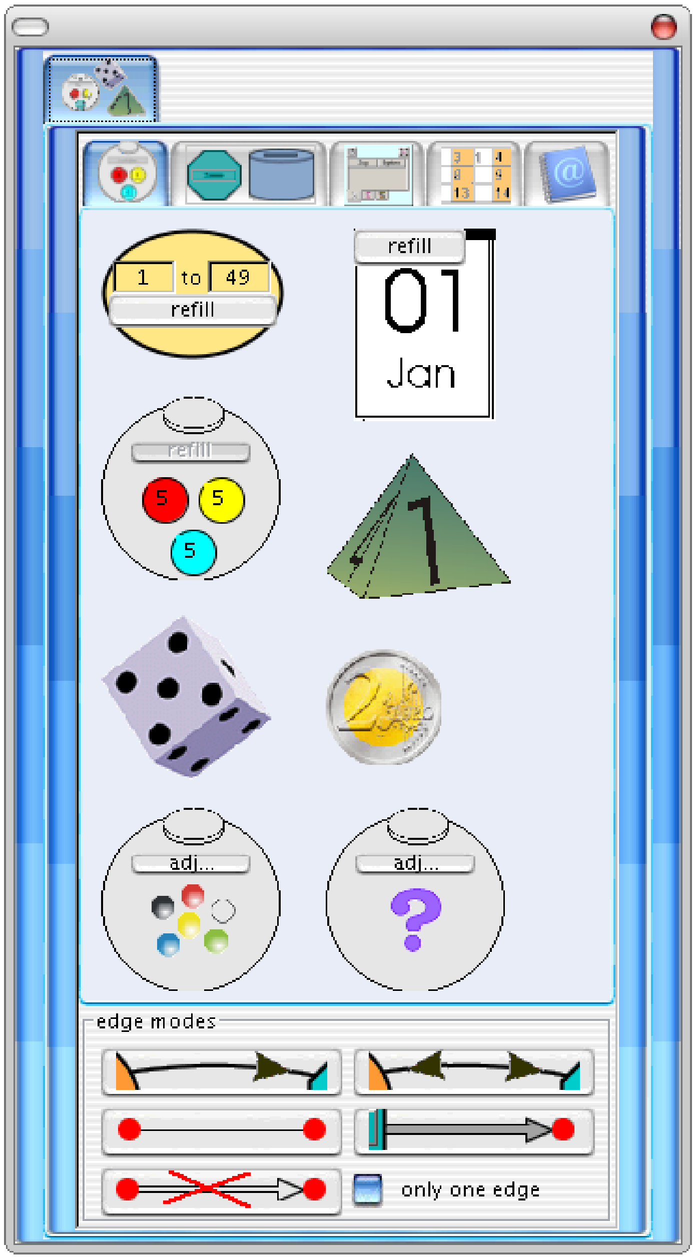


Figure 7.3: Stochastics Palette.

7.4.3 Related Queries

It is possible to create a specific query that helps to identify whether a solution is being provided successfully, in a similar way to the previous scenario. Moreover, in this case it is possible to solve the “Toto 3” problem (c.f. 7.4.1) in two different ways. Using a specific query it is possible to guess what approach each student is taking, and using this information to guide the students into e.g. discovering the alternative way to solve the same problem.

This information is also available to be viewed as a summarized state or situation. The teacher will start monitoring this state and ask for the detailed information including the state for each student as necessary.

7.4.4 Discussion and Results

The following two aspects are relevant in this exercise: the collaboration activities that the students carry out while solving the problems, and the result of the modeling. When analyzing the collaboration activities, it is necessary to consider that the students can collaborate on-line, using the coupling facilities available in CiCv2, but they also may choose to work on the same computer by using direct (not computer-mediated) collaboration. The use of direct collaboration, without the intervention of the computers, needs to be detected by the teacher.

The class was divided in groups of 2 to 3 students, and they had the choice to collaborate face-to-face, using the MatchMaker collaboration mechanism, or both. Most of the students created sessions for the MatchMaker collaboration, but in two cases only one student participated in the session. It can be seen by direct observation in the classroom and by using the video sequences, that in these cases the students collaborated face to face rather than using the MatchMaker server.

The statistics gathered from the collaboration log files as the session progresses did not show major changes in the distribution of actions among the participants. Generally it was possible to identify one or two students in a group that had a participation that was slightly over the level of the other participants. As the teacher can see these numbers, it is possible to take a closer look at the groups that show a pronounced difference of participation among its members, to verify that they are for example collaborating using direct face to face communication, or take action when a problem is encountered.

In the case of the Birthday problem, it was possible to identify common errors like using the wrong edge to connect a Calendar Urn to a drawing node. As expected, several users made the mistake of using drawing edge that did not return the values back to the urn, resulting in zero instances of two persons having the same birthday, no matter how many persons form the group. When the teacher detects this kind of errors, she determines the best way to handle the situation.

While it is possible to detect the common errors either in the classroom by looking carefully at each students model, it is a time-consuming and error-prone task. Using the video footing of the class does not show this kind of error.

7.5 System Dynamics Modelling

The System Dynamics reference frame (Bollen et al., 2002) allows to simulate the interactions in complex feedback systems over time. The interactions are based on stocks and flows connected by feedback loops. Using these models, the students can model real-world as well as imaginary situations, find the right balance for a system and verify the consequences of altering relations. Being a very general modeling language, System Dynamics can be applied in many types of domains, including ecology or biology (growth of population), physics (radio active decay, simulation of systems such as a water rocket), social science and medicine (spread of diseases).

7.5.1 Session Plan

First the teacher presents the System Dynamics Reference Frame and the principles for modeling using System Dynamics. The students have not used the System Dynamics concepts previously nor they have used the corresponding Reference Frame.

The second activity is the analysis of a prepared model that simulates the physical properties relevant to the launch of a water rocket. This model makes use of several inter-related variables, being relatively complex but still easy to understand.

As the last activity, the students are asked to find the balance in an ecology model where interdependencies between one species (rabbits), its food source and a predatory species (wolfs) do exist. The objective is to find the necessary relations and initial values as to find a cyclical behaviour of the system.

7.5.2 System Dynamics Reference Frame

The System Dynamics reference frame has two basic types of nodes: Stock nodes and Rate nodes. They can be seen in the System Dynamics Palette shown in figure 7.4. The Stock nodes maintain the current value of a variable inside the model. The Rate nodes specify the amount that is to be transfered between several stocks on each time iteration.

A Stock node can have a limited or infinite amount of stock available. In the case of a limited amount, it is possible to define the initial value, which is set every time the simulation is reset. Also, a stock node is identified by a name or identifier that can be used to compute the value of a Rate node in each iteration. In figure 7.4, two stock nodes can be seen: the first with the label *Stock* and the second on the right, in a cloud-like shape. The second one has an unlimited stock, whereas the first one has a limit.

Rate nodes specify the amount of stock that is to be transfered between two Stock nodes in each iteration. As already mentioned, the rate nodes can take the values of several Stock nodes, and use some math to compute the rate for the stock transfer in the next iteration. Additionally, the rate nodes can take information from two other types of nodes: a constant node that always delivers the same number, or a clock node. A clock node delivers a cyclically changing number as defined by a table. It is necessary to connect an Info Edge from the information source to the rate node, in order for the rate node to get the information from stock nodes, constant nodes or clock nodes. For each iteration, the rate of the stock flow for the next iteration is computed as a function of all nodes connected through info edges to this rate node.

The source stock node has to be connected to a rate node, using a Flow Edge that is available on the same palette, in order to transfer stock from one stock node to another. This rate node specifies the amount of stock to be transfered in the next iteration and it is connected to the second stock node that will receive the stock to be transfered, also using a Flow Edge.

Once a model has been created, it is simulated using the buttons and sliders inside the panels labeled *Step Control* and *Timeline*. It is possible to control the number of time units that are processed each time the button on its left is pressed by setting the *Step Control* slider. For each time unit, the corresponding stock is transferred and the rates are updated. Hitting the *Reset* button sets all of the stock nodes current value to the initial value. It is possible to use a Graph Display node to see the history of a stock node through all iterations. This Graph Display node is defined in a separate reference frame, called Function Reference Frame.

7.5.3 Related Queries

In this scenario, the used queries were mostly the common queries described in 7.1. The two most significant queries used are the “Model Complexity” query that shows the size of each model constructed by the students, along with the Student “Activity Indicator” query. In some cases, it was detected that the activity of the student changed from normal to a very low level. However, this is not always indication of the actual work of a student. In several cases the student stopped

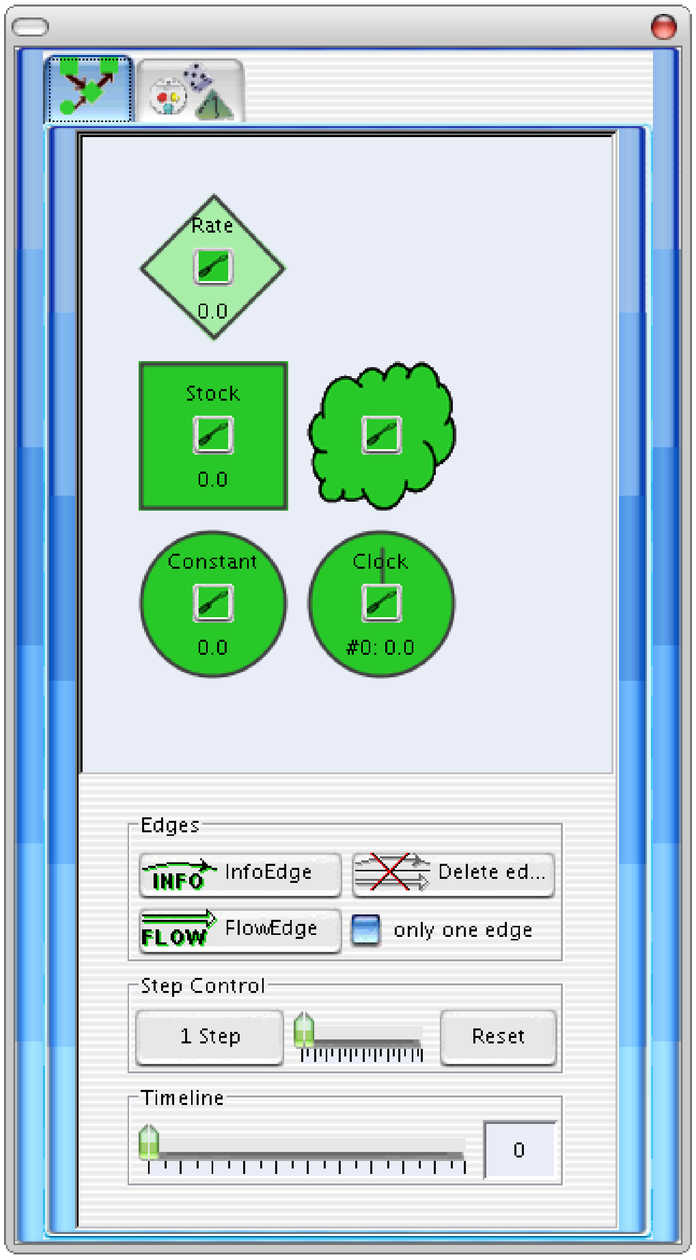


Figure 7.4: System Dynamics Palette.

working at his computer to help or share information with fellow students. In almost all of these cases, the students also worked together using only one of the two available notebooks at least some of the time.

7.5.4 Discussion and Results

In this case, the students had more freedom to model the situation than in the other case studies, without having to adhere to a precise characterization of a model that is considered to fulfill the requested task. Therefore, the teacher made use of queries that are not specific for the domain being modeled, and based on the results she identifies potentially interesting student documents.

It was possible to identify cases in which students were working on models that appeared extremely complex when compared to the model solution available to the teacher. In all cases where the difference was above twice as many nodes and edges as expected, a close look revealed that they were creating a new model on the same page as the first one. Other students also decided to start a new model because the first one was not satisfactory, but by creating the new model on another page, or by deleting the previous model before starting over. The three approaches seen in this session do provide different results. A teacher needs to be able to interpret these results, verify that the real cause of the results effectively fits with the possible interpretations (e.g. by approaching a student or using a query to view the students model) and possibly take remedial actions.

It is not always necessary for each teacher to make their own interpretation of possible causes for the results of a query. The courseware should include the reasoning and possible interpretations of the results. A teacher should expect the courseware to include the queries that are specific to this classroom situation or activity. Additionally, the courseware should include some suggestions of how to react based on the results. For example, in this case it should provide range of values and relations among values that have been found to be typical in a setting in which no problems arise. Additionally, the courseware should indicate values and relations among values that have been found in situations where certain problems occurred, such as students not knowing how to use the modeling tools, lack of familiarity with the theory of the model, or when the students have already solved the posed problems. A teacher may interpret the results accordingly, verify or alternatively disambiguate the interpretation and decide what action to take by using these hints.

In the examples shown in this chapter the teacher had no need to perform changes on the queries more than modify at most one parameter for a specific query. Thus, it is not possible to make any evaluation on the usability of the querying system on a deeper level.

Chapter 8

Discussion and Conclusions

The hypothesis stated in 1.2 is the following:

Hypothesis: It is possible to identify situations in a classroom, based on the information available within the system. These situations can be identified automatically by using a tool and that reduces the effort required for the identification process.

It is quite clear that the information available in a classroom environment, specifically in the case of the CiCv2, makes it possible to identify several different classroom situations, by looking at the experiences in the scenarios described in 7. It is then feasible to select recommendations according to the identified situations and to apply them as stated in the Best Practices being used.

In the case of the MatchMaker exercise (7.3 on page 123), it was possible to find students that shared a common unforeseen problem and act to mitigate the negative consequences. It is important to execute the action in time, otherwise the session will have no positive value at all for the affected students. In this sense, detecting the problem after the session has ended would not be enough, hence a querying system to be used during the sessions is much more useful than any post-session evaluation, including video footage and any kind of assessment that does not deliver immediate results.

Common errors can be useful as a pedagogical tool for initiating discussions in which the students know why the solution was thought to be correct, and learn which assumption or part of the solution was wrong. In the exercise involving the stochastic palette (see 7.4 on page 129) it is possible to see examples of how the common errors can be detected and shown to the teacher in an aggregated form. It is possible for the teacher to ask the students to tell the kind of solution they used to solve the proposed problem, but it is helpful for the teacher to have a list of students that have made each common mistake. For example, it would not be necessary to interrupt the students' work every time the teacher wants to determine how many students are making common mistakes. The teacher can access the necessary information and interrupt the work only when it turns out to be necessary.

As seen with the System Dynamics session (see 7.5 on page 133), it is possible to measure abstract values that help to correctly identify potential problems. This helps a teacher to use her limited time resources on solving problems rather than looking for them.

In all of the settings, efforts were made to access the same information by other means. In all cases, the effort placed on behalf of the teacher or person dedicated to this task was significantly higher. Also, the results would be available after a relatively long time or even after the session. In particular, the results of such efforts are no more accurate than the querying results, and in cases where the teacher or an assistant needs to look at the students work, the results are generally less accurate. That is because the viewer cannot generally see the whole model the student is working on, and he can only focus on one student at a time, so a summary would contain information gathered at different times.

8.1 Discussion of the proposed solution

In this section it will be discussed how the proposed solution contributes to solving the problem of finding Best Practices in a Computer-integrated Classroom environment. Also, an important aspect of the discussion is how this solution can be applied to other contexts considering other target groups as well as other learning environments than the ones considered this far.

8.1.1 Validation of proposed Best Practices

The teacher needs to compare the outcome of the activities with the outcomes expected by the proposed best practices in order to validate a best practice. However, it is necessary to analyze not only the outcome, but also whether the best practices were applied in the planned way. Since best practices can be applied correctly or incorrectly, and it is possible that the measures indicate a correct or incorrect application in both cases, it is necessary to analyze the four possible situations in the validation process:

- **Best practices applied correctly, measures indicate correct application**: in this case, the querying system shows a great advantage, since it reduces the effort and time devoted to evaluating the measures for the proposed recommendations.
- **Best practices applied correctly, measures indicate incorrect application**: the used measures are proven to be inadequate to conclude any decision on the application of the best practices. The recommendations are applied in a scenario different to the original, and it is possible that not all variables were considered when defining them. This is an opportunity to study the case more closely and improve the definition of the Best Practices. It requires a greater effort on the teachers behalf, but that effort is part of the validation process that is needed in order for the best practices to be meaningful in other contexts.
- **Best practices applied incorrectly, measures indicate correct application**: this would be the worst case when applying the best practices without any other evaluation. The false sense

of security provided by the measures that do not reflect the reality may defer the detection of a problem in the learning process of the students. Therefore it is vital that the validation process includes testing in a wide variety of situations, trying actively to trigger this kind of failure. Only when confidence is strong that this kind of error will not appear when applying the best practices in everyday use, the validation period might be considered complete.

- **Best practices applied incorrectly, measures indicate incorrect application**: in this case, just as the first one, increases the confidence in the system. The effort saved through the use of the automated measures to help the teacher decide the actions to follow and to conclude that problems do exist is an advantage. But at the same time, this situation might indicate that the proposed Best Practices are incomplete. Unless there is a serious problem that impedes a normal application of the practices, the recommendations should include actions to be taken in order to overcome minor problems during the application of the practices.

In conclusion, it is possible to diminish the effort required on behalf of the teacher to apply existing and validated Best Practices in an environment like the CiCv2. It is, however, important that the validation process is as thorough as possible, in order to avoid the situation in which a teacher does not detect the problems that are present in a classroom session and should be solved.

The accuracy of the validation process is not generally verifiable. It depends on the specific metrics that are being used to determine the correct application of a best practice, and on the evaluation of whether the outcome is successful or not. This is an area that needs attention in future investigations.

8.1.2 Methodology to Define, Validate and Apply Best Practices

The definition of Best Practices is relatively straightforward at first, as it does not differ much from current practices. The first step is the definition of the recommendations, that can be based on already existing Best Practices. The most important work regarding the definition of Best Practices is to establish how the classroom situation is to be measured, and what actions or recommendations apply in each possible situation. This step is not trivial and it requires skills beyond the teaching of the subject matter, since new queries have to be constructed and adapted to the particular situation.

A teacher would get the Best Practices along with the courseware for a given scenario. The Best Practices include the queries that provide measures to the teacher, who may then decide which recommendation or action to follow based on the result of these measures.

Validation of Best Practices is necessarily a collaborative process. As already mentioned, it is necessary to apply the Best Practices in as diverse contexts as possible, in order to make sure that all of the variables are being considered. During the validation, the teachers should apply the Best Practices and use the provided queries, but they also should verify that the results of the measures do make sense and that the recommendations effectively influence the classroom situation in a positive way. When finding a problem, it is necessary to improve the Best Practices by sharing the experience and finding the best solution that solves this particular problem and keeps the Best Practices applicable to previous experiences. In this regard, the process is like the process used in several successful Open Source Software developments: the users (in this case the teachers) submit bug reports and together with the developers (in this case the persons defining the Best Practices) make changes to the definition until the problem has been fixed. This is only possible in a collaborative process, where every teacher has access to deliver her own experience and problems.

It makes sense to consider the definition and validation of best practices as a learning process on how to best learn a given subject matter in certain situation. As students use collaborative processes to improve their knowledge and learning skills, the teachers can also benefit from the collaborative process of exchanging their experience and determine how to best learn a given subject in several different situations, as to improve their understanding of the process.

8.1.3 Privacy and Monitoring of Students' Activities

The solution being proposed in this thesis has been centered on the technical aspects to provide the teacher with valuable information at the right time. However, non-technical aspects can have an important impact in the viability of any application of this technology. One of the non-technical aspects to consider is the privacy of the students. It is necessary for the teacher to be allowed to use the information available in the classroom, and in some cases this can conflict with the students' right to privacy.

The importance of this aspect depends on the culture in which the activity monitoring is to be inserted. In a country with strong values regarding respect of individual privacy the issues will be different as another country in which the

In this thesis it has been argued that the scope should be as wide as possible, which means that the system will probably be used in several different cultural contexts. It is therefore necessary to look carefully into every aspect regarding privacy issues with the use of these tools.

Generally, privacy issues are brought up when private or sensitive information is made publicly available, or when such information is collected for the benefit of a third party like an enterprise or a government. In the CiCv2 the collected information is used for helping the teacher in her guidance in order to make the classroom experience as best as possible. The information is not intended to be published, and it is not accessible to other persons but the student and the teacher. In some cases the information is used to present a problem regarding current best practices to fellow teachers, and thus help improve the best practices for all involved students. It is advisable to provide and enforce a privacy policy that does not disclose personally identifiable information on the students in each of these cases. Nevertheless, it might be necessary to make the students aware of the information that will be collected, the purposes of that collection and to get their consent (or the consent of their parent or guardian when applicable).

It is technically possible to add privacy rules to the CiCv2, in which the student might define which content should be considered private and thus not available to the teacher or proposed querying system. For all practical issues, this is equivalent to the student taking personal notes. However, in some circumstances the privacy rules may present some problems when a student inadvertently makes information private without intending to do so. This would have the consequence that the teacher cannot discriminate a situation where the student is not participating at all in the session and another in which the student participates but has marked the relevant documents or pages as private.

The CiCv2, as well as most other implementations of a Computer-integrated Classroom, collects only internal information. Any information or activities managed outside the scope of CiCv2 are “invisible” to the system, and thus to the querying system presented in this thesis. This gives some control of exactly what information is managed back to the student. If the student wants to maintain some activities out of the scope of the teacher, this can be perfectly possible by simply using other tools to handle that information. The teacher will obviously still be able to evaluate the performance of the student, so when the student engages in other activities while he is expected to be working on some assignment it will be detected by the teacher. In cases where the students are supposed to be exclusively focusing on the proposed work, the system can be configured to allow only interactions that occur through the use of CiCv2 and block all other uses of the computers. The decision depends on the pedagogic methodology to be used, the profile of the learners, and several other variables.

Whether or not students have access to perform private activities during classroom sessions is up to the teacher or educational institution to decide. In the case of allowing other activities, the teacher will still have the ability to see each student’s commitment to the assigned tasks, without interfering in the privacy of the student. This can be seen as a positive feedback where the teacher asks the student to fulfill the expected goals, keeping an account of how much time the student actually spent making changes to the data that will be delivered as a result. This is already more information than is available to a teacher using traditional classroom tools, and is not invading the student’s right to privacy.

It can be said that the access to the data on behalf of the teacher is comparable to asking the students deliver their results in form of a written homework. The teacher needs the result to be delivered in order to grade the student’s work and evaluate the overall progress of a class. In the case of CiCv2, however, it is also possible for the teacher to have information on the process through which the result is created by the students. However, this information is unavailable in a traditional classroom for technical reasons, not because there was any intent to hide it. Assuming the students are aware of what information is available to the teacher, the access to that data on behalf of the teacher is equivalent to asking the students to deliver not only the result of a task, but also the process in which the task was solved. This is not a new idea, since it is often used in traditional classrooms, obviously at a higher cost since the process has to be incorporated explicitly by the students.

8.1.4 Applicability in other scenarios

The CiCv2 including the querying system has been used in few specific scenarios in the context of this thesis. Thus, it is necessary to ask whether the same or similar uses can be made in other contexts. In this section it will be considered how the system can be used by other students and in situations that are not related directly to face to face classroom sessions.

Other Students

The CiC has been successfully applied to other student profiles before. One of the implementations of a CiC is the Nimis project 2.7.2 in which children of ages between 4 and 8 have used a specially designed version of a CiC. The information available in this case would still be relevant. The use of other methodologies for learning, other goals and a different context than the previously examined means that the best practices need to be different. And the same way, the queries used to get useful information from the system will certainly be different.

The same is true when applying the system in other contexts, for example different cultures, learning styles or other differences. The information is still relevant in all cases. It might be that in some contexts, where collaboration is not an option, the features allowing us to handle the collaboration might be of no use, but the system can still be applied using the rest of the information.

e-Learning

In distant learning scenarios in general, and e-Learning in particular, the information available to the teachers, tutors and other professionals is comparable to the information used in the CiCv2. However, in a remote and generally time-shifted environment the need to access the information on demand and in a given time frame is no longer a priority. Therefore, one of the benefits the CiCv2 querying system provides in the face to face scenario, namely giving feedback to the teacher within an almost real-time approach, is not an advantage in most cases.

However, given that the information available in a distant learning environment is of the same nature, the same queries can be applied, taking care of conversion to the specific file formats. Moreover, the use of information technology is seeing an increasing use of XML to store all of the relevant data, and it is possible to use queries in the face to face context as well as in distributed, time shifted scenarios. The resulting recommendations that form part of Best Practices will need to consider the context, but they also can be shared to some extent, and applied according to the limitations of each environment.

Traditional Classrooms

Any Best Practice developed using the proposed tools as part of their methodology can also be applied in most cases to a traditional classroom. The main disadvantage is that it is not possible to validate whether the practices are being applied correctly, and it might require additional work on behalf of the teacher or other professionals assisting her. The downside is that in many cases, the activities that are common in CiC-like classrooms are not easily adapted for use in traditional classrooms, since they involve the use of tools that require computers. This limits the application of specific best practices.

8.2 Other Solutions

The system proposed in this thesis is not the only way to define Best Practices. Traditionally, Best Practices have been defined by distinguished professionals, based on their own experience. The methodology used in those cases can vary, but according to Ruchti (2002), Best Practices are usually developed through “anecdotal and descriptive research”. However, it is perfectly possible to develop effective Best Practices this way. In this case, a teacher implementing these practices will not be able to compare objective measures to the intended results. The measures have to be replaced by some other means in if the teacher intends to verify how the application of Best Practices is progressing. In that case, a teacher will need to assign some resources, either personally or having another person help out, to verify the application of Best Practices.

One of the barriers to a widespread adoption of information technologies inside the classrooms has generally been the cost of such implementations. It is tempting to ask whether it is possible to achieve a similar result using a system that does not involve the usage of a CiC or similar environment. It is certainly possible to use such a similar system in the context of a traditional classroom. Observers can accurately deliver data on the student’s work, and the results should not differ exceedingly from the data obtained automatically. However, in that case the cost of having those observers, who should be trained professionals, can be quite high, so the cost advantage is not a benefit.

When considering a traditional classroom in which the teacher does not have any observers to aid in the gathering of the necessary information, the teacher can take on all of the workload to collect the information. This activity is obviously time consuming, and the teacher cannot be expected to finish all of her work and additionally perform computations that might in some cases be relatively complex. As a consequence, the teacher may take some time and have the results for another session and take the time in between to gather any necessary information, the same way a teacher normally would grade homeworks or assignments in order to evaluate the progress of the students. However, it is difficult to use that mechanism manually in order to get results within the time frame of a classroom session.

It is certainly possible to consider other information systems that process information available in a CiC setting. Some examples are presented in Barros and Verdejo (2000) and Mühlenbrock (2001). These examples are useful for the specific scenarios they are defined for, and can be used in a complementary way to the querying system described in this thesis. Moreover, it is possible to link those systems and use them in conjunction. They are to be considered complementary systems.

8.3 Relevance

The usage of Information Technologies in education has been strongly suggested for some time. However, the results are not always as good as expected. As a matter of fact, a study titled "The Impact of Information and Communication Technologies on Pupil Learning and Attainment" (Harrison et al., 2002) states in its conclusions that *there is no consistent relationship between the average amount of ICT use reported for any subject at a given key stage and its apparent effectiveness in raising standards. It therefore seems likely that the type of use is all important*. When considering the usage of Information Technologies in education, it is important to consider how they will be used. The consistent usage of Best Practices can help considerably in achieving the expected results.

According to (Calculator, 1991), best practices *are continuously challenged, modified, discarded, and then re-invented with new fervor at a later time or in a novel context*. Also, the corresponding *procedures should be replicable and their effects measurable*. The approach presented in this thesis allows for replicable Best Practices, making an advancement in a particularly important aspect.

8.4 Conclusions

It is possible to verify both hypothesis defined in this thesis, first to identify situations in a classroom by using information available in the classroom system, and second to create a tool that helps reducing the effort for following a set of recommendations. Moreover, the data needed as input for the system to obtain the information tends to be available in any CiC-like classroom scenario. In cases where the information is not readily available, it is possible to modify the software so it provides the necessary information.

The information needed for the querying tool to work is completely specified in 6.4.1, where the `QueryResourceProvider` interface is described. It is relatively simple to implement this interface, and in that case the querying tool can be used in scenarios outside the CiCv2. One of the requirements for the results is that they be available in XML. Generally, the data can be converted to XML on demand, but as XML is becoming ever more pervasive, specifically in the context of digital educational material and related tools, it should be straightforward to implement the required methods almost seamlessly.

It is also possible to use the same concepts to e-learning, adapting some of the concepts to the fact that in e-learning we do not have the same kind of sessions, but rather longer, asynchronous interactions. But in e-learning scenarios the students also interact with their peers, modify documents, work on assignments and this information is available on a repository. Therefore, it can be inferred that the presented work can be used not only in the context of CiCv2, but potentially in a broad scope of applications with a relatively small amount of effort.

8.5 Future Work

During the course of this thesis, a number of interesting issues have come up that can be targeted in future projects. In this section we mention the most interesting topics that were identified.

8.5.1 Specification of Situations and Best Practices

A good specification of the situations and best practices is an important issue. The work presented here is a basis that allows, once a situation has been specified, to identify whether it has been reached or not. The same is true about best practices: once a best practice has been defined, the tool and methodology presented in this thesis allows us to automatically obtain information on whether the best practice is being applied as intended and allow us to validate or dismiss a candidate best practice. Some advances have been made in work by Hernández-Leo in Hernández-Leo (2005), and it is interesting to evaluate whether the proposed definitions fit the requirements.

8.5.2 Generating repositories Courseware including Best Practices

As it has been shown that it is possible and desirable to use Courseware that includes not only Best Practices but also queries that help the teacher in applying these practices, it is necessary to use such a system in several environments and evaluate the results.

8.5.3 Applying the Queries in other Learning Environments

Certain parts of the Query Reference Implementation are dependent on the Computer-integrated Classroom V2 (CiCv2, see 5), particularly the presentation of the results, the XML representation of the document data as well as the format of the log files. However, the implementation of the queries is not necessarily dependent on the existence of the CiCv2 architecture. It is possible to use the queries in other environments, as long as the queries can have an implementation for the QueryResource (6.4.1) and a way to connect queries using the QueryConnection, QueryConnectionLink and thus SubQuery interfaces (6.4.2).

It makes sense to consider the creation of an ecological approach to learner modelling as proposed by McCalla (McCalla, 2004). The querying system can then be used to access information that is not necessarily generated inside a CiC, and even combine information from several independent systems.

8.5.4 Improving the Query Development process

The development of queries in the CiCv2 environment can be improved, specially in the development of the XQuery definitions, by allowing graphical tools and other aids to be used. As an example, one of the XQuery engines used in the implementation of the XQuery, QizXOpen (Franc, 2005), provides a graphical user interface for defining queries. The usage of such a tool, or alternatives like "XQuery by Example" (Braga et al., 2005), can improve the query definition productivity.

8.5.5 Defining strict typing for Queries

The queries used within the experiments do not use type checking. It makes sense to define types in some cases, in order to facilitate the reuse of queries in diverse contexts. A type would define the exact format of input a query is expecting, or how the output format for a particular query is defined. When the output type of a query fits with the input format of another, it can be assured that both queries can be connected.

8.5.6 Integration of external Processing

It has been shown that it is useful to present several sources of information to the teacher in a consistent presentation, and particularly without involving the teacher in the manipulation and interaction with several independent sources. XQuery is a relatively powerful language that allows a wide range of processing to be done, but it would be a huge benefit to be able to integrate external applications or processes and incorporate the results thereof back into the information flow. Certain programming languages provide benefits in processing complex analysis on the data over XQuery, and in most cases, it is useful to reuse an existing implementation instead of implementing it all over again.

8.5.7 Interactive Queries

It was possible to face some limitations that can be overcome by including "interactive queries" during the usage of the queries. That is, queries that ask the teacher for input at some stage, and then continue to execute. For example, it might be possible to select one alternative among a list generated on the fly. This way, it will be possible for the teacher to avoid typing the name of the student and the number of the page she wants to look at, but simply selecting the name of the student from a list presented after starting the query, then selecting one of the possible page numbers also from a list. This has several advantages, not only limited to the teacher's comfort, but reducing the potential of errors by writing wrong input. But mostly an interactive query allows making the system more usable, considering that the teacher might not have a keyboard nearby in the classroom setting, but typically has access to a pointer on the whiteboard, so avoiding the typing of input can be quite a benefit.

Appendix A

CiCv2 Implementation Details

A.1 Security and Authentication

Users of CiCv2 have to authenticate at the repository before any interaction takes place. A user identifies himself using an object of class `Student` or `Teacher`, both of which are extensions of class `Person` as seen in figure A.1. The repository contains information about the users, their role (Student or Teacher) and the authentication scheme. The default authentication used within the CiCv2 is a password for each user, that is kept in form of a hash. Based on the stored information, an object of class `X500Principal` [1] is generated. All the security access verifications are based on that `X500Principal`, making the system flexible for using a wide range of authentication mechanisms. Any authentication, from plain login and password, over to sophisticated retinal scan or physical token based certificates can be easily adapted using the standard X.509 PKI Certificates.

On each request, the identity of the connecting `Person` is verified using the X500Principal object and the requested target. As will be seen further on, the target can be of several kinds, including the execution of some query, retrieval or uploading of a file, or request for information.

[1] The complete name is `javax.security.auth.X500Principal`

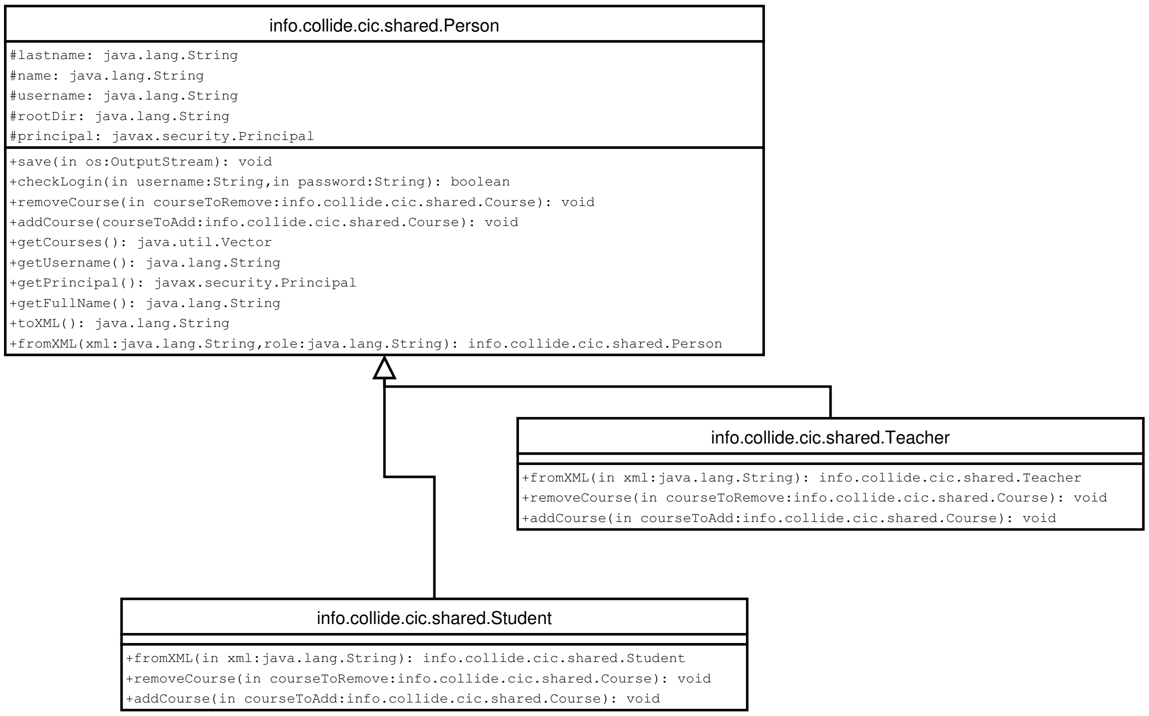

info.collide.cic.shared.Person
#lastname: java.lang.String
#name: java.lang.String
#username: java.lang.String
#rootDir: java.lang.String
#principal: javax.security.Principal
+save(in os:OutputStream): void
+checkLogin(in username:String,in password:String): boolean
+removeCourse(in courseToRemove:info.collide.cic.shared.Course): void
+addCourse(courseToAdd:info.collide.cic.shared.Course): void
+getCourses(): java.util.Vector
+getUsername(): java.lang.String
+getPrincipal(): javax.security.Principal
+getFullName(): java.lang.String
+toXML(): java.lang.String
+fromXML(xml:java.lang.String,role:java.lang.String): info.collide.cic.shared.Person
info.collide.cic.shared.Teacher
+fromXML(in xml:java.lang.String): info.collide.cic.shared.Teacher
+removeCourse(in courseToRemove:info.collide.cic.shared.Course): void
+addCourse(in courseToAdd:info.collide.cic.shared.Course): void
info.collide.cic.shared.Student
+fromXML(in xml:java.lang.String): info.collide.cic.shared.Student
+removeCourse(in courseToRemove:info.collide.cic.shared.Course): void
+addCourse(in courseToAdd:info.collide.cic.shared.Course): void

Figure A.1: Simplified Class Diagram for Person

Features of the `java.lang.SecurityManager` (provided by the Java language)are used to guarantee that `Person` will be granted access only to authorized files and resources. Prior to accessing some resource, the Java libraries (part of the Java Virtual Machine or JVM that executes the Java code) verify the access restrictions the JVM may be imposing on the `Person`. In the case of CiCv2, a custom security manager is used, that receives all of the requests and can decide whether to grant the access or throwing a `java.lang.SecurityException`, in which case the requested action is aborted.

The Security Manager used in the CiCv2 Repository grants all file-related actions, and checks the actions that involve files in the following way:

- All files needed for internal uses of a JVM are granted read-only access. This includes all files inside the JVM properties `java.class.path`, `java.library.path`, `java.home` and other system-specific paths.
- Users need to write and read files in the `java.io.tmpdir` directory for some queries, so access to this resource is also granted.
- The root directory where the repository stores its files needs to be granted read access.
- Each user may access its own home directory inside the repository.
- Course home directories: each student may read the course home directory for all courses in which he is registered, a teacher may read and write into course directories in which she is registered.
- Other permissions: the above mentioned default permissions may be changed on a per-user basis, granting additional access for users to specific files or directories.

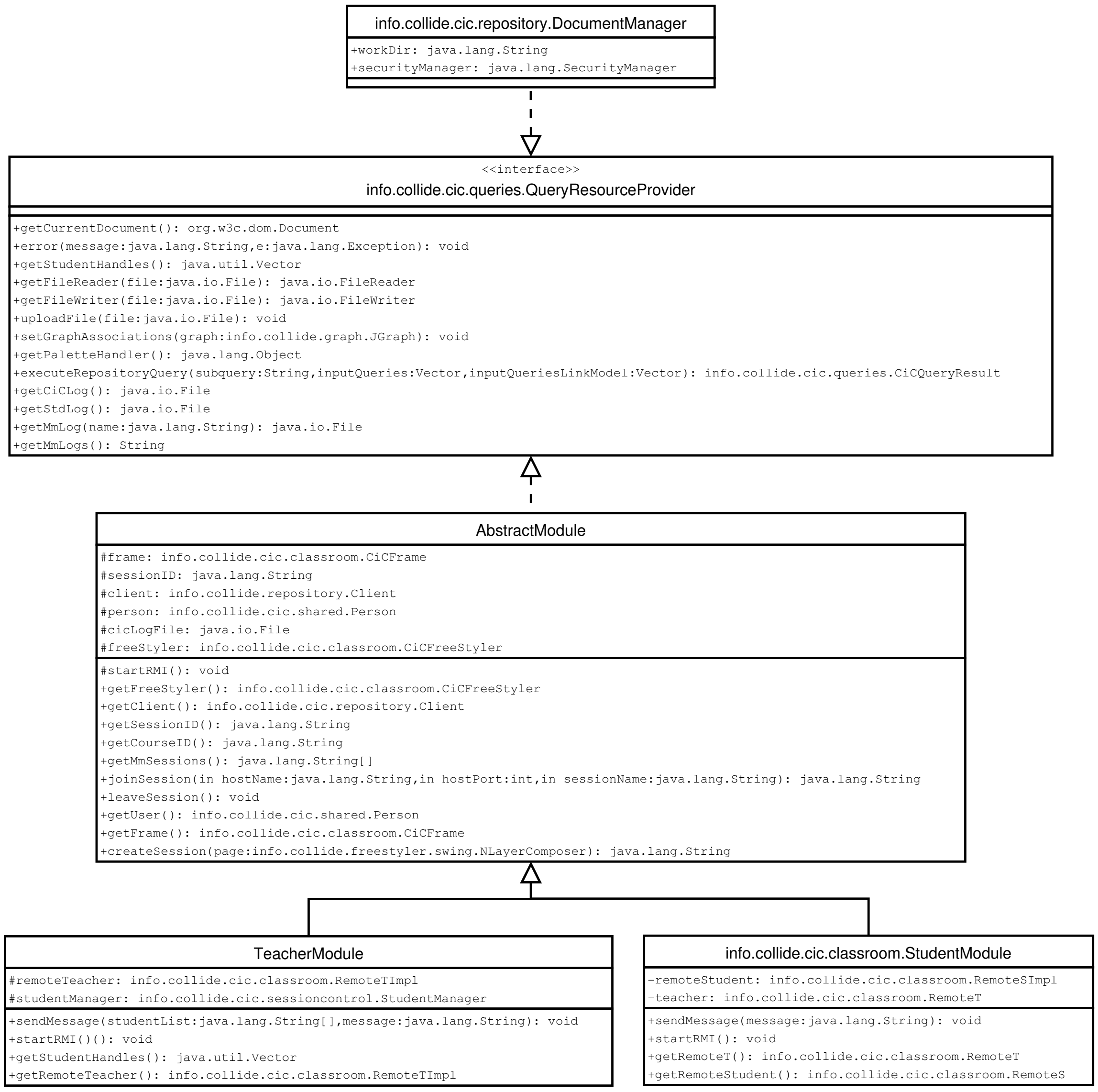

info.collide.cic.repository.DocumentManager
+workDir: java.lang.String
+securityManager: java.lang.SecurityManager
<<interface>>
info.collide.cic.queries.QueryResourceProvider
+getCurrentDocument(): org.w3c.dom.Document
+error(message:java.lang.String,e:java.lang.Exception): void
+getStudentHandles(): java.util.Vector
+getFileReader(file:java.io.File): java.io.FileReader
+getFileWriter(file:java.io.File): java.io.FileWriter
+uploadFile(file:java.io.File): void
+setGraphAssociations(graph:info.collide.graph.JGraph): void
+getPaletteHandler(): java.lang.Object
+executeRepositoryQuery(subquery:String,inputQueries:Vector,inputQueriesLinkModel:Vector): info.collide.cic.queries.CiCQueryResult
+getCiCLog(): java.io.File
+getStdLog(): java.io.File
+getMmLog(name:java.lang.String): java.io.File
+getMmLogs(): String
AbstractModule
#frame: info.collide.cic.classroom.CiCFrame
#sessionID: java.lang.String
#client: info.collide.repository.Client
#person: info.collide.cic.shared.Person
#cicLogFile: java.io.File
#freeStyler: info.collide.cic.classroom.CiCFreeStyler
#startRMI(): void
+getFreeStyler(): info.collide.cic.classroom.CiCFreeStyler
+getClient(): info.collide.cic.repository.Client
+getSessionID(): java.lang.String
+getCourseID(): java.lang.String
+getMmSessions(): java.lang.String[]
+joinSession(in hostName:java.lang.String,in hostPort:int,in sessionName:java.lang.String): java.lang.String
+leaveSession(): void
+getUser(): info.collide.cic.shared.Person
+getFrame(): info.collide.cic.classroom.CiCFrame
+createSession(page:info.collide.freestyler.swing.NLayerComposer): java.lang.String
TeacherModule
#remoteTeacher: info.collide.cic.classroom.RemoteTImpl
#studentManager: info.collide.cic.sessioncontrol.StudentManager
+sendMessage(studentList:java.lang.String[],message:java.lang.String): void
+startRMI()(): void
+getStudentHandles(): java.util.Vector
+getRemoteTeacher(): info.collide.cic.classroom.RemoteTImpl
info.collide.cic.classroom.StudentModule
-remoteStudent: info.collide.cic.classroom.RemoteSImpl
-teacher: info.collide.cic.classroom.RemoteT
+sendMessage(message:java.lang.String): void
+startRMI(): void
+getRemoteT(): info.collide.cic.classroom.RemoteT
+getRemoteStudent(): info.collide.cic.classroom.RemoteS

Figure A.2: Simplified Class Diagram for Applications

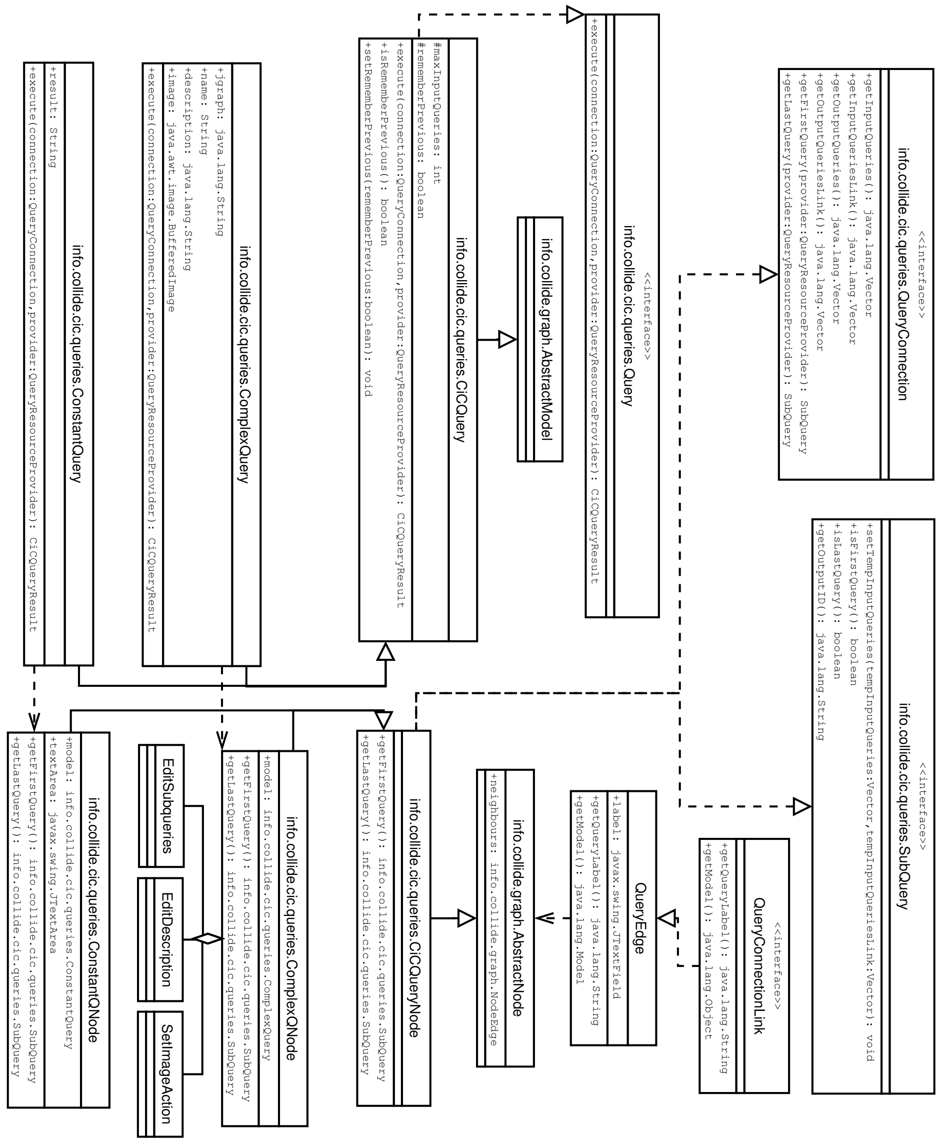

info.collide.cic.queries.ConstantQuery
+result: String
+execute(connection:QueryConnection,provider:QueryResourceProvider): CiCQueryResult
info.collide.cic.queries.ComplexQuery
+jgraph: java.lang.String
+name: String
+description: java.lang.String
+image: java.awt.image.BufferedImage
+execute(connection:QueryConnection,provider:QueryResourceProvider): CiCQueryResult
info.collide.cic.queries.CiCQuery
#maxInputQueries: int
#rememberPrevious: boolean
+execute(connection:QueryConnection,provider:QueryResourceProvider): CiCQueryResult
+isRememberPrevious(): boolean
+setRememberPrevious(rememberPrevious:boolean): void
info.collide.graph.AbstractModel
<<interface>>
info.collide.cic.queries.Query
+execute(connection:QueryConnection,provider:QueryResourceProvider): CiCQueryResult
<<interface>>
info.collide.cic.queries.QueryConnection
+getInputQueries(): java.lang.Vector
+getInputQueriesLink(): java.lang.Vector
+getOutputQueries(): java.lang.Vector
+getOutputQueriesLink(): java.lang.Vector
+getFirstQuery(provider:QueryResourceProvider): SubQuery
+getLastQuery(provider:QueryResourceProvider): SubQuery
<<interface>>
info.collide.cic.queries.SubQuery
+setTempInputQueries(tempInputQueries:Vector,tempInputQueriesLink:Vector): void
+isFirstQuery(): boolean
+isLastQuery(): boolean
+getOutputID(): java.lang.String
<<interface>>
QueryConnectionLink
+getQueryLabel(): java.lang.String
+getModel(): java.lang.Object
QueryEdge
+label: javax.swing.JTextField
+getQueryLabel(): java.lang.String
+getModel(): java.lang.Model
info.collide.graph.AbstractNode
+neighbours: info.collide.graph.NodeEdge
info.collide.cic.queries.CiCQueryNode
+getFirstQuery(): info.collide.cic.queries.SubQuery
+getLastQuery(): info.collide.cic.queries.SubQuery
info.collide.cic.queries.ComplexQNode
+model: info.collide.cic.queries.ComplexQuery
+getFirstQuery(): info.collide.cic.queries.SubQuery
+getLastQuery(): info.collide.cic.queries.SubQuery
EditSubqueries
EditDescription
SetImageAction
info.collide.cic.queries.ConstantQNode
+model: info.collide.cic.queries.ConstantQuery
+textArea: javax.swing.JTextArea
+getFirstQuery(): info.collide.cic.queries.SubQuery
+getLastQuery(): info.collide.cic.queries.SubQuery

Figure A.3: Simplified Class Diagram for Queries

A.2 Query Reference Frame Implementation Details

A.2.1 Query Resource Provider

The methods available for all queries through the `QueryResourceProvider` are the following:

1. `public org.w3c.dom.Document getCurrentDocument();`

   This method returns the current document in the current context. This method is implemented in the Student and Teacher applications, but in the Repository, it always returns the `null` value.

2. `public void error(String s, Exception e);`

   This method logs errors and sends them to a central location where they can be handled.

3. `public java.util.Vector getStudentHandles();`

   This method returns a `Vector` of `StudentHandle`s inside the Teacher application, one for each student connected to the current CiCv2 session. In other contexts the method just returns the `null` object.

4. `public java.io.FileReader getFileReader(java.io.File file)`

   This method is used whenever a query intends to open a file for reading. Since the CiCv2 uses a virtual file system that gets mapped to different parts of the local filesystem and the repository, it is necessary to perform the translations from virtual names to the corresponding local files. Also, before the file is opened, it may have to be downloaded from the repository, which is also done in the Teacher and Student applications.

5. `public java.io.FileWriter getFileWriter(java.io.File file, boolean append)`

   This method is similar to `getFileReader( ... )`, but it opens the file in write mode. The `boolean` argument determines whether a potentially existing file gets overwritten or appended to.

6. `public void uploadFile(java.io.File file)`

   After having opened a file using the `getFileWriter( ... )` method, the file has to be uploaded to the repository in order to maintain the consistency of the file.

7. `public void setGraphAssociations(info.collide.graph.JGraph graph);`

   This method is specific to the CiCv2 implementation. Since the visual languages are not defined in advance, it is necessary to add associations that will allow the resolution of XML constructs to their corresponding Java objects. This is important for the complex queries (section 6.5.5), which use a saved `JGraph` (see 5.5.1) to manage the sub-queries.

8. `public Object getPaletteHandler();`

   In the complex queries, just as the `setGraphAssociations( ... )` method, this method is used to set up the `JGraph` (see 5.5.1). Additionally, this method is needed for the also CiCv2-specific `ObjectCreationQuery` implementation.

9. `public CiCQueryResult executeRepositoryQuery(String subquery, java.util.Vector inputQueries, java.util.Vector inputQueriesLinkModel)`

   This method receives a query in its XML form, as a String named `subquery`. It sends the query to the repository where it is decoded and executed. The result of this execution is received as a `CiCQueryResult` object. The method then returns that result. In the repository implementation, this method simply executes the query, since it already is located in the repository context.

10. `public java.io.File getCiCLog();`

    This method returns a file to read the CiCv2 actions log, which is written in XML form. In the repository implementation, it returns the `null` object (see 5.7).

11. `public java.io.File getStdLog();`

    As the previous method, this one returns an open file referencing the Standard and Error output of the Teacher or Student application (see 5.7).

12. `public java.io.File getMmLog(String name);`

    Whenever an application creates a Matchmaker collaboration, the Teacher application opens a logfile in which all actions related to that Matchmaker collaboration are logged (see 5.7).

Since the logging is performed on the Teacher application, where the MatchMaker server resides, this method returns the `null` object in the Student application and repository.

13. `public String getMmLogs();`

    This method returns a list of all currently available MatchMaker collaborations. As in the previous method, it only makes sense in the Teacher application and returns the `null` object in both the repository and the Student application.

A.3 Basic building blocks for queries

The implementation of several atomic queries is described in this section, following the model presented in 6.3.1.

A.3.1 CiC Query

info.collide.cic.queries.CiCQuery is the root class for all implemented queries in this thesis. It is an abstract class that implements some useful methods used in the subclasses. This class has no functionality in itself and it is not used as a query.

When a query is executed, the result is expected to be displayed in some way to the user. Afterwards, the same query can be executed again and the behavior in this case depends on the boolean variable defined in the `CiCQuery` class: `rememberPrevious`. If the variable has value of `true`, the previously displayed result, if it still exists, is replaced by the newly calculated result. If the variable, on the contrary, had been set to `false`, then a new result is displayed along with the previous one, allowing the teacher to compare them.

Each query also sets another variable: `maxInputQueries`, which is used by the classes implementing `QueryConnectionLink` to determine whether a new connection link will be allowed or not. This way it is avoided to connect an input to a query that will not make use of it, reducing the potential confusion.

A.3.2 Constant Query

This is the most basic form a query can take. It stores a predefined value (a `String`), and each time the query is executed, this predefined value is returned. This type of query has several uses, including:

- input of user provided information
- debugging of queries for special input
- inclusion of special values

A.3.3 Query Result

It has to be noted that the result of any query, an object of the class `CiCQueryResult` is returned containing the result. This result can be set up based on one of two types, using the following constructors:

- `public CiCQueryResult(java.lang.String result);`
- `public CiCQueryResult(org.w3c.dom.Node result);`

Accordingly, a Query Result can be retrieved using any of the following methods, independently of the way the result was defined:

- `public String asString()`: it returns the result as a String. If the Query Result was defined as a `Node`, a conversion is made to transform the Document Object Model (DOM) linked by the `Node` as a String.
- `public Node asNode()`: it returns root element (of type `Node`) of the Document Object Model (DOM). If the Query Result was defined as a `String`, that `String` is parsed as if it was a XML document. If the parsing fails, the String is used as the only Child of a root element that forms a new XML document.

Additionally, an object of type `CiCQueryResult` can be used as a query in itself; it will always return the same result just like an object of class `ConstantQuery`. The purpose of this feature is to allow the development of complex queries, storing any intermediate result that can then be used to continue modifying the query.

A.3.4 File Query

This query allows a File to be read. The filename argument can be provided in two different ways:

- Predefined Filename: it is a name that is defined before the execution of this query.
- Input from another query: using a `QueryConnectionLink`, a filename is returned from a query that is executed immediately before the File Query.

The `getFileReader( ... )` method from the `QueryResourceProvider` (cf. 6.4.1) is used for opening files. Some special filenames are defined, that have particular meanings:

- "ciclog:": the `getCiCLog()` method is used instead of the `getFileReader( ... )` method from the `QueryResourceProvider`. It returns a sequence of all events recorded in the CiCv2 actions log.

- “stdlog:”: the `getStdLog()` method is used instead of the `getFileReader( ...)` method from the `QueryResourceProvider`. It returns a sequence containing the standard and error output of the application.
- “mmlogs:”: the `getMmLogs()` method is used instead of the `getFileReader( ... )` method from the `QueryResourceProvider`. The returned value is a list of all the MatchMaker collaborations in the current session.
- “mmlog:`name`”: the `getMmLog(name)` method is used instead of the `getFileReader( ... )` method from the `QueryResourceProvider`. The `String name` is taken from the argument.

A.3.5 Current Document Query

An XML representation of the current Document is returned using the `getCurrentDocument()` method. This query makes sense only in the Student and Teacher Applications, which return the complete document that is currently opened. In the case of the repository, the invocation of the method would return the `null` object.

A.3.6 Multiplexion Query

A Multiplexion Query acts providing multiple inputs in one. For each input that is connected to a Multiplexion Query, a subtree in the XML output is created with the output of that particular query. In the example presented in figure A.4, the output generated is the one displayed in figure A.5.

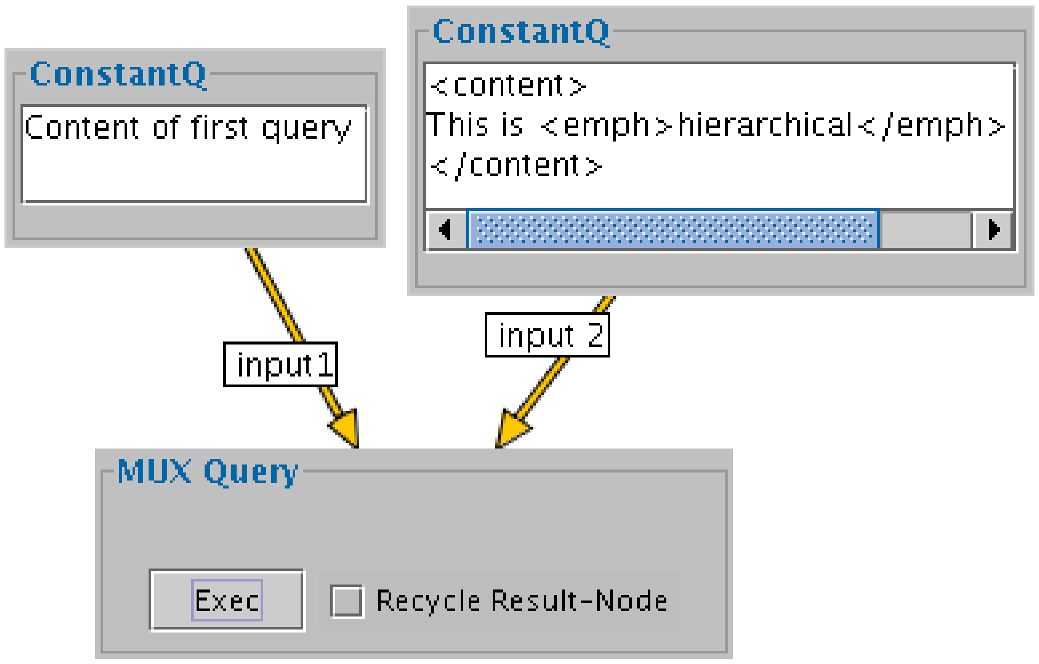


Figure A.4: Multiplexion Query Example

```
<MUXQuery>
  <CiCQuery label="input1">
    <CiCQueryResult>Content of first query</CiCQueryResult>
  </CiCQuery>
  <CiCQuery label="input 2">
    <content>
This is <emph>hierarchical</emph> content.
</content>
  </CiCQuery>
</MUXQuery>
```

Figure A.5: Result of MUX Query

A.3.7 Diff Query

The concept of difference between documents is widely used within the version management arena and Unix community (MacKenzie et al., 2003). The difference are expressed as the smallest number of defined (and possibly weighted) operations that, applied to the base document, transform it into the target document. This shortest list of operations is called edit distance (Tai, 1979). In several tasks such as programming, the basic structural element is a line. Based on this, the operations are defined on deletion, insertion and modification of complete lines or even groups of lines.

Generalizing this concept to structured documents (Hajiaghayi, 2001) or trees (Barnard et al., 1995), so as the Document Object Model (DOM) generated from XML documents, the operations are performed either on nodes or subtrees. The traditional calculation of difference can be seen as a structured document differencing where each line is one element of the structure. The result in this case would be a tree with depth 1 and with the root having one child for each line in the document. The generally used algorithm is the Longest Common Subsequence or LCS (Hirschberg, 1977), where the sequence can be of characters, nodes or items that present the desired granularity.

The nature of the operations also can be generalized to lead to a smaller edit distance between documents. For example, a movement operation can substitute the need to include two operations: one elimination followed by one insertion of the same content into a different part of the document. This operation requires the ability to detect the movement of nodes, leaves (Selkow, 1977) or subtrees (Lu, 1979) within the structure. It is clear that the usage of this movement operation can improve the edit distance between the documents. However, the complexity of determining the difference based on these operations increases computing time and memory requirements. It has been shown (Zhang et al., 1992) that change detection in unordered trees is NP-Complete in the general case. Some significant improvements can potentially be reached by keeping track of changes within the tool being used, or by using some heuristics that take advantage of special properties found in particular documents (Cobéna et al., 2002b; Wang et al., 2003). See also (Cobéna et al., 2002a) for a comparison of several systems computing changes in structured documents, particularly XML documents.

The Diff Query implementation is based on the work described by Wang et al. (2003), with some modifications that make the Diff Query easier to integrate with the other queries. In particular, the result of executing the X-Diff tool without modifications is not well-formed XML, so it cannot be used as a DOM in other queries.

The following base XML structure will be modified in several ways to show the result of the Diff Query execution. When no difference is found between the two inputs to the Diff Query, the result is the intact XML structure. The original XML structure is the following:

```
<file>
  <part number="1">
    sample text
    <title>Part 1</title>
    <filename>part1.xml</filename>
    <author>jharding</author>
    <length>6834</length>
  </part>
</file>
```

The X-Diff implementation detects the following changes to a XML structure:

- Deletion of a subtree. This event is made visible by enclosing the subtree that was deleted in the "destination" XML model inside a tag named *cic-diff-query-delete*, indicating the type (*subtree*) in a parameter. In the following example, the tag "filename" and its sub-nodes were removed from the original, the result is:

  ```
  <file>
  <part number="1">
  sample text
  <title>Part 1</title>
  ```

```
<filename>
      <cic-diff-query-delete name="filename" type="subtree"/>
part1.xml</filename>
<author>jharding</author>
<length>6834</length>
</part>
</file>
```

- Insertion of a new subtree. Same as above, but the inserted data is not enclosed inside new tag, but the subtree is pointed out adding a new tag *<cic-diff-query-insert type="subtree" tag="tagname">* where the tagname refers to the inserted subtree. When a new subtree, consisting of the XML fragment *<editor>jhp</editor>* after the tag *author*, the result is the following:

```
<file>
<part number="1">
sample text
<title>Part 1</title>
<filename>part1.xml</filename>
<author>jharding</author>
<length>6834</length>
<editor>
      <cic-diff-query-insert tag="editor" type="subtree"/>
jhp</editor>
</part>
</file>
```

- Deletion of a textnode. The removed text is shown inside a tag named *cic-diff-query-delete*, having argument *type* with valuetext. When removing the line containing the string *example text*, the following output is produced:

```
<file>
<part number="1">
```

```
    <cic-diff-query-delete type="text">sample text</cic-diff-query-delete>
<title>Part 1</title>
<filename>part1.xml</filename>
<author>jharding</author>
<length>6834</length>
</part>
</file>
```

- Insertion of a textnode. Identical to the deletion in the previous case, but in the inverse way. When adding the text ”sample text 2” after the tag *length*, the result is:

```
<file>
<part number="1">sample text<title>Part 1</title>
<filename>part1.xml</filename>
<author>jharding</author>
<length>6834</length>
<cic-diff-query-insert type="text">sample text 2</cic-diff-query-insert>
</part>
</file>
```

- Modification of a textnode. When changing the text inside the *length* tag from 6834 to 6835, the result is:

```
<file>
<part number="1">sample text<title>Part 1</title>
<filename>part1.xml</filename>
<author>jharding</author>
<length>6835<cic-diff-query-update type="text">
        <from>6834</from>
      </cic-diff-query-update>
    </length>
</part>
</file>
```

- Insertion of an attribute. When inserting an attribute named *unit* and value *kbytes* into the tag *length*, the result is:

```
<file>
<part number="1">sample text<title>Part 1</title>
<filename>part1.xml</filename>
<author>jharding</author>
\item Insertion of a textnode. Identical to the deletion in the previous case,
but in the inverse way. When adding the text "sample text 2" after the tag
\emph{length}, the result is:

\begin{verbatim}
<file>
<part number="1">sample text<title>Part 1</title>
<filename>part1.xml</filename>
<author>jharding</author>
<length>6834</length>
<cic-diff-query-insert type="text">sample text 2</cic-diff-query-insert>
</part>
</file>
```

- Modification of a textnode. When changing the text inside the *length* tag from 6834 to 6835, the result is:

```
<file>
<part number="1">sample text<title>Part 1</title>
<filename>part1.xml</filename>
<author>jharding</author>
<length>6835<cic-diff-query-update type="text">
        <from>6834</from>
      </cic-diff-query-update>
    </length>
</part>
```

```
</file>
```

- Insertion of an attribute. When inserting an attribute named *unit* and value *kbytes* into the tag *length*, the result is:

```
<file>
<part number="1">sample text<title>Part 1</title>
<filename>part1.xml</filename>
<author>jharding</author>
<length unit="kbytes">
      <cic-diff-query-insert name="unit" type="attribute"/>6834</length>
</part>
</file>
```

- Deletion of an attribute. When deleting the attribute *number* from the tag *part*, the result is:

```
<file>
<part number="1">
    <cic-diff-query-delete name="number" type="attribute"/>sample text<title>Part 
<filename>part1.xml</filename>
<author>jharding</author>
<length>6835<cic-diff-query-update type="text">
        <from>6834</from>
      </cic-diff-query-update>
    </length>
</part>
</file>
```

- Modification of an attribute. When modifying the attribute *number* inside the tag *part* from value *1* to value *2*, the result is:

```
<file>
<part number="2">
```

```
        <cic-diff-query-update name="number" type="attribute">
          <from>1</from>
        </cic-diff-query-update>sample text<title>Part 1</title>
    <filename>part1.xml</filename>
    <author>jharding</author>
    <length>6834</length>
    </part>
    </file>
```

The modified version of the X-Diff tool that is integrated as a Query into the framework always produces well-formed XML. This enables the use of XQuery and XPath expressions to further process the result, keeping the principle that the output of a CiCQuery should always be parseable XML or XML fragment.

A.3.8 XQuery

The implementation of the XQuery needs a text that includes the XQuery expression, and a XQuery processing engine. It is possible to choose from several query processing engines, so the development is not limited to the specifics of one implementation. This allows to test the same expression using different implementations, comparing the results. It has proven to be helpful in the process of defining the queries, since some engines provide better feedback for finding errors in the expressions than the others. Once the expressions are final, however, the results have been equivalent using all the involved engines.

Currently, the XQuery engines used in the XQuery implementation are the following ones:

- *IPSI-XQ* (Fankhauser and Lehti, 2003), Fraunhofer Institut
- *Saxon* (Kay, 2005)

- *Qiz/Open* (Franc, 2005)

A.3.9 Context Changes

As mentioned, any query, and complex queries in particular, can be executed in any context: either in the Student application, Teacher application or repository. Making use of this, two queries have been defined that, when executed, previously change the context in which the contained queries are to be executed. The query is transported over the network to another location and executed remotely.

Limiting changes of context exclusively to special Complex Queries diminishes the potential confusion the context changes could imply. It is clear that all of the queries displayed in the same graphical container, and connected with each other using edges, are to be executed in the same context.

Repository Query

The Repository Query is executed in the context of the Teacher application. When executing, this query creates an XML representation of the grouped queries it contains, and it sends this representation to the repository along with a serialized version of the input that the Repository Query might have attached. The XML representation is transformed into the corresponding `CiCQuery` objects and the execution is performed in the same way a plain `ComplexQuery` would be executed, with the exception that the `QueryResourceProvider` given to each subquery is the Repository instead of the Teacher Application. This way, the same query is executed, but in another context, which leads to a different result.

Student Query

In the same way the repository query works, the Student Query sends the grouped sub-queries to a student in order to be executed in that context. This procedure is executed for every connected student, and the results are grouped together in an XML structure that combines each result into its own subtree.

As an example, the result of executing a StudentQuery in which each student returns a document fragment consisting of a single XML tag `<a/>` is shown below. In this case, two students are active in the classroom session, with usernames `pedro` and `jose`.

```
<StudentQueryResult>
  <Student username="pedro">
    <a/>
  </Student>
  <Student username="jose">
    <a/>
  </Student>
</StudentQueryResult>
```

In a normal CiCv2 setting, each Student Application is executed in a separate computer, so the execution of a Student Query is really done in a distributed computing environment.

However, simply executing the query for each student, waiting for the results to arrive before sending the query to the next student makes little use of that distributed environment. In order to improve the result, it is necessary that the Teacher application first instructs all of the Student applications to start processing the query and collects the results afterwards. This way the processing gets executed in parallel rather than in sequential order.

To solve this situation, a new `Thread` is created for each student application registered in the current session. Each thread requests the execution of the subquery for the particular student it is tied to, and it waits for the result. The Teacher Application collects the results as they arrive, and it has a limit of time during which the results may arrive. When the remaining time is over, the result is returned, ignoring any pending results. This way, the result of the query is not blocked by a problem of one student application, like a network interruption, high workload or any other difficulty. It also means that the result of a StudentQuery may not include the details for every student registered in the session.

A.3.10 Save Query

This query opens a file as specified by one of its properties. The file name can be appended with the current date, so that the repeated execution of the query does not overwrite previous results. Also, this query can optionally deliver the input it receives as its output, so the result gets saved into a file but it is also used as input to the next query. As a safety measure, the query can be set to not overwrite an existing filename, generating an error message instead.

A.3.11 Object Creation Query

The *Object Creation Query* implemented for the CiCv2 queries has two basic modes of operation, with a default fall-back mode. The selection of the modes depends on the content that is given as input to this query. The first mode triggers the creation of new pages which can be either appended to the existing document, or replace the existing document in the Teacher Application. Using the second mode of operation, the XML result is parsed as if it were a group of nodes. When the parsing and instantiation of the identified objects is successful, the newly created nodes are added to the current page of the document opened in the Teacher Application. When neither of the modes described above completes successfully, the *Object Creation Query* falls back to a safe mode that acts as most of the other queries by returning a *Query Result* with its associated graphical representation.

Generally, either an *Object Creation Query* or a *Save Query* (described above) will be found as the last query in any complex query. These queries are in charge of handling the final result of a query. In the case of the *Object Creation Query*, it will present the information to the user in a form other than a plain XML listing. On the other hand, a *Save Query* would store the information in a file for later use, perhaps in the form of a FreeStyler XML document.

Nevertheless, in some cases an *Object Creation Query* or *Save Query* is not the last but the penultimate query, followed only by a *Timed Query* as described below. In that case, the *Timed Query* will not receive any input through its connection, since the query before it already made the final actions. In this case, the connection between these queries only uses the control flow shown in figure 6.2.

A.3.12 Timed Query

A Timed Query can trigger the execution of the queries it is connected to in three possible ways:

1. manually, when the teacher presses the corresponding button
2. periodically, indicating the frequency at which the query is to be repeated
3. on a programmed time of the present day

It is also possible to combine the periodical querying and programmed time. In that case, the query will be executed for the first time on the set time, and from then on, it will be repeated with the defined frequency.

A.4 XQuery Implementation for Java Programming Scenario

The XQuery in part (b) of figure 7.2 on page 127 contains following text:

```
<AnnotationNode FontSize="10" creationtime="1110487750862" creator="jhp"
    lastModificator="none" package="info.collide.plugins.mindmap"
    sticky="false" uiLocked="false" EditorType="JEditorPane">
    <Content>
    {let $students := input()//Student/result
    let $ret := concat("<HTML border='1'><table><tr><th>Student</th>",
                      "<th>Page 1</th><th>Page 2</th><th>Page 3</th>",
                      "<th>Page 4</th><th>Page 5</th></tr>")
    return $ret}
    {
for $student in input()//Student
    let $un := string($student/@username)
    let $st := for $td in $student/tr/td
        let $p := $td/child::text()
    return concat("<td>", $p, "</td>")
    return concat("<tr><td>", $un, "</td>", $st,
        "</tr>")
    }
    {"</table></HTML>"} </Content>
</AnnotationNode>
```

The XQuery in part (c) of figure 7.2 contains following text:

```
let $pages := input()/DocumentRoot/SessionData/Workspaces/Workspace
let $p := $pages[position() = 1]//JavaNode/Content/child::text()
let $p1 := if (contains($p, ".createSession("))
                  then
                      if (contains($p, ".inSession()"))
then "create + Ver"
else "createOnly"
                  else "none"
let $p := $pages[position() = 2]//JavaNode/Content/child::text()
let $p2 := if (contains($p, ".getSessions()"))
                  then
                      if (contains($p, "println("))
                      then "get + print"
                      else "getOnly"
                  else "none"
let $p := $pages[position() = 3]//JavaNode/Content/child::text()
let $p3 := if (contains($p, ".joinSession("))
                  then
                      if (contains($p, ".getSessions()"))
                          then "get + join"
                          else "joinOnly"
                  else "none"
let $p := $pages[position() = 4]//JavaNode/Content/child::text()
let $p4 := if (contains($p, ".joinSession("))
                  then
                      if (contains($p, ".createObject(")
                           or contains($p, ".changeObject(")
                           or contains($p, ".deleteObject(")
                           or contains($p, ".execAction(")
                          )
                          then "join + mod"
                          else "joinOnly"
                  else "none"
let $p := $pages[position() = 5]//JavaNode/Content/child::text()
let $p5 := if (contains($p, ".readSyncTree()"))
                  then
                      if (contains($p, "println("))
                      then "read + print"
                      else "getOnly"
                  else "none"

return <tr><td>{$p1}</td><td>{$p2}</td><td>{$p3}</td>
            <td>{$p4}</td><td>{$p5}</td></tr>
```

Appendix B

Examples

B.1 Definition and Validation of Best Practices

It is not really possible to separate definition and validation of Best Practices, since they form part of one process that includes several iterations of definition-validation. Considering the example presented in 4.2, following queries are proposed to assist a teacher during the definition and validation process:

1. **Number (or, less summarized, the list) of students that have opened the document suggested by the teacher:** during the initial presentation, it is likely that the teacher suggests the students to open a document. This would contain the same material that the teacher is showing, or it may contain material to complement the presentation.
2. **Number of changes the students have made to the current document:** if the students are making a lot of changes to the currently opened document, they might be interested in whatever it is they are working on. Using the outcome of the previous query it should be possible to discover if the students are using the suggested documents, or rather are engaging

in activities that have no relation with the current subject. There is no guarantee that this query will effectively show the actual occurrence, but it may be used as a clue.

3. **Show the page that student "Juan Pérez" is currently modifying:** in many situations it will be useful to show the teacher the page a student is currently working on. It can be seen as an equivalent of walking around the classroom and look at the work the students are performing, with some important differences. For instance, normally the student would have no indication that the teacher is watching her work, and can not avoid the situation. This has obvious privacy implications, that can be dealt with in the system for example by allowing a student to block a page from being accessed by the teacher. This query can be applied if the teacher wants the student to show her work to the class or needs to verify if the meaning of the previous queries is really what she is interpreting.

4. **Determine if student "Juan Pérez" has already worked on exercise 6:** a teacher can see the progress of a particular student by looking at how many exercises have been tackled by him or her. This does not include an indication of how successful the student has resolved the problems or if they are solved at all, but it can be used to interpret several aspects. Together with other queries it is possible to infer some possible situations, such as:

   - *excessive difficulty of some problems* (or *lack of preparation on behalf of the students* to solve them) if they are being started by many students but not finished or incorrectly solved by most of them
   - *lack of interest*, *hyperactivity* or some other issue that might require the teacher's attention if a student starts working on the problems in no particular order, leaving most unfinished
   - *unchallenged students* if they have completed successfully most of the assigned exercises, more if some start working on exercises that are supposed to be tackled later on

5. **Evaluate how many correct answers does student "Juan Pérez" have in the proposed multiple choice quiz:** in the case of a multiple choice quiz, it is relatively easy to determine correct answers automatically. This is a common tool to quickly evaluate a rough estimate of the students understanding, and can help both on its own and combined with other information for the teacher to take the right decisions inside the classroom.

6. **Perform query A for each active student:** some of the above queries have been stated referring to a particular student. Using them in this case would result in an overview of the results for each students. Possibly this information is not useful if presented in this form to the teacher, but it may be used to summarize the information of the students as a group, as explained in the next example.

7. **Summarize information for query B:** in several occasions a teacher needs less specific information, by means of summarizing data into a table, calculate averages, maximum and minimum values, gaining a wider view of the situation. This also includes filtering, for example, selecting all the students that have finished all of the exercises would require to perform a query on each active student and then show only the names of the students that concern in this case.

8. **Show groups of students interacting through collaboration mechanisms:** a teacher can see when students interact in a face-to-face situation, but they cannot know directly about the interactions occurring through the use of technology. In the case of the CiC, students may use shared workspaces, and in group work they are encouraged to do so. When working in groups, a teacher would like to identify the groups that are sharing workspaces, having this way access to the information to see if the students working in a designated group are interacting this way and if some students are left out of this kind of interaction.

The results of these queries provide useful information, but in most cases it is still necessary to give them a meaning before they can be used. This is the work to be performed in the stages of definition and validation of best practices. The relevant queries have to be defined, and their result needs some guidelines for interpretation. This happens through the process of validation, where proposed queries are challenged, refined and complemented with new queries that allow to disambiguate situations in which more than one interpretation is possible.

B.2 Application of Best Practices

Once the interpretation and the relevance of the queries is clear, the best practices can be applied. A set of recommendations contained in the best practices can have several levels of completeness. In the best case, recommendations exist regarding the possible actions a teacher can take in the possible situations inside the classroom. For example, if there is an indication that the students did not understand some part of the subject treated in the session, a teacher should decide to review that part in depth before continuing with the activities as suggested in the best practices. If the best practices are complete for this particular subject, they would consider that case and the courseware may include some material that can be used to review that part of the subject.

It is possible not only to verify if the best practices are being applied but also, and most importantly, if the expected outcomes are being achieved. This is the ultimate goal of the presented system. Moreover, a teacher should be able to interpret the suggested queries and act upon the results to guide the students towards the completion of the pedagogic goals. Thus, the courseware used by the teachers should include suggested queries and an explanation on how to interpret them, so the teachers can add the technological help to their own skills and take the most advantage out of the classroom environment.

Appendix C

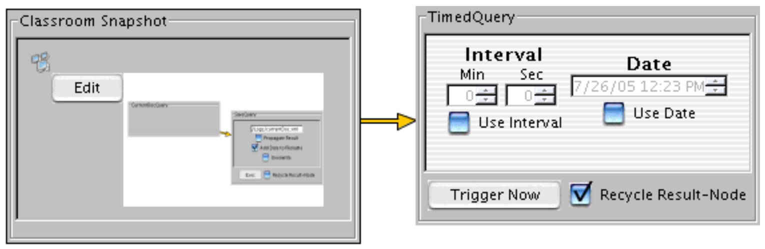
Implementation of sample Queries

The detailed implementation of several queries is shown below. In cases where the content of several subqueries is not visible, it has been transcribed, indicating the type and number of the query. The order in which the transcription is made corresponds to the order of the data flow as described in 6.2.

C.1 “Classroom Snapshot” Implementation

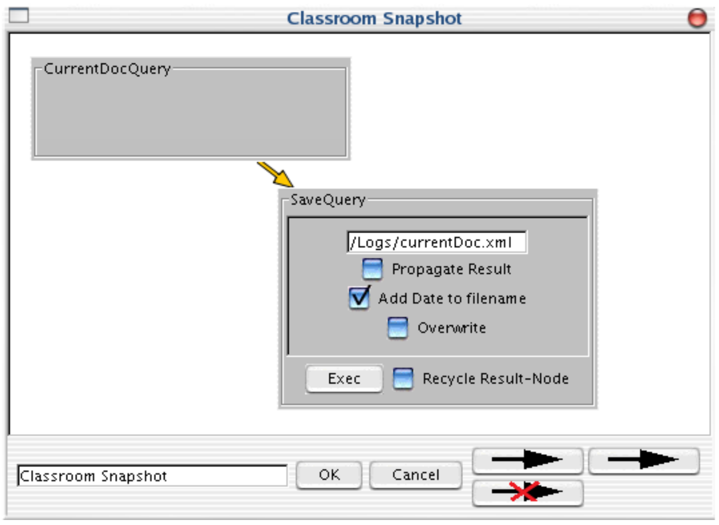
The *Classroom Snapshot* query consists of two visible queries (as seen in figure C.1 (a)), connected by an edge that does not need to have a specific label. The first query (in the information flow order) is a Student Query that saves the current document in each student application. This query always delivers an empty result, since the desired effect of the query is the creation of a file on the repository for each active student. The second query is a *TimeQuery*, and triggers the execution of the previous query at a defined interval, at a fixed date or at a defined interval but starting after a fixed date.

(a)



0
0
☐ Use Interval
7/26/05 12:23 PM
☐ Use Date
✓ Recycle Result-Node

(b)



/Logs/currentDoc.xml
☐ Propagate Result
✓ Add Date to filename
☐ Overwrite
☐ Recycle Result-Node
Classroom Snapshot

Figure C.1: "Classroom Snapshot" Implementation

The first query, a *StudentQuery* and thus a *ComplexQuery*, consists of two subqueries (see figure C.1 (b)). The first of them is a *CurrentDocumentQuery*, and its output is sent to a *Save-Query* that has its base filename set as `/Logs/currentDoc.xml`. Since the option “Add Date to Filename” is set, the filename gets the date added between `currentDoc` and `.xml`, resulting in a filename of the form *currentDoc-YYYY-MM-DD-HH-MM-SS.xml*, where `YYYY`, `MM`, `DD`, `HH`, `MM` and `SS` are filled with the respective values of the current year, month, day, hour of day, minute and second.

C.2 “List Students” Implementation

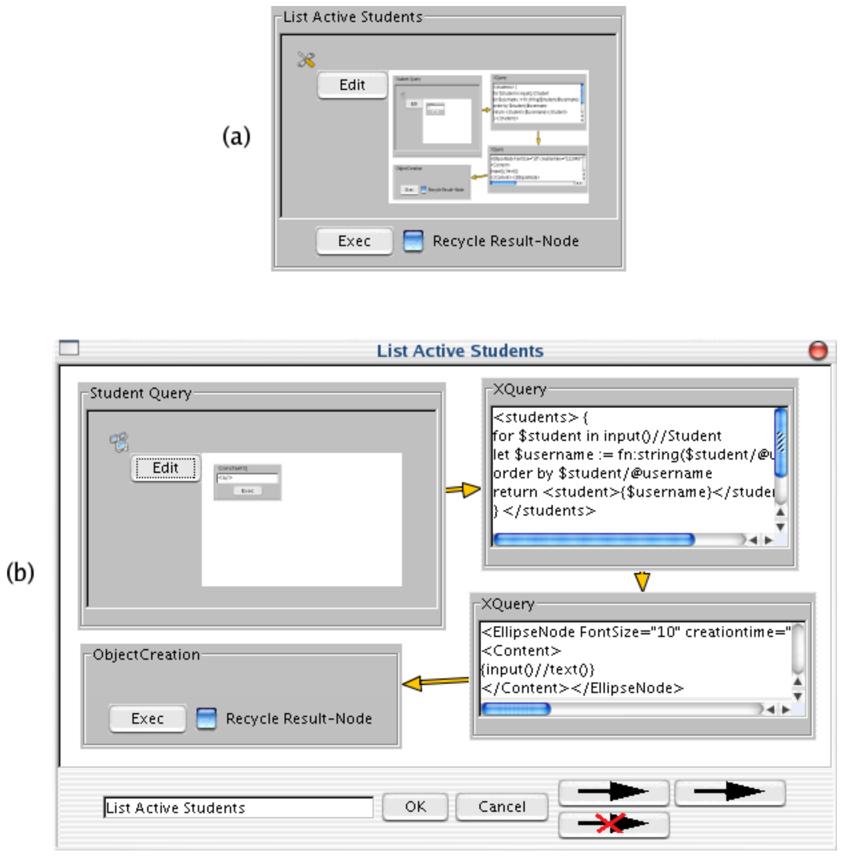
The “List Students” query executes a trivial remote query on each student application, receiving a list of all currently active students. Figure C.2 shows two parts. Part (a) is the query as presented to the teacher, and part (b) is the detail of the complex query showing its subqueries.

C.2.1 XQuery 1 of part (b)

```
<students> {
for $student in input()//Student
let $username := fn:string($student/@username)
order by $student/@username
return <student>{$username}</student>
} </students>
```

C.2.2 XQuery 2 of part (b)

```
<EllipseNode FontSize="10" creationtime="1110487750862" creator="jhp"
```



```
<students> {
for $student in input()//Student
let $username := fn:string($student/@u
order by $student/@username
return <student>{$username}</studer
} </students>
```

```
<EllipseNode FontSize="10" creationtime="
<Content>
{input()//text()}
</Content></EllipseNode>
```

Figure C.2: “List Students” Implementation

```
    lastModificator="none" package="info.collide.plugins.mindmap"
    sticky="false" uiLocked="false">
<Content>
{input()//text()}
</Content></EllipseNode>
```

C.3 “Current Page from Student” Implementation

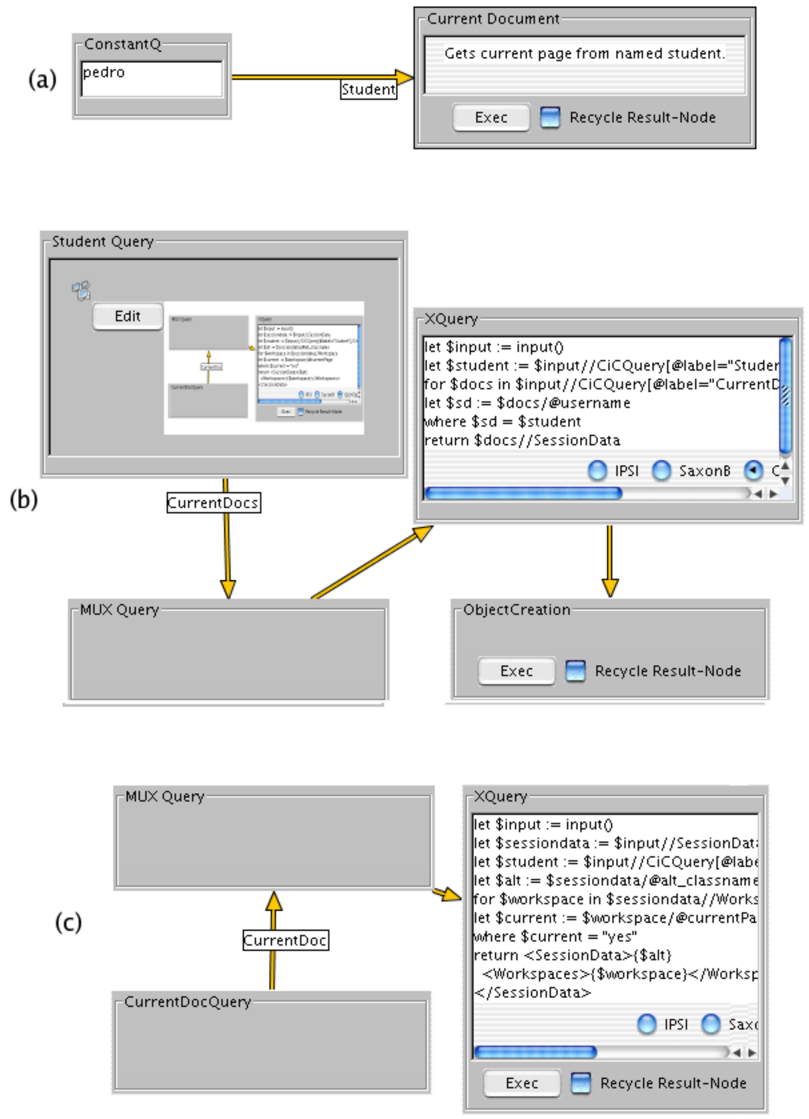
Figure C.3 shows the three parts of the “Current Page from Student” implementation. Part (a) is the query as presented to the teacher, part (b) is the detail of the subqueries corresponding to the complex query labeled “Current Document” in part (a), and part (c) corresponds to the subqueries of the query labeled “Student Query” in part (b).

C.3.1 XQuery of part (b)

```
let $input := input()
let $student :=
    $input//CiCQuery[@label="Student"]/CiCQueryResult/child::text()
for $docs in $input//CiCQuery[@label="CurrentDocs"]//Student
let $sd := $docs/@username
where $sd = $student
return $docs//SessionData
```

C.3.2 XQuery of part (c)

```
let $input := input()
```



Gets current page from named student.

```
let $input := input()
let $student := $input//CiCQuery[@label="Studer
for $docs in $input//CiCQuery[@label="CurrentD
let $sd := $docs/@username
where $sd = $student
return $docs//SessionData
```

```
let $input := input()
let $sessiondata := $input//SessionDat
let $student := $input//CiCQuery[@labe
let $alt := $sessiondata/@alt_classname
for $workspace in $sessiondata//Works
let $current := $workspace/@currentPa
where $current = "yes"
return <SessionData>{$alt}
 <Workspaces>{$workspace}</Worksp
</SessionData>
```

Figure C.3: “Current Page from Student” Implementation

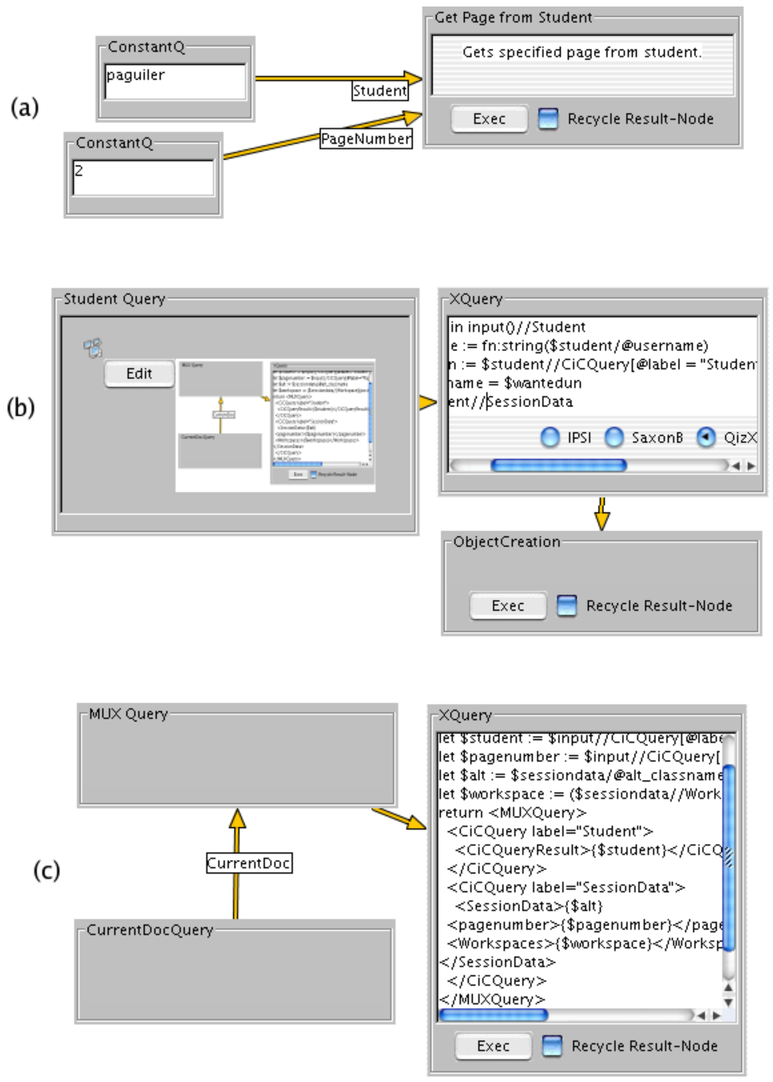
```
let $sessiondata := $input//SessionData
let $student :=
    $input//CiCQuery[@label="Student"]/CiCQueryResult/child::text()
let $alt := $sessiondata/@alt_classname
for $workspace in $sessiondata//Workspace
let $current := $workspace/@currentPage
where $current = "yes"
return <SessionData>{$alt}
  <Workspaces>{$workspace}</Workspaces>
</SessionData>
```

C.4 “Specified Page from Student” Implementation

Figure C.4 shows the three parts of the “Specified Page from Student” implementation. Part (a) is the query as presented to the teacher, including the two arguments a teacher will typically manipulate. Part (b) is the detail of the subqueries corresponding to the complex query labeled “Current Document” in part (a), and part (c) corresponds to the subqueries of the query labeled “Student Query” in part (b).

C.4.1 XQuery of part (b)

```
for $student in input()//Student
let $username := fn:string($student/@username)
let $wantedun := $student//CiCQuery[@label =
    "Student"]/CiCQueryResult/child::text()
where $username = $wantedun
return $student//SessionData
```



```
in input()//Student
e := fn:string($student/@username)
n := $student//CiCQuery[@label = "Studen
name = $wantedun
ent//SessionData
```

```
let $student := $input//CiCQuery[@labe
let $pagenumber := $input//CiCQuery[
let $alt := $sessiondata/@alt_classname
let $workspace := ($sessiondata//Work
return <MUXQuery>
 <CiCQuery label="Student">
  <CiCQueryResult>{$student}</CiCQ
 </CiCQuery>
 <CiCQuery label="SessionData">
  <SessionData>{$alt}
 <pagenumber>{$pagenumber}</page
 <Workspaces>{$workspace}</Worksp
</SessionData>
 </CiCQuery>
</MUXQuery>
```

Figure C.4: “Specified Page from Student” Implementation

C.4.2 XQuery of part (c)

```
let $input := input()
let $sessiondata := $input//SessionData
let $student :=
    $input//CiCQuery[@label="Student"]/CiCQueryResult/child::text()
let $pagenumber :=
    $input//CiCQuery[@label="PageNumber"]/CiCQueryResult/child::text()
let $alt := $sessiondata/@alt_classname
let $workspace := ($sessiondata//Workspace)[position() = $pagenumber]
return <MUXQuery>
  <CiCQuery label="Student">
    <CiCQueryResult>{$student}</CiCQueryResult>
  </CiCQuery>
  <CiCQuery label="SessionData">
    <SessionData>{$alt}
  <pagenumber>{$pagenumber}</pagenumber>
  <Workspaces>{$workspace}</Workspaces>
</SessionData>
  </CiCQuery>
</MUXQuery>
```

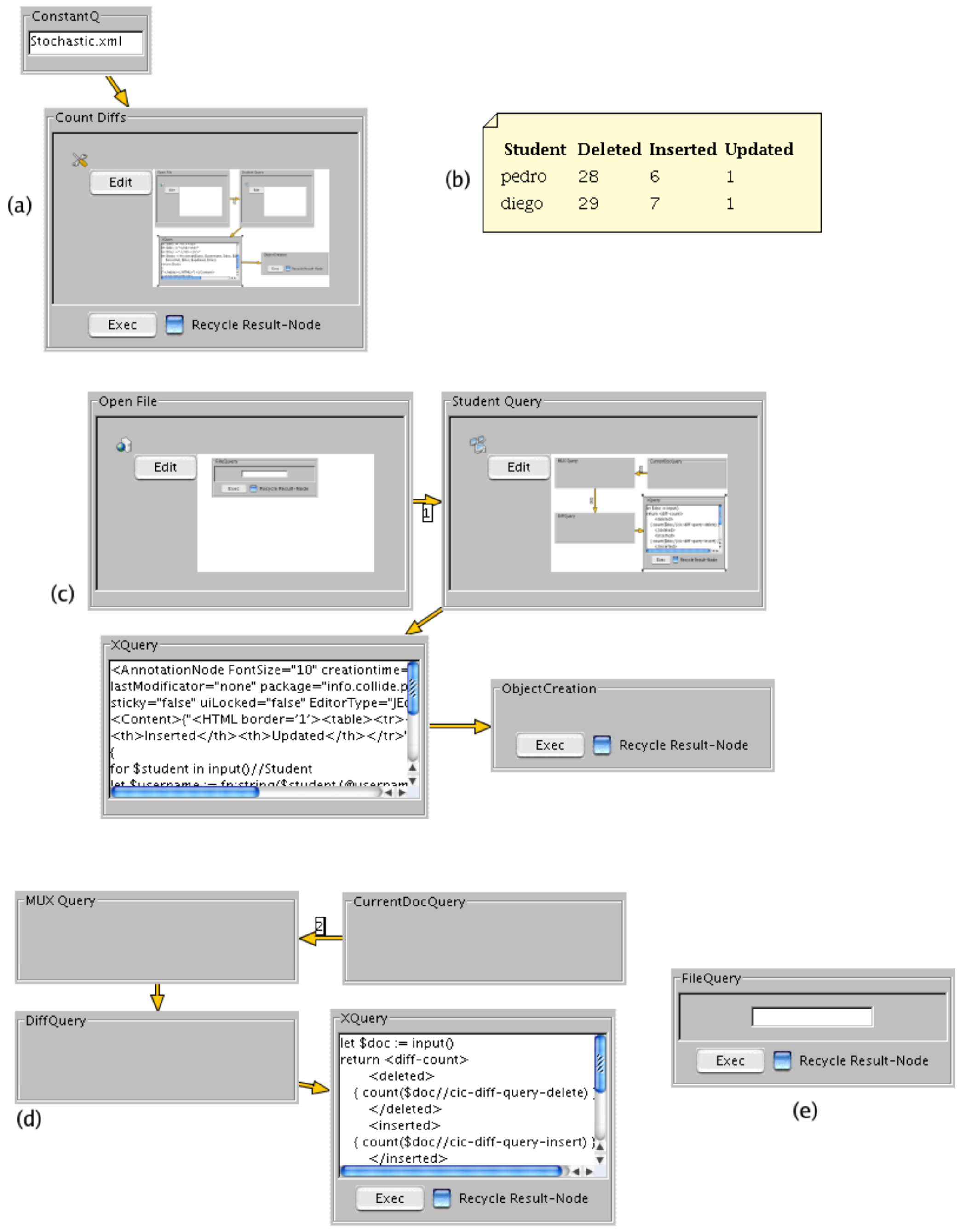
C.5 “Diff Count” Implementation

The “Diff Count” query, as presented in figure C.5, is presented in five parts including a sample result. Part (a) is the visible part for a teacher, in which the filename is given as a parameter to a complex query. Part (b) shows a sample result of executing this query.

The complex query of part (a) is built of the subqueries presented in part (c). The first query in that part consists of a Repository query, which reads the specified file and returns its XML structure. The content of this Repository Query is a simple FileQuery, as displayed in part (e). Part (d) is the query executed for each student through the StudentQuery of part (c).

C.5.1 XQuery of part (c)

```
<AnnotationNode FontSize="10" creationtime="1110487750862" creator="jhp"
lastModificator="none" package="info.collide.plugins.mindmap"
sticky="false" uiLocked="false" EditorType="JEditorPane">
<Content>{"<HTML border='1'><table><tr><th>Student</th><th>Deleted</th>
<th>Inserted</th><th>Updated</th></tr>"}
{
for $student in input()//Student
let $username := fn:string($student/@username)
let $diff-count := $student/diff-count
let $deleted := $diff-count/deleted/child::text()
let $inserted := $diff-count/inserted/child::text()
let $updated := $diff-count/updated/child::text()
let $uno := "<tr><td>"
let $dos := "</td><td>"
let $tres := "</td></tr>"
let $todo := fn:concat($uno, $username, $dos, $deleted, $dos,
     $inserted, $dos, $updated, $tres)
return $todo
}
{"</table></HTML>"} </Content>
</AnnotationNode>
```



Student	Deleted	Inserted	Updated
pedro	28	6	1
diego	29	7	1

```
<AnnotationNode FontSize="10" creationtime=
lastModificator="none" package="info.collide.p
sticky="false" uiLocked="false" EditorType="JE
<Content>{"<HTML border='1'><table><tr>
<th>Inserted</th><th>Updated</th></tr>'
{
for $student in input()//Student
let $username := fn:string($student/@usernam
```

```
let $doc := input()
return <diff-count>
      <deleted>
   { count($doc//cic-diff-query-delete)
      </deleted>
      <inserted>
   { count($doc//cic-diff-query-insert) }
      </inserted>
```

Figure C.5: “Diff Count” Implementation

C.5.2 XQuery of part (d)

```
let $doc := input()
return <diff-count>
        <deleted>
    { count($doc//cic-diff-query-delete) }
        </deleted>
        <inserted>
    { count($doc//cic-diff-query-insert) }
        </inserted>
        <updated>
    { count($doc//cic-diff-query-update) }
        </updated>
    </diff-count>
```

C.5.3 Sample output of part (d)

```
<StudentQueryResult>
  <Student username="pedro">
    <diff-count>
  <deleted>28</deleted>
  <inserted>6</inserted>
  <updated>1</updated>
</diff-count>
  </Student>
  <Student username="diego">
    <diff-count>
  <deleted>29</deleted>
  <inserted>7</inserted>
  <updated>1</updated>
```

```
</diff-count>
  </Student>
</StudentQueryResult>
```

C.6 "Opened File" Implementation

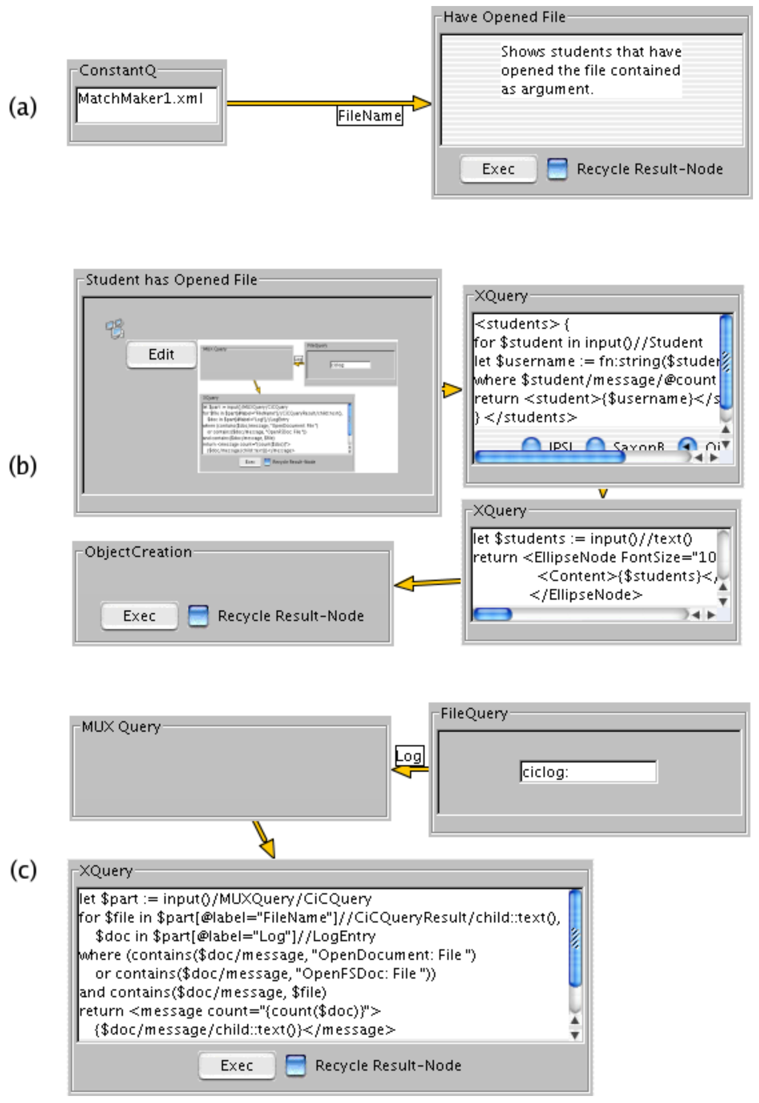
The "Opened File" query shows a list of students that have opened a specified file. In figure C.6, Part (a) shows the query as visible by the teacher. Part (b) shows the detail of the subqueries in the "Have Opened File" query of part (a). Part (c) shows the detail of the subqueries in the "Student has Opened File" query of part (b).

C.6.1 XQuery 1 of part (b)

```
<students> {
for $student in input()//Student
let $username := fn:string($student/@username)
where $student/message/@count > 0
return <student>{$username}</student>
} </students>
```

C.6.2 XQuery 2 of part (b)

```
let $students := input()//text()
return <EllipseNode FontSize="10" creationtime="1110487750862" creator="jhp"
    lastModificator="none" package="info.collide.plugins.mindmap"
```



```
<students> {
for $student in input()//Student
let $username := fn:string($studer
where $student/message/@count
return <student>{$username}</s
} </students>
```

```
let $students := input()//text()
return <EllipseNode FontSize="10
            <Content>{$students}</
          </EllipseNode>
```

```
let $part := input()/MUXQuery/CiCQuery
for $file in $part[@label="FileName"]//CiCQueryResult/child::text(),
   $doc in $part[@label="Log"]//LogEntry
where (contains($doc/message, "OpenDocument: File ")
   or contains($doc/message, "OpenFSDoc: File "))
and contains($doc/message, $file)
return <message count="{count($doc)}">
   {$doc/message/child::text()}</message>
```

Figure C.6: “Opened File” Implementation

```
    sticky="false" uiLocked="false">
<Content>{$students}</Content>
</EllipseNode>
```

C.6.3 XQuery of part (c)

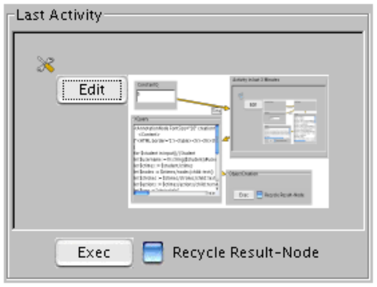
```
let $part := input()/MUXQuery/CiCQuery
for $file in $part[@label="FileName"]//CiCQueryResult/child::text(),
    $doc in $part[@label="Log"]//LogEntry
where (contains($doc/message, "OpenDocument: File ")
      or contains($doc/message, "OpenFSDoc: File "))
and contains($doc/message, $file)
return <message count="{count($doc)}">{$doc/message/child::text()}</message>
```

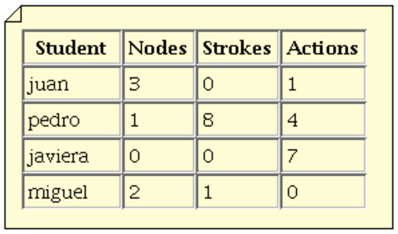
C.7 “Last Activity” Implementation

Figure C.7 shows the implementation of “Last Activity” query as visible by the teacher, along with a sample result of the execution of said query. In figure C.8 we see the detail of the subqueries in the query presented in figure C.7 (part (a)) and the detail of the query “Activity in last X Minutes” (part (b)).

C.7.1 XQuery of part 1

```
<AnnotationNode FontSize="10" creationtime="1110487750862" creator="jhp"
    lastModificator="none" package="info.collide.plugins.mindmap"
    sticky="false" uiLocked="false" EditorType="JEditorPane">
```

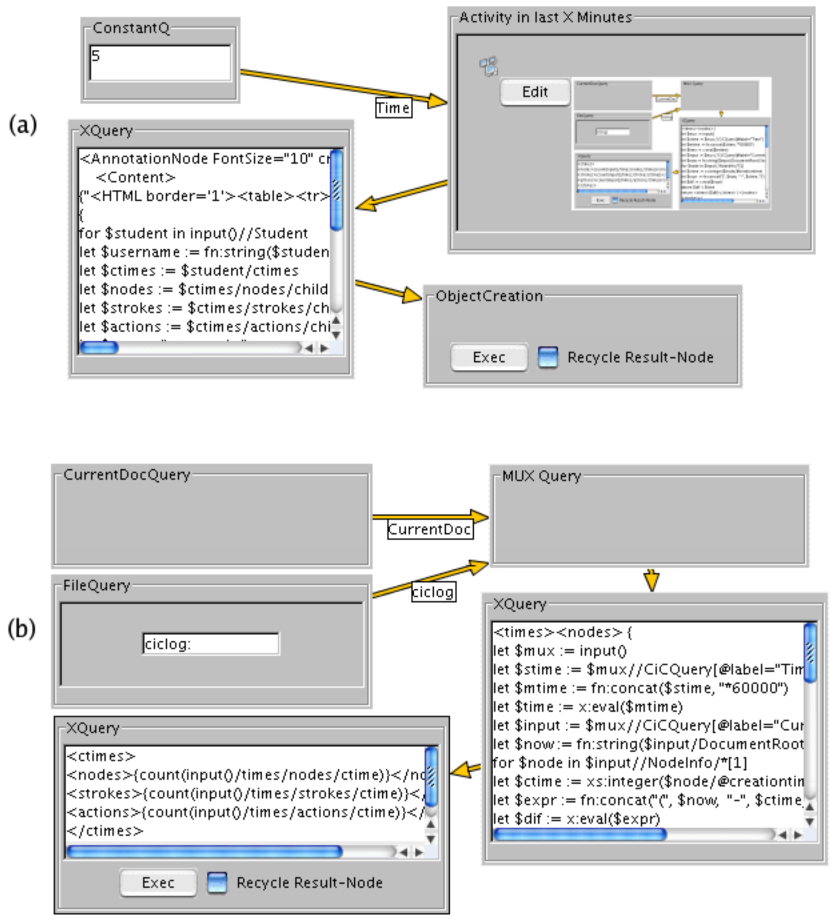




Student	Nodes	Strokes	Actions
juan	3	0	1
pedro	1	8	4
javiera	0	0	7
miguel	2	1	0

Figure C.7: “Last Activity” Implementation, part 1

```
    <Content>
{"<HTML border='1'><table><tr><th>Student</th><th>Nodes</th>
  <th>Strokes</th><th>Actions</th></tr>"}
{
for $student in input()//Student
let $username := fn:string($student/@username)
let $ctimes := $student/ctimes
let $nodes := $ctimes/nodes/child::text()
let $strokes := $ctimes/strokes/child::text()
let $actions := $ctimes/actions/child::text()
let $uno := "<tr><td>"
let $dos := "</td><td>"
let $tres := "</td></tr>"
let $todo := fn:concat($uno, $username, $dos, $nodes, $dos, $strokes, $dos,
    $actions, $tres)
return $todo
}
```



```
<AnnotationNode FontSize="10" cr
   <Content>
{"<HTML border='1'><table><tr>
{
for $student in input()//Student
let $username := fn:string($studen
let $ctimes := $student/ctimes
let $nodes := $ctimes/nodes/child
let $strokes := $ctimes/strokes/ch
let $actions := $ctimes/actions/chi
```

```
<times><nodes> {
let $mux := input()
let $stime := $mux//CiCQuery[@label="Tim
let $mtime := fn:concat($stime, "*60000")
let $time := x:eval($mtime)
let $input := $mux//CiCQuery[@label="Cur
let $now := fn:string($input/DocumentRoot
for $node in $input//NodeInfo/*[1]
let $ctime := xs:integer($node/@creationtin
let $expr := fn:concat("(", $now, "-", $ctime,
let $dif := x:eval($expr)
```

```
<ctimes>
<nodes>{count(input()/times/nodes/ctime)}</nc
<strokes>{count(input()/times/strokes/ctime)}</
<actions>{count(input()/times/actions/ctime)}</
</ctimes>
```

Figure C.8: “Last Activity” Implementation, part 2

```
{"</table></HTML>"} </Content>
</AnnotationNode>
```

C.7.2 XQuery 1 of part 2

```
<times><nodes> {
```

```
let $mux := input()
let $stime := $mux//CiCQuery[@label="Time"]
let $mtime := fn:concat($stime, "*60000")
let $time := x:eval($mtime)
let $input := $mux//CiCQuery[@label="CurrentDoc"]
let $now := fn:string($input/DocumentRoot/SessionData/@writeTime)
for $node in $input//NodeInfo/*[1]
let $ctime := xs:integer($node/@creationtime)
let $expr := fn:concat("(", $now,  "-", $ctime, ")")
let $dif := x:eval($expr)
where $dif < $time
return <ctime>{$dif}</ctime> } </nodes>
<strokes> {
let $mux := input()
let $stime := $mux//CiCQuery[@label="Time"]
let $mtime := fn:concat($stime, "*60000")
let $time := x:eval($mtime)
let $input := $mux//CiCQuery[@label="CurrentDoc"]
let $now := fn:string($input/DocumentRoot/SessionData/@writeTime)
for $stroke in $input//Stroke/@modified
let $ctime := xs:integer($stroke)
let $expr := fn:concat("(", $now,  "-", $ctime, ")")
let $dif := x:eval($expr)
where $dif < $time
return <ctime>{$dif}</ctime>}</strokes>
<actions>{
let $mux := input()
let $stime := $mux//CiCQuery[@label="Time"]
let $mtime := fn:concat($stime, "*60000")
let $time := x:eval($mtime)
let $input := $mux//CiCQuery[@label="ciclog"]
let $now := fn:current-time()
```

```
for $log in $input//LogEntry/@time
let $ctime := xs:time($log)
let $dif := xs:time($now) - xs:time($ctime)
where $dif < $time
return <ctime>{$dif}</ctime>}</actions>
</times>
```

C.7.3 Sample result for XQuery 1 of part 2

```
<times>
  <nodes/>
  <strokes/>
  <actions>
    <ctime>39167.012</ctime>
    <ctime>39145.012</ctime>
    <ctime>39141.012</ctime>
    <ctime>26513.012000000002</ctime>
    <ctime>26488.012000000002</ctime>
    <ctime>25188.012000000002</ctime>
    <ctime>25185.012000000002</ctime>
    <ctime>23709.012000000002</ctime>
    <ctime>23700.012000000002</ctime>
    <ctime>20864.012000000002</ctime>
    <ctime>20851.012000000002</ctime>
    <ctime>19572.012000000002</ctime>
    <ctime>18202.012000000002</ctime>
    <ctime>18178.012000000002</ctime>
  </actions>
</times>
```

C.7.4 XQuery 2, part 2

```
<ctimes>
<nodes>{count(input()/times/nodes/ctime)}</nodes>
<strokes>{count(input()/times/strokes/ctime)}</strokes>
<actions>{count(input()/times/actions/ctime)}</actions>
</ctimes>
```

C.8 “List MatchMaker Collaborations” Implementation

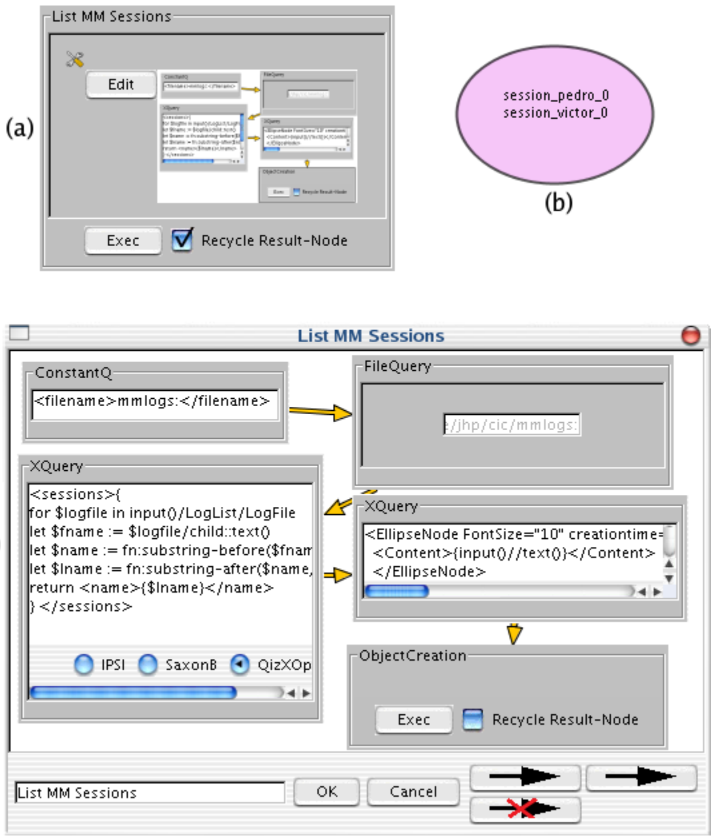
Figure C.9 presents in part (a) the “List MM Sessions” as seen by a teacher, a sample result of the execution thereof in part (b) and a detail of the implementation in part (c).

C.8.1 XQuery 1

```
<sessions>{
for $logfile in input()/LogList/LogFile
let $fname := $logfile/child::text()
let $name := fn:substring-before($fname, ".log")
let $lname := fn:substring-after($name, "_mm_")
return <name>{$lname}</name>
} </sessions>
```

C.8.2 XQuery 2

```
<EllipseNode FontSize="10" creationtime="1110487750862" creator="jhp"
```



```
<filename>mmlogs:</filename>
```

```
<sessions>{
for $logfile in input()/LogList/LogFile
let $fname := $logfile/child::text()
let $name := fn:substring-before($fnam
let $lname := fn:substring-after($name,
return <name>{$lname}</name>
} </sessions>
```

```
<EllipseNode FontSize="10" creationtime=
 <Content>{input()//text()}</Content>
 </EllipseNode>
```

```
List MM Sessions
```

Figure C.9: “List MatchMaker Collaborations” Implementation

```
    lastModificator="none" package="info.collide.plugins.mindmap"
    sticky="false" uiLocked="false">
  <Content>{input()//text()}</Content>
</EllipseNode>
```

C.9 “MatchMaker Activity Indicator” Implementation

Figure C.10 shows the “MatchMaker Activity Indicator” query as available to the teacher in part (a). Part (b) of the same figure shows the implementation detail, and part (c) show a sample output when executing the query.

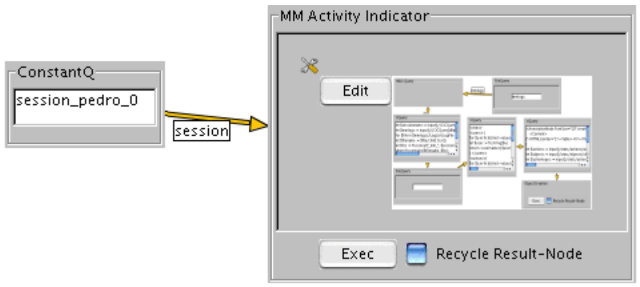
C.9.1 XQuery 1

```
let $sessionname :=
    input()//CiCQuery[@label="session"]/CiCQueryResult/child::text()
let $mmlogs := input()//CiCQuery[@label="mmlogs"]
for $file in $mmlogs//LogList/LogFile
let $filename := $file/child::text()
let $fss := fn:concat("_mm_", $sessionname, ".log")
where fn:contains($filename, $fss)
return <filename>mmlog:{$filename}</filename>
```

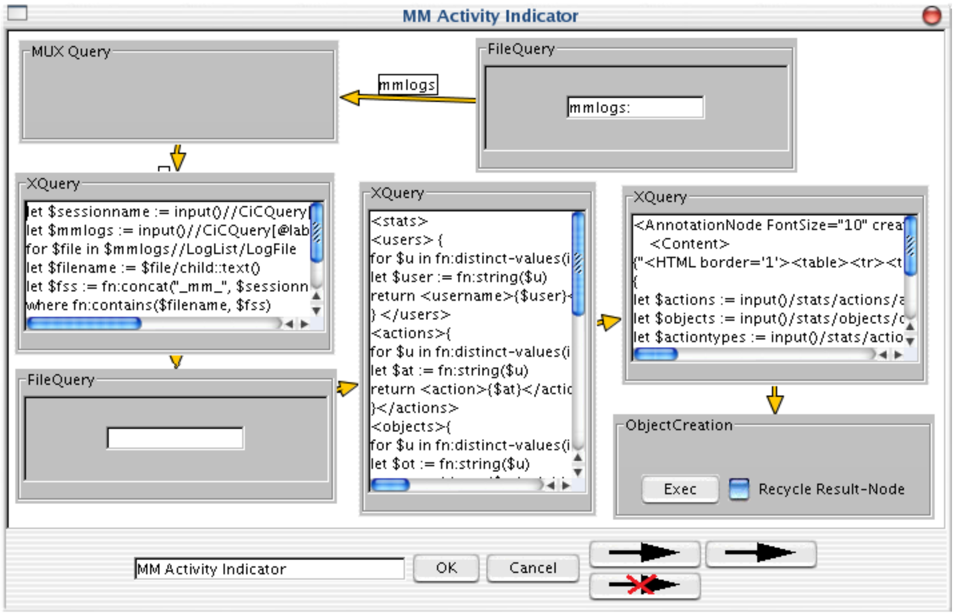
C.9.2 Sample output of XQuery 1

```
<filename>mmlog:26_6_2005_mm_session_pedro_0.log</filename>
```

(a)



(b)

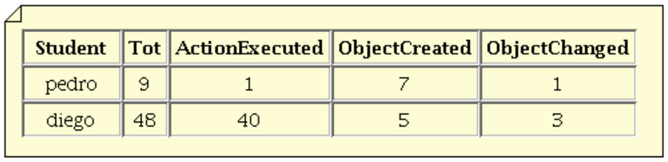


```
let $sessionname := input()//CiCQuery[
let $mmlogs := input()//CiCQuery[@lab
for $file in $mmlogs//LogList/LogFile
let $filename := $file/child::text()
let $fss := fn:concat("_mm_", $sessionn
where fn:contains($filename, $fss)
```

```
<stats>
<users> {
for $u in fn:distinct-values(i
let $user := fn:string($u)
return <username>{$user}<
} </users>
<actions>{
for $u in fn:distinct-values(i
let $at := fn:string($u)
return <action>{$at}</actic
}</actions>
<objects>{
for $u in fn:distinct-values(i
let $ot := fn:string($u)
```

```
<AnnotationNode FontSize="10" crea
  <Content>
{"<HTML border='1'><table><tr><t
{
let $actions := input()/stats/actions/a
let $objects := input()/stats/objects/c
let $actiontypes := input()/stats/actio
```

(c)



Student	Tot	ActionExecuted	ObjectCreated	ObjectChanged
pedro	9	1	7	1
diego	48	40	5	3

Figure C.10: “MatchMaker Activity Indicator” Implementation

C.9.3 XQuery 2

```
<stats>
<users> {
for $u in fn:distinct-values(input()//SyncAction/@user)
let $user := fn:string($u)
return <username>{$user}</username>
} </users>
<actions>{
for $u in fn:distinct-values(input()//SyncAction/@action)
let $at := fn:string($u)
return <action>{$at}</action>
}</actions>
<objects>{
for $u in fn:distinct-values(input()//SyncAction/@objectType)
let $ot := fn:string($u)
return <object>{$ot}</object>
}</objects>
<actiontypes>{
for $u in fn:distinct-values(input()//SyncAction/@typeOfAction)
let $ot := fn:string($u)
return <actiontype>{$ot}</actiontype>
}</actiontypes>
<SyncActions> {
let $sct := fn:current-dateTime()
let $epoch := xs:dateTime("1970-01-01T00:00:00.000-04:00")
let $ct := xs:integer($sct - $epoch)
for $sa in input()//SyncAction
let $time := $ct - xs:integer($sa/@time) div 1000
return <SyncAction secondsSince="{$time}"> {$sa/@user}
    {$sa/@action} {$sa/@objectType} {$sa/@typeOfAction}</SyncAction>
} </SyncActions>
```

```
</stats>
```

C.9.4 Sample output of XQuery 2

```
<stats>
  <users>
    <username>pedro</username>
    <username>diego</username>
  </users>
  <actions>
    <action>objectCreated</action>
    <action>objectChanged</action>
    <action>actionExecuted</action>
  </actions>
  <objects>
    <object>class info.collide.mm.sync.SyncTree</object>
    <object>class info.collide.xml.helpers.Point</object>
    <object>class info.collide.draw.Stroke</object>
  </objects>
  <actiontypes>
    <actiontype>not set</actiontype>
    <actiontype>setLocation</actiontype>
    <actiontype>addPoint</actiontype>
  </actiontypes>
  <SyncActions>
    <SyncAction secondsSince="1207.41" user="pedro"
        action="objectCreated"
        objectType="class info.collide.mm.sync.SyncTree"
        typeOfAction="not set"/>
    <SyncAction secondsSince="1205.098" user="pedro"
```

```
        action="objectCreated"
        objectType="class info.collide.mm.sync.SyncTree"
        typeOfAction="not set"/>
    <SyncAction secondsSince="1200.071" user="pedro"
        action="objectCreated"
        objectType="class info.collide.mm.sync.SyncTree"
        typeOfAction="not set"/>
    <SyncAction secondsSince="1192.523" user="pedro"
        action="objectCreated"
        objectType="class info.collide.mm.sync.SyncTree"
        typeOfAction="not set"/>
    <SyncAction secondsSince="1188.724" user="pedro"
        action="objectCreated"
        objectType="class info.collide.mm.sync.SyncTree"
        typeOfAction="not set"/>
    <SyncAction secondsSince="1186.712" user="pedro"
        action="objectCreated"
        objectType="class info.collide.mm.sync.SyncTree"
        typeOfAction="not set"/>
    <SyncAction secondsSince="1182.016" user="pedro"
        action="objectCreated"
        objectType="class info.collide.mm.sync.SyncTree"
        typeOfAction="not set"/>
    <SyncAction secondsSince="1180.349" user="pedro"
        action="objectChanged"
        objectType="class info.collide.xml.helpers.Point"
        typeOfAction="not set"/>
    <SyncAction secondsSince="1180.334" user="pedro"
        action="actionExecuted"
        objectType="class info.collide.xml.helpers.Point"
        typeOfAction="setLocation"/>
        ...

    <SyncAction secondsSince="1163.207" user="diego"
        action="objectChanged"
        objectType="class info.collide.draw.Stroke"
        typeOfAction="not set"/>
    <SyncAction secondsSince="1162.416" user="diego"
        action="objectCreated"
        objectType="class info.collide.draw.Stroke"
        typeOfAction="not set"/>
    <SyncAction secondsSince="1162.321" user="diego"
        action="actionExecuted"
        objectType="class info.collide.xml.helpers.Point"
        typeOfAction="addPoint"/>
    <SyncAction secondsSince="1162.301" user="diego"
        action="actionExecuted"
        objectType="class info.collide.xml.helpers.Point"
        typeOfAction="addPoint"/>
    <SyncAction secondsSince="1162.265" user="diego"
        action="actionExecuted"
        objectType="class info.collide.xml.helpers.Point"
        typeOfAction="addPoint"/>
    <SyncAction secondsSince="1162.237" user="diego"
        action="actionExecuted"
        objectType="class info.collide.xml.helpers.Point"
        typeOfAction="addPoint"/>
    <SyncAction secondsSince="1162.017" user="diego"
        action="actionExecuted"
        objectType="class info.collide.xml.helpers.Point"
        typeOfAction="addPoint"/>
    <SyncAction secondsSince="1161.989" user="diego"
        action="objectChanged"
        objectType="class info.collide.draw.Stroke"
        typeOfAction="not set"/>
    <SyncAction secondsSince="1152.453" user="diego"
        action="objectCreated"
        objectType="class info.collide.mm.sync.SyncTree"
        typeOfAction="not set"/>
    <SyncAction secondsSince="1149.156" user="diego"
        action="objectCreated"
        objectType="class info.collide.mm.sync.SyncTree"
        typeOfAction="not set"/>
  </SyncActions>
</stats>
```

C.9.5 XQuery 3

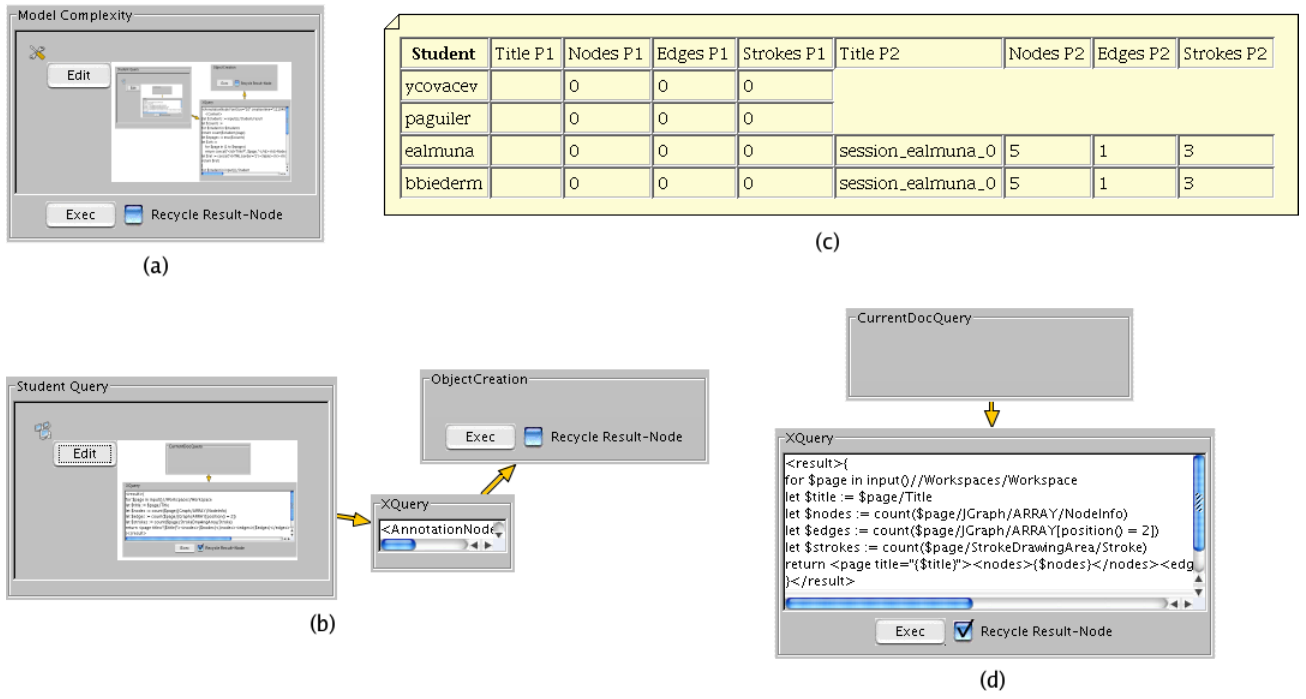
```
<AnnotationNode FontSize="10" creationtime="1110487750862" creator="jhp"
    lastModificator="none" package="info.collide.plugins.mindmap"
    sticky="false" uiLocked="false" EditorType="JEditorPane">
    <Content>
{"<HTML border='1'><table><tr><th>Student</th><th>Tot</th>
    <th>ActionExecuted</th><th>ObjectCreated</th>
    <th>ObjectChanged</th></tr>"}
{
let $actions := input()/stats/actions/action
let $objects := input()/stats/objects/object
let $actiontypes := input()/stats/actiontypes/actiontype
for $user in input()/stats/users/username
let $uname := $user/child::text()
let $syncactions :=
    input()/stats/SyncActions/SyncAction[fn:string(@user) eq $uname]
let $uno := "<tr><td align='center'>"
let $dos := "</td><td align='center'>"
let $tres := "</td></tr>"
let $ae := count($syncactions[@action eq "actionExecuted"])
let $ocr := count($syncactions[@action eq "objectCreated"])
let $och := count($syncactions[@action eq "objectChanged"])
let $todo := fn:concat($uno, $uname, $dos, count($syncactions), $dos,
    $ae, $dos, $ocr, $dos, $och, $tres)
return $todo
}
{"</table></HTML>"} </Content>
</AnnotationNode>
```

C.10 "Model Complexity" Implementation

Figure C.11 shows the query as presented to the teacher in part (a), and the details of the implementation in part (b). Part (c) is a sample output of the execution of the query, and part (d) the detail of the Student Query shown in part (b).

C.10.1 XQuery 1 of part (a)

```
<AnnotationNode FontSize="10" creationtime="1110487750862" creator="jhp"
    lastModificator="none" package="info.collide.plugins.mindmap"
    sticky="false" uiLocked="false" EditorType="JEditorPane">
    <Content>
{let $students := input()//Student/result
let $counts :=
for $student in $students
```



Student	Title P1	Nodes P1	Edges P1	Strokes P1	Title P2	Nodes P2	Edges P2	Strokes P2
ycovacev		0	0	0				
paguiler		0	0	0				
ealmuna		0	0	0	session_ealmuna_0	5	1	3
bbiederm		0	0	0	session_ealmuna_0	5	1	3

```
<result>{
for $page in input()//Workspaces/Workspace
let $title := $page/Title
let $nodes := count($page/JGraph/ARRAY/NodeInfo)
let $edges := count($page/JGraph/ARRAY[position() = 2])
let $strokes := count($page/StrokeDrawingArea/Stroke)
return <page title="{$title}"><nodes>{$nodes}</nodes><edg
}</result>
```

Figure C.11: “Model Complexity” Implementation

```
return count($student/page)
let $npages := max($counts)
let $sth :=
    for $page in (1 to $npages)
    return concat("<td>Title P", $page, "</td><td>Nodes P", $page,
        "</td><td>Edges P", $page, "</td><td>Strokes P", $page, "</td>")
let $ret :=
    concat("<HTML border='1'><table><tr><th>Student</th>", $sth, "</tr>")
return $ret}
{
for $student in input()//Student
let $phtml :=
    for $page in $student/result/page
    let $title := string($page/@title)
    let $nodes := $page/nodes/text()
    let $edges := $page/edges/text()
    let $strokes := $page/strokes/text()
    let $ph := concat("<td>", $title, "</td><td>", $nodes, "</td><td>",
        $edges, "</td><td>", $strokes, "</td>")
    return $ph
let $tot :=
    concat("<tr><td>", string($student/@username), "</td>", $phtml, "</tr>")
return $tot
}
{"</table></HTML>"} </Content>
</AnnotationNode>
```

C.10.2 XQuery of part (b)

```
<result>{
```

```
for $page in input()//Workspaces/Workspace
let $title := $page/Title
let $nodes := count($page/JGraph/ARRAY/NodeInfo)
let $edges := count($page/JGraph/ARRAY[position() = 2])
let $strokes := count($page/StrokeDrawingArea/Stroke)
return <page title="{$title}"><nodes>{$nodes}</nodes><edges>{$edges}
    </edges><strokes>{$strokes}</strokes></page>
}</result>
```

Bibliography

Abowd, G. D. (1999). Classroom 2000: An experiment with the instrumentation of a living educational environment. *IBM Systems Journal, Special issue on Pervasive Computing*, 38(4):508–530. 15, 24

Abowd, G. D., Atkeson, C., Feinstein, A., Goolamabbas, Y., Hmelo, C., Register, S., Sawhney, N., and Tani, M. (1996). Classroom 2000: Enhancing classroom interaction and review. Technical report, GVU Center. 24

Abowd, G. D., Harvel, L. D., and Brotherton, J. A. (2000). Building a digital library of captured educational experiences. In Kambayashi, Y., Wiederhold, G., Klavans, J., Winiwarter, W., and Tarumi, H., editors, *Kyoto International Conference on Digital Libraries 2000*, pages 395–402, Kyoto, Japan. 24

Alarcón, R., Guerrero, L. A., Ochoa, S., and Pino, J. A. (2005). Context in collaborative mobile scenarios. In *Proceedings of the Fifth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'05), Workshop on Context and Groupware*, volume 133, Paris, France. CEUR Proceedings. 23

Altshuler, A. (1992). *Breaking Through Bureaucracy*. Berkeley: University of California Press. 33

Baloian, N. (1997). *Strukturierte Erstellung und kooperative Nutzung von Instruktionsmaterial in einem Computerintegrierten Klassenraum*. PhD thesis, Gerhard-Mercator-Universität Gesamthochschule Duisburg, Duisburg, Germany. 29

Baloian, N., Berges, A., Buschmann, S., Gaßner, K., Hardings, J., Hoppe, H. U., and Luther, W. (2002a). Document management in a computer integrated classroom. In Haake, J. M. and Pino, J. A., editors, *Proceedings of CRIWG 2002, 8th International Workshop on Groupware*, volume 2440 of *Lecture Notes on Computer Science*, pages 35–46, Heidelberg, Germany. Springer-Verlag. 8

Baloian, N., Hoppe, H., and Kling, U. (1995). Structured authoring and cooperative use of instructional multimedia material for the computer-integrated classroom. In Maurer, H., editor, *Proceedings of the ED-MEDIA 95, World Conference on Multimedia and Hypermedia*, pages 81–86, Graz, Austria. Association for the Advancement of Computing in Education. 14, 29

Baloian, N., Hoppe, H. U., and Luther, W. (2002b). Teaching strategies to automatically configure course material in undergraduate lectures. 14

Baloian, N., Hoppe, U., and Luther, W. (2001). Structuring lesson material to flexibly support teaching in a computer integrated classroom. Technical Report Forschungsbericht 763, Universität Dortmund Fachbereich Informatik. 14

Bannon, L. J. and Schmidt, K. (1991). Cscw: Four characters in search of a context. Studies in Computer Supported Cooperative Work: Theory, Practice and Design. Previously printed in EC-SCW '89. Proceedings of the First European Conference on Computer Supported Cooperative Work, Gatwick, London, 13-15 September, 1989, pp. 358-372. 13

Barnard, D. T., Clarke, G., and Duncan, N. (1995). Tree-to-tree correction for document trees technical report 95-372. Technical report, Queens University. This is a corrected and expanded version of Technical Report 91-315. 166

Barros, B. and Verdejo, M. F. (2000). Analysing student interaction processes in order to improve collaboration. the degree approach. *International Journal of Artificial Intelligence in Education*, 11(3):221–241. 149

Beck, E. E. (1994). *Practices of Collaboration in Writing and Their Support*. PhD thesis, School of Cognitive and Computing Sciences, University of Sussex. Available as Technical Report CSRP 340, ISSN 1350-3162. 22

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. Addison-Wesley. 67

Beck, K., Coplien, J. O., Crocker, R., Dominick, L., Meszaros, G., Paulisch, F., and Vlissides, J. (1996). Industrial experience with design patterns. In *Proceedings of the 18th International Conference on Software Engineering*, pages 103–114. IEEE Computer Society Press. 33

Bendixsen, Synnøve de Guchteneire, P. (2003). Best practices in immigration services planning. *Journal of Policy Analysis and Management*, 22(4):677–682. 35, 37

Berge, Z. L. and Collins, M. P. (1995). *Computer-Mediated Communication and the Online Classroom*, volume 3 of *Computer-Mediated Communication and the Online Classroom*, chapter From Marks in the Sand to Computer Conferencing Via Fiber Optics. Hampton Press, Cresskill, New Jersey, USA.

Boag, S., Chamberlin, D., Fernández, M. F., Florescu, D., Robie, J., and Siméon, J. (2001). Xquery 1.0: An xml query language. http://www.w3.org/TR/xquery/. Last accessed: 13th June 2005. 66, 99, 115

Bollen, L., Hoppe, H. U., Milrad, M., and Pinkwart, N. (2002). Collaborative modelling in group learning environments. In I., D. P., Mollona, E., Diker, V. G., Langer, R. S., and Rowe, J. I., editors, *Proceedings of the 20th International Conference of the System Dynamics Society*, page 53, Palermo, Italy. System Dynamics Society. 31, 133

Borges, M. and Pino, J. A. (1999). Awareness mechanisms for coordination in asynchronous cscw. In *Proceedings of the 9th Workshop on Information Technologies and Systems (WITS '99)*, pages 69–74, Charlotte, North Carolina. 22

Borges, M. R., Brézillon, P., Pino, J. A., and Pomerol, J.-C. (2004). Context-based awareness in group work. In *Proceedings of 17th FLAIRS conference*, Florida, USA. 13

Bos, N., Gergle, D., Olson, J. S., and Olson, G. M. (2001). Being there versus seeing there: Trust via video. In *Proceedings of CHI 2001 Conference on Human Factors in Computing Systems*, New York. ACM Press. 16

Braga, D., Campi, A., and Ceri, S. (2005). Xqbe (xquery by example): a visual interface to the standard xml query language. *ACM Transaction On Database Systems TODS*, 30(2):398 – 443. 152

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. (2004). Extensible markup language (third edition). http://www.w3.org/TR/REC-xml. Last accessed: 13th June 2005. 66, 115

Bretschneider, S., Marc-Aurele, F. J., and Wu, J. (2005). Best practices research: A methodological guide for the perplexed. *Journal of Public Administration Research and Theory*, 15(2):307–323. 33, 38, 66

Brézillon, P., Borges, M., Pino, J., and Pomerol, J.-C. (2004). Context-awareness in group work: three case studies. In Meredith, R., Shanks, G., Arnott, D., and Carlsson, S., editors, *Proceedings of the IFIP International Conference on Decision Support Systems (DSS-2004): Decision Support in Uncertain and Complex World*, pages 115–124. 23

Brown, A. L. and Campione, J. C. (1994). Guided discovery in a community of learners. In McGilly, K., editor, *Classroom Lessons: Integrating Cognitive Theory and Classroom Practices*, chapter 9, pages 229–270. MIT Press. 19

Brown, J. S., Collins, A., and Dguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher*, 18:33–42. 19

Bruner, J. (1996). *The Culture of Education*. MIT Press, Cambridge, MA, USA. 19

Buckingham Shum, S., Marshall, S., Brier, J., and Evans, T. (2001). Lyceum: Internet voice groupware for distance learning. In Dillenbourg, P., Eurelings, A., and Hakkarainen, K., editors, *Proceedings of Euro CSCL 2001*, Maastricht, the Netherlands. Maastricht McLuhan Institute. 16

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M., Sommerlad, P., and Stal, M. (1996). *Pattern-oriented software architecture, volume 1: A system of patterns*. John Wiley & Sons, Chichester, England. 64

Calculator, S. Jorgenson, C. M. (1991). Integrating aac instruction into regular education settings: Expounding on best practices. *Augmentative and Alternative Communication*, 7:204–214. 149

Catarci, T. and Santucci, G. (1995). Are visual query languages easier to use than traditional ones? an experimental proof. In *Proceedings of HCI '95*, pages 323–338. 96

Cedergren, M. (2003). Open Content and Value Creation. *First Monday*, 8(8). 68

Chickering, A. W. and Gamson, Z. F. (1987). Seven principles for good practice in undergraduate education. *AAHE Bulletin*, 39(7):3–7. 38

Chizmar, J. F. and Williams, D. B. (1997). Internet delivery of instruction: Issues of best teaching practice, administrative hurdles, and old-fashioned politics. In *'The Information Profession and the Information Professional', proceedings of the 1997 CAUSE conference.* 5

Clark, H. H. and Schaefer, E. F. (1989). Contributing to discourse. *Cognitive Science*, 13:259 – 294. 17, 18

Clark, J. (1999). Xsl transformations (xslt) version 1.0. http://www.w3.org/TR/xslt. Last accessed: 13th June 2005. 66, 99, 115

Clark, J. and DeRose, S. (1999). Xmlpath language (xpath) version 1.0. http://www.w3.org/TR/xpath. Last accessed: 13th June 2005. 115

Cobéna, G. (2003). *Gestion des changements pour les données semi-structurées du Web (Change management of semi-structured data on the Web).* PhD thesis, l École Polytechnique, France.

Cobéna, G., Abdessalem, T., and Hinnach, Y. (2002a). A comparative study for xml change detection. 18eme Journees Bases de Données Avancées (BDA'02). GemoReport221. 166

Cobéna, G., Abiteboul, S., and Marian, A. (2002b). Detecting changes in xml documents. In *Proceedings of International Conference on Data Engineering (ICDE) 2002*, San Jose, California, USA. 166

Coldex (2005). Coldex project homepage. http://www.coldex.info/. Last accessed: 13th June 2005. 31

Collazos, C., Guerrero, L. A., and Pino, J. A. (2003). Knowledge construction awareness. *Journal of Student-Centered Learning*, 1(2):73–82. 19

Collide (2003). Jgraph howto. Last accessed 15th June, 2005. 84

Cortez, C., Nussbaum, M., López, X., Rodríguez, P., Santelices, R., Rosas, R., and Marianov, V. (2005). Teacher's support with ad-hoc collaborative networks. *Journal of Computer Assisted Learning*, 21:171–180. 8, 17

Crook, C. (1994). *Computers and the Collaborative Experience of Learning*. Routledge, London. 18

Crosby, P. (1980). *Quality is Free*. Mentor Books, New York. 33

David, J. M. N. and Borges, M. R. (2001). Selectivity of awareness components in asynchronous cscw environments. In *Proceedings of CRIWG 2001 (7th. Intl. Workshop on Groupware)*, pages 115–124, Los Alamitos, CA. IEEE Comp. Soc. Press. 22

Dawes, S. S., Pardo, T. A., Green, D. E., McInerney, C. R., Connelly, D. R., and DiCaterino, A. (1997). Tying a sensible knot - a practical guide to state-local information systems. June, Center for Technology in Government University at Albany, SUNY. 34

Deming, W. E. (1982). *Out of the Crisis*. MIT Press International. 33

Dillenbourg, P. (1999). What do you mean by 'collaborative learning'? In Dillenbourg, P., editor, *Collaborative Learning: Cognitive and Computational Approaches*, Advances in learning and instruction, chapter 1, pages 1–19. Elsevier, Oxford, first edition 1999 edition. 19

Dillenbourg, P., Baker, M., Blaye, A., and O'Malley, C. (1996). The evolution of research on collaborative learning. In Peter Reimann and Hans Spada, editors, *Learning in Humans and Machines: Towards an Interdisciplinary Learning Science*, chapter 11, pages 189 – 211. Elsevier Science.

Dix, A., Finlay, J., Abowd, G. D., and Beale, R. (1998). *Human Computer Interaction*. Prentice Hall. 2

Dourish, P. and Bellotti, V. (1992). Awareness and coordination in shared workspaces. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'92)*, pages 107–114, Toronto, Ontario. ACM Press. 13, 22

Edelson, D. C., Pea, R. D., and Gomez, L. (1996). Constructivism in the collaboratory. In Wilson, B. G., editor, *Constructivist learning environments: Case studies in instructional design*, pages 151–164. Educational Technology Publications, Englewood Cliffs. 25

Eklund, J., Brusilovsky, P., and Schwarz, E. (1997). Adaptive textbooks on the world wide web. In *Proceedings Ausweb97, Third Australian World Wide Web Conference*, Queensland, Australia. 14, 15

Ellis, C. A., Gibbs, S. J., and Rein, G. (1991). Groupware: some issues and experiences. *Communications of the ACM*, 31(1):39–58. 13

Elrod, S., Bruce, R., Gold, R., Goldberg, D., Halasz, F. G., Jr., W. C. J., Lee, D., McCall, K., Pedersen, E. R., Pier, K. A., Tang, J. C., and Welch, B. (1992). Liveboard: A large interactive display supporting group meetings, presentations, and remote collaboration. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, Desks, Video, and Screens, pages 599–607, New York, USA. ACM Press. 28

Engeström, Y. (1987). *Learning by Expanding: An Activity - Theoretical Approach to Developmental Research*. Orienta-Konsultit Oy, Helsinki. 19

Fankhauser, P. and Lehti, P. (2003). Ipsi-xq: Eine implementierung von xquery und seiner formalen semantik. *Information Technology*, 45(3):133–136. 172

FET (2001). The disappearing computer initiative. http://www.disappearing-computer.net/. EU-funded proactive initiative of the Future and Emerging Technologies (FET) activity of the Information Society Technologies (IST) research program. 30

Franc, X. (2005). Qizx/open project homepage. http://www.xfra.net/qizxopen/. 152, 173

Frank, C. R. and Uy, F. L. (2004). Ethnography for teacher education. *Journal of Teacher Education*, 55(3):269–283. 69

Fussell, S. R., Kraut, R. E., Siegel, J., and Brennan, S. E. (2002). Relationships among speech, vision, and action in collaborative physical tasks. In *CHI '02: CHI '02 extended abstracts on Human factors in computing systems*, pages 916–917, New York, NY, USA. ACM Press. 20

Galegher, J. and Kraut, R. E. (1990). Computer-mediated communication for intellectual teamwork: A field experiment in group writing. In *Proceedings of CSCW'90 Computer-Supported Cooperative Work*, pages 65–78, Los Angeles, USA. ACM Press, NY. 22

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley. 33, 102

Gardner, M. (1970). Mathematical games: The fantastic combinations of john conway's new solitarire game 'life'. *Scientific American*, 4(223):120–123. 68

Gardner, M. (1983). *Life, and other Mathematical Amusements*, chapter 20-22 (The Game of Life, Parts I-III). W. H. Freeman, New York. 68

Goodman, B., Geier, M., Haverty, L., Linton, F., and McCready, R. (2001). A framework for asynchronous collaborative learning and problem solving. In Moore, J. D., Redfield, C. L., and Johnson, W. L., editors, *Proceedings of the 10 th International Conference on Artificial Intelligence in Education AIED 2001*, volume 68 of *Frontiers in Artificial Intelligence and Applications*, San Antonio, Texas. IOS Press. 32

Graf, F. and Schnaider, M. (1998). Ideals mts - a modular training system for the future. In *Proceedings of EDMedia 98*, Freiburg, Germany. 14, 15

Grudin, J. (1990a). The computer reaches out: the historical continuity of interface design. In *Proceedings of CHI'90*, pages 261 – 268, Seattle, Washington, United States. ACM Press. 13

Grudin, J. (1990b). Interface. In *Proceedings of the Conference on Computer Supported Cooperative Work CSCW'90*, pages 269–279, Los Angeles, CA. ACM Press. 13

Grudin, J. (1994). Computer-supported cooperative work: History and focus. *IEEE Computer*, 27(5):19–26. 13

Gutwin, C., Greenberg, S., and Roseman, M. (1996). Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation. In Sasse, A., Cunningham, J., and Winder, R., editors, *People and Computers XI (Proceedings of the HCI'96)*, pages 281–298, London, UK. Springer-Verlag. 13

Guye-Vuillème, A., Capin, T. K., Pandzic, I. S., Thalmann, N. M., and Daniel Thalmann (1999). Nonverbal communication interface for collaborative virtual environments. *The Virtual Reality Journal*, 4:49–59. 16

Hajiaghayi, M. T. (2001). Comparing of sgml documents. Available at http://www.mit.edu/ hajiagha/pub.html. 166

Hardings, J., Baloian, N., and Hoppe, H. U. (2003). A document-centered architecture for classroom collaboration. In Hoppe et al. (2003), pages 419–421. 8

Harel, I. (1991). *Constructionism*, chapter The silent observer and holistic note taker, pages 449–464. Ablex Publishing Corporation, Norwood, NJ. 69

Harrison, C., Comber, C., Fisher, T., Haw, K., Lewin, C., Lunzer, E., McFarlane, A., Mavers, D., Scrimshaw, P., Somekh, B., and Watling, R. (2002). The impact of information and communication technologies on pupil learning and attainment. Technical report, British Educational Communications and Technology Agency. 149

Hayman, C. A. (2002). Electronic government (e-government): Deal makers and deal breakers - analysis of e-government state of texas texasonline and other state-level planning and implementation best practices. Master's thesis, University of Texas at Austin. 33

Hernández-Leo, Davinia Asensio-Pérez, J. I. D. Y. A. B.-L. M. L. J. A. I. M. V.-F. E. D. (2005). Reusing ims-ld formalized best practices in collaborative learning structuring. *Advanced Technology for Learning*, 2(4). 8, 9, 40, 151

Hiltz, S. R. and Wellman, B. (1997). Asynchronous learning networks as a virtual classroom. *Communications of the ACM*, 40(9):44–49. 2

Hirschberg, D. S. (1977). Algorithms for the longest common subsequence problem. *Journal of the ACM*, 24(4):664–675. 166

Hoppe, H. U. (2002). Computers in the classroom - a disappearing phenomenon? In Dimitracopoulou, A., editor, *Proceedings of the 3rd Hellenic Conference with International Participation on Information and Communication Technologies in Education (HICTE)*, pages 19–30, University of the Aegean, Rhodes, Greece. Kastaniotis Editions, Inter@ctive. 29, 31

Hoppe, H. U., Baloian, N., and Zhao, J. (1993). Computer support for teacher-centered classroom interaction. In *Proceedings of ICCE '93*, pages 211–217, Taipei (Taiwan). 28, 29, 75

Hoppe, H. U. and Gaßner, K. (2002). Integrating collaborative concept mapping tools with group memory and retrieval functions. In Stahl (2002), pages 716–725. 83, 87, 92

Hoppe, H. U., Lingnau, A., Machado, I., Paiva, A., Prada, R., and Tewissen, F. (2000). Supporting collaborative activities in computer integrated classrooms the nimis approach. In *Proceedings of 6th International Workshop on Groupware (CRIWG 2000)*, pages 94–103, Los Alamitos, California, USA. IEEE Computer Society Press. 3, 28, 30, 75

Hoppe, H. U., Luther, W., Mühlenbrock, M., Otten, W., and Tewissen, F. (1999). Interactive presentation support for an electronic lecture hall - a practice report. In Cumming, G., Gomez, L., and Okamoto, T., editors, *Advanced Research in Computers and Communications in Education*, pages 923–930. IOS Press, Amsterdam, The Netherlands. 28

Hoppe, H. U., Verdejo, F., and Kay, J., editors (2003). *Proceedings of 11th International Conference on Artificial Intelligence in Education (AIED 2003)*, Amsterdam. IOS Press. 223, 225, 230

Hutchins, E. (1995). *Cognition in the Wild*. MIT Press. 19

Hutchins, E. L., Hollan, J. D., and Norman, D. A. (1986). Direct manipulation interfaces. In Norman, D. A. and Draper, S. W., editors, *User centered system design*, pages 87–124. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, USA. 30

IEEE (2002). Ieee 1484.12.1 standard for learning object metadata. http://ltsc.ieee.org/wg12/par1484-12-1.html. 6, 14

IMS Global Learning Consortium, I. (2003a). Ims content packaging specification v. 1.1.3. http://www.imsproject.org/content/packaging/. 14

IMS Global Learning Consortium, I. (2003b). Ims learning design specification. http://www.imsglobal.org/learningdesign/. 40

IMS Global Learning Consortium, I. (2004). Ims learning resource meta-data specification, version 1.3. http://www.imsglobal.org/metadata/. 14

Initiative, A. D. L. A. (2004). The sharable content object reference model (scorm). http://www.adlnet.org/index.cfm?fuseaction=scormabt. 14

Ishii, H. and Ullmer, B. (1997). Tangible bits: Towards seamless interfaces between people, bits and atoms. In Pemberton, S., editor, *Proceedings of Conference on Human Factors in Computing Systems CHI '97*, Atlanta, GA, USA. 29

Jackson, B. (2000). Review of leading asynchronous, web based course delivery tools. http://www.knowledgeability.biz/weblearning/reviewasych.htm. Last accessed: Nov 22, 2002. 16

Jansen, M. (2003). Matchmaker tng - a framework to support collaborative java applications. In Hoppe et al. (2003). 50, 86, 90, 93

Jansen, M., Pinkwart, N., and Tewissen, F. (2001). Matchmaker - flexible synchronisation von java-anwendungen. Technical Report Forschungsbericht 763, Universität Dortmund Fachbereich Informatik. 50, 93

Johnson, D. W. and Johnson, R. T. (1998). *Learning Together and Alone: Cooperative, Competitive, and Individualistic Learning*. Prentice-Hall. 17

Johnson, H. and Hyde, J. (2003). Towards modeling individual and collaborative construction of jigsaws using task knowledge structures (tks). *ACM Trans. Comput.-Hum. Interact.*, 10(4):339–387. 20, 21

Juran, J. and Gryna, F. M. (1988). *Juran's Quality Control Handbook*. McGraw-Hill. 33

Kay, M. H. (2005). Saxon homepage: The xslt and xquery processor. Online at http://saxon.sourceforge.net/. 172

Koschmann, T. (1996a). *CSCL: theory and practice of an emerging paradigm*. Computers, Cognition and Work. Lawrence Erlbaum Associates, Mahwah, New Jersey. 17, 18, 226, 231

Koschmann, T. (1996b). *Paradigm Shifta and Instructional Technology: An Introduction*, chapter 1, pages 1–23. In Koschmann (1996a). 17, 18

Koschmann, T. (1999). Computer support for collaboration and learning. *The Journal of the Learning Sciences*, 8(3/4):495–497.

Koschmann, T. (2001). Revisiting the paradigms of instructional technology. In Kennedy, G., Keppell, M., McNaught, C., and Petrovic, T., editors, *Proceedings of ASCILITE 2001*. 17, 18

Koschmann, T. D. (1994). Toward a theory of computer support for collaborative learning. *The Journal of the Learning Sciences*, 3(3):218–225. 18, 19

Krasner, G. E. and Pope, S. T. (1988). A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *Journal of Object-Oriented Programming*, 1(3):26–49. 64

Lave, J. (1988). *Cognition in Practice : Mind, Mathematics and Culture in Everyday Life*. Cambridge University. 19

Lave, J. (1991). Situating learning in communities of practice. In Resnick, L., Levine, J., and Teasley, S., editors, *Perspectives on Socially Shared Cognition*, pages 63–83. American Psychological Association APA, Washington DC, USA. 19

Lave, J. and Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge University Press, New York. 19

Lessig, L. (2004). *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*, chapter "Us, Now", pages 276–286. 68

Lin, J., Ho, C., Sadiq, W., and Orlowska, M. E. (2002). Using workflow technology to manage flexible e-learning services. *Educational Technology & Society*, 5(4). 15

Lingnau, A. and Hoppe, U. (2002). Modelling and supporting learning activities in a computer-integrated classroom. In Stahl (2002), pages 589–590. 3, 28, 30, 75

Lingnau, A., Kuhn, M., Harrer, A., Hofmann, D., Fendrich, M., and Hoppe, H. U. (2003). Enriching traditional classroom scenarios by seamless integration of interactive media. In Devedzic, V., Spector, J. M., Sampson, D. G., and Kinshuk, editors, *Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT)*, pages 135–139, Los Alamitos, CA (USA). IEEE Press. 31, 129

Lipponen, L. (2001). Supporting collaboration with computers. Perspectives of CSCL in Europe: A Review. A report for the European Commission, ITCOLE Project, IST-2000-26249. 18

Lipponen, L. (2002). Exploring foundations for computer-supported collaborative learning. In Stahl (2002). 17, 18, 19

Lu, S. (1979). A tree-to-tree distance and its application to cluster analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-1*, 2:19–224. 166

Macías, J. A. and Castells, P. (2001). An authoring tool for building adaptive learning guidance systems on the web. In Jiming Liu, Pong Chi Yuen, Chun Hung Li, Joseph Kee-Yin Ng, and Toru Ishida, editors, *Proceedings of Active Media Technology, 6th International Computer Science Conference, AMT 2001*, volume 2252 of *Lecture Notes in Computer Science*, pages 268–278, Heidelberg, Germany. Springer-Verlag. 14

MacKenzie, D., Eggert, P., and Stallman, R. (2003). *Comparing and Merging Files with GNU diff and patch*. Network Theory Ltd. 166

Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40. 22, 69

Malone, T. W. and Crowston, K. (1994). The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119. 13

Martin, L. L., Myers, S., Nelson, J., Phillips, W., Smith, H., Almquist, D., Chandler, M., and Blickem, T. (2004). Orange county central receiving center (crc) phase 2: Review of best practices in community mental health & substance abuse services. Technical report, Center for Community Partnerships College of Health & Public Affairs University of Central Florida. 33

Marzano, R. J., Pickering, D., and Pollock, J. E. (2001). *Classroom Instruction That Works: Research-Based Strategies for Increasing Student Achievement*. Association for Supervision & Curriculum Deve. 39

McCalla, G. (2004). The ecological approach to the design of e-learning environments: Purpose-based capture and use of information about learners. *Journal of Interactive Media in Education*. 43, 93, 94, 151

Mead, G. H. (1934). *Mind, Self and Society*. University Chicago Press. 19

Minsky, M. (1986). *The Society of Mind*. Simon and Schuster, NY, ; ACM CR 8803-0161. 19

Mold, James W. Gregory, M. E. (2003). Best practices research. *Family Medicine Journal*, 35(3):131–134. 33, 34

Mühlenbrock, M. (2001). *Action-based Collaboration Analysis for Group Learning*. PhD thesis, Gerhard-Mercator-Universität Duisburg. 93, 149

Mühlenbrock, M., Tewissen, F., and Hoppe, H. U. (1997). A framework system for intelligent support in open distributed learning environments. In du Boulay, B. and Mizoguchi, R., editors, *Artificial intelligence in education: Knowledge and media in learning systems*, pages 191–198. 15

Mühlhäuser, M. and Trompler, C. (2002). Learning in the digital age: Paving a smooth path with digital lecture halls. In *35th Hawaii Intl. Conference on System Sciences HiCSS*, Los Alamitos, CA. IEEE CS press. 24, 27

Mulder, M. C., Lidtke, D., and Stokes, G. E. (1997). Enterprise enhanced education: an information technology enabled extension of traditional learning environments. In *SIGCSE '97: Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education*, pages 355–359, New York, NY, USA. ACM Press. 2

Murray, T. (1998). Authoring knowledge based tutors: Tools for content, instructional strategy, student model, and interface design. *Journal of the Learning Sciences*, 7(1):5 – 64. 14

Myers, S., Smith, H. P., and Martin, L. L. (2004). Conducting best practices research in public affairs. In *First Annual Public Affairs Research Conference*. 34, 66

Niu, X., McCalla, G., and Vassileva, J. (2003). Purpose-based user modelling in a multi-agent portfolio management system. In *Proceedings of UM 2003: International Conference on User Modeling,*, pages 398–402, Johnstown, Pennsylvania. 93

Norman, D. A. (1998). *The invisible computer: why good products can fail, the personal computer is so complex, and information appliances are the solution*. MIT Press, Cambridge, Massachussets, USA; London, England. 30

Norman, D. A. (2002). *The Design of Everyday Things*. Basic Books. Previously published as The Psychology of Everyday Things. 4

Ochoa Delorenzi, S. F. (2002). *Tesis para optar al grado de Doctor en Ciencias de la Ingeniería*. PhD thesis, Pontificia Universidad Católica de Chile. 2, 6

Odon, S. L., Brantlinger, E., Gersten, R., Horner, R. H., Thompson, B., and Harris, K. R. (2005). Research in special education: Scientific methods and evidence-based practices. *Exceptional Children*, 71(2):137–148. 39

O'Neill, E. J., Johnson, P., and Johnson, H. (1999). Representations and user-developer interaction in cooperative analysis and design. *Human Computer Interaction*, 14(1&2):43–91. 20

Overman, E. Boyd, J. (1994). Best practice research & postbureaucratic reform. *Journal of Public Administration Research & Theory*, 4(1):67–83. 34

Paavola, S., Lipponen, L., and Hakkarainen, K. (2002). Epistemological foundations for cscl: A comparison of three models of innovative knowledge communities. In Stahl (2002). 19

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. Basic Books, New York. 18

Paterson, B. L., Bottorff, J. L., and Hewat, R. (2003). Blending observational methods: Possibilities, strategies, and challenges. *International Journal of Qualitative Methods*, 2(1). 69

Paulk, M. C., Weber, C. V., Curtis, B., and Chrissis, M. B. (1995). *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley. 66

Pea, R. D. (1993). Distributed multimedia learning environments: The collaborative visualization project. *Communications of the ACM*, 36(5):60–63. 16, 19, 25

Piaget, J. (1969). *Judgement and reasoning in the child*. Routledge and Paul, London. 19

Pinelle, D., Gutwin, C., and Greenberg, S. (2003). Task analysis for groupware usability evaluation: Modeling shared-workspace tasks with the mechanics of collaboration. *ACM Trans. Comput.-Hum. Interact.*, 10(4):281–311. 20

Pinelle, D., Gutwin, C., and Greenberg, S. (2004). Collaboration usability analysis: task analysis for groupware usability evaluations. *interactions*, 11(2):7–8. 20

Pinkwart, N. (2003). A plug-in architecture for graph based collaborative modeling systems. In Hoppe et al. (2003), pages 535–536. 32, 87

Pinkwart, N. (2005). *Collaborative Modeling in Graph Based Environments*. PhD thesis, Universität Duisburg-Essen. 87

Raymond, E. S. (1998). *The Cathedral and the Bazaar*, chapter 3, pages 27–78. O'Reilly and Associates, Sebastopol, California. 67

Reichen, J. (1991). *Lesen durch Schreiben. (German teacher's guide to "Reading through Writing")*. Heinevetter Lehrmittel Verlag. 3, 30

Retalis, S., Makrakis, V., and Skordalakis, E. (1997). Eont courseware development methodology. In *Proceedings of EdMedia '97*, pages 1073–1079, Calgary, Alberta, Canada. 2, 6

Rodriguez Artacho, M., Verdejo, M., Mayorga, J., and Calero, M. (1999). Using a high-level language to describe and create web-based learning scenarios. In *Proceedings of the IEEE Frontiers in Education Conference. FIE 99*. 14

Rogers, C. (1969). *Freedom to Learn*. Merill, Columbus, OH, USA. 5

Rontu, M. (2004). Visual queries for a student information system. Master's thesis, Department of Computer Science and Engineering, Helsinki University of Technology. 96

Roschelle, J. and Teasley, S. (1995). The construction of shared knowledge in collaborative problem solving. F 128 - NATO ASI, pages 69 – 97. Springer-Verlag, Berlin. Proceedings resulting from ARW, Maratea (Italy) , 1989. 17, 18

Ruchti, Wendy P. Odell, M. L. (2002). Comparison and evaluation of online and classroom instruction in elementary science teaching methods courses. In *Proceedings of the Northwest NOVA Cyber-Conference 2002*. 33, 52, 148

Scacchi, W. (2003). When is Free/Open Source Software Development Faster, Better, and Cheaper than Software Engineering? Working Paper available via http://www.ics.uci.edu/%7Ewscacchi/Papers/New/Scacchi-BookChapter.pdf (checked: 05 Sep 2005). 68

Scardamalia, M. and Bereiter, C. (1996). Computer support for knowledge-building communities. In Koschmann (1996a), pages 249 – 268. 5, 19

Seed (2005). Seed project homepage. http://ilios.cti.gr/seed. Last accessed 13th June 2005. 30

Selkow, S. M. (1977). The tree-to-tree editing problem. *Information Processing Letters*, 6(6):184–186. 166

Sohlenkamp, M. (1999). *Supporting Group Awareness in Multi-User Environments through Perceptualization*. Number 6 in GMD Research Series. GMD - Forschungszentrum Informationstechnik GmbH.

Sparks, R., Dooley, S., Meiskey, L., and Blumenthal, R. (1999). The leap authoring tool: Supporting complex courseware authoring through ruse, rapid prototyping and interactive visualizations. *International Journal of Artificial Intelligence in Education*, 10:75–97. 2, 6

Stahl, G. (2000). A model of collaborative knowledge-building. In Fishman, B. and O'Connor-Divelbiss, S., editors, *Fourth International Conference of the Learning Sciences*, pages 70–77, Mahwah, NJ. Erlbaum. 19

Stahl, G., editor (2002). *Proceedings of CSCL 2002. Computer Support for Collaborative Learning: Foundations for a CSCL Community*. Lawrence Erlbaum Associates. 224, 227, 230

Stallman, R. M. (2002). *Free software, free society: selected essays of richard m. stallman*. GNU Press, Boston, Massachusetts. 67

Stefik, M., Foster, G., Bobrow, D. G., Kahn, K., Lanning, S., and Suchman, L. (1987). Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Communications of the ACM*, 30(1):32–47. 23

Steinacker, A., Seeberg, C., Fischer, S., and Steinmetz, R. (1999). Multibook: Metadata for the Web. In *2nd International Conference on New Learning Technologies, Bern, Swizzerland*, pages 16–24. 15

Streitz, N. A., Geißler, J., Holmer, T., Konomi, S., Müller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P., and Steinmetz, R. (1999). i-land: An interactive landscape for creativitiy and innovation. In *ACM Conference on Human Factors in Computing Systems CHI '99*, pages 120–127, New York, USA. ACM Press. 28

Sugimoto, M., Hosoi, K., and Hashizume, H. (2004). Caretta: a system for supporting face-to-face collaboration by integrating personal and shared spaces. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 41–48, New York, NY, USA. ACM Press. 22, 26

Tai, K.-C. (1979). The tree-to-tree correction problem. *Journal of the ACM*, 26(3):422–433. 166

Tewissen, F., Baloian, N., Hoppe, H. U., and Reimberg, E. (2000a). "matchmaker" synchronising objects in replicated software-architectures. In *Proceedings of 6th International Workshop on Groupware (CRIWG 2000)*, Los Alamitos, California, USA. IEEE Computer Society Press. 86

Tewissen, F., Lingnau, A., and Hoppe, H. U. (2000b). "today's talking typewriter" supporting early literacy in a classroom environment. In Gauthier, G., Frasson, C., and VanLehn, K., editors, *Intelligent Tutoring Systems, 5th International Conference, Proceedings of ITS 2000*, volume 1839 of *Lecture Notes in Computer Science*, Montréal, Canada. Springer-Verlag. 30

Tewissen, F., Lingnau, A., Hoppe, H. U., Mannhaupt, G., and Nischk, D. (2001). Collaborative writing in a computer-integrated classroom for early learning. In *Proceedings of Euro CSCL 2001*. 3, 28, 30, 75

Tomasello, M. (1999). *The cultural origins of human cognition*. Harvard University Press, Cambridge, MA. 19

University, M. (1999). Comparison of online course delivery software products. http://www.marshall.edu/it/cit/webct/compare/comparison.html. Last accessed: Nov 22, 2002. 16

Van Marcke, K. (1992). Instructional expertise. In C. Frasson, G. G. . G. M., editor, *Intelligent Tutoring Systems*, volume 608 of *Lecture Notes in Computer Science*, pages 234–243. Springer-Verlag. 42

Vygotsky, L. S. (1962). *Thought and language*. Harvard University Press, Cambridge, MA. 19

Vygotsky, L. S. (1978). *Mind in society: The Development of Higher Psychological Processes*. Harvard University Press, Cambridge, Massachusetts and London, England. 19

Walker Tileston, D. E. (2005). *Ten Best Teaching Practices: How Brain Research, Learning Styles, and Standards Define Teaching Competencies*. Corwin Press. 39

Wang, H. W. and Chee, Y. S. (2001). Supporting workspace awareness in distance learning environments: Issues and experiences in the development of a collaborative learning system. In *Proceedings of ICCE/SchoolNet 2001 - Ninth International Conference on Computers in Education*, pages 1109–1116, Seoul, South Korea. 16

Wang, Y., DeWitt, D. J., and Cai, J.-Y. (2003). X-diff: a fast change detection algorithm for xml documents. In *Proceedings of ICDE 2003, 19th International Conference on Data Engineering*, Bangalore, India. IEEE Computer Society. 166, 167

Wegerif, R. (1998). The social dimension of asynchronous learning networks. *Journal of Asynchronous Learning Networks*, 2(1):34–49. 16

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):94–104. 30

Wundt, W. (1921). *Elements of Folk Psychology*. Allen and Unwin, London. 19

Yue, K., Yang, T., Ding, W., and Chen, P. (2004). A model for open content communities to support effective learning and teaching. In *Proceedings of the IADIS International Conference on Web-based Communities*, pages 533–536, Lisbon, Portugal. 68

Zemelman, S., Daniels, H., and Hyde, A. (2005). *Best Practice: Today's Standards for Teaching and Learning in America's Schools*. Heinemann. 39

Zhang, K., Statman, R., and Shasha, D. (1992). On the editing distance between unordered labeled trees. *Information Processing Letters*, 42(3):133–139. Republished in International Journal of Foundations of Computer Science, Volume 7, Number 1, March 1996. 166

Zurita, G. and Nussbaum, M. (2004). Mcscl: Mobile computer supported collaborative learning. *Computers and Education*, 42:289–314. 2