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

JENS HARDINGS PERL

Profesores guía: Nelson Baloian
Ulrich Hoppe
José A. Pino
Contents

1 Introduction .................................................. 1

1.1 Motivation ................................................. 1

1.1.1 Benefits of using IT in classrooms ......................... 2

1.1.2 Challenges to widespread use of Computers in Classrooms .......... 5

1.1.3 Facing the Challenges .................................. 7

1.2 Hypotheses ............................................... 8

1.3 Global Objectives ....................................... 10

1.4 Specific Objectives ..................................... 10

1.5 Contributions .......................................... 10

2 Computer-integrated Classrooms ......................... 13

2.1 CSCW and Groupware .................................. 13
CONTENTS

2.2 Computer-Supported Collaborative Learning (CSCL) ................................. 14

2.3 Context and Awareness ............................................................................... 17

2.4 Computer-supported Teaching and Learning ............................................... 18
   2.4.1 Distance learning .................................................................................. 18
   2.4.2 Course Authoring ............................................................................... 19
   2.4.3 Course Presentation ............................................................................ 20
   2.4.4 Course Outcome Analysis .................................................................. 21

2.5 Computer-enabled classrooms ..................................................................... 21
   2.5.1 Colab ................................................................................................. 21
   2.5.2 CoVis (http://www.covis.nwu.edu/info/) ............................................. 22
   2.5.3 eClass (formerly Classroom2000) ...................................................... 22

2.6 Computer-integrated Classroom (CiC) ......................................................... 22
   2.6.1 Cosoft ............................................................................................... 24
   2.6.2 NIMIS .............................................................................................. 25
   2.6.3 SEED ............................................................................................... 25
   2.6.4 COLDEX .......................................................................................... 26

2.7 CiCv2: a Domain-independent CiC Implementation ..................................... 26
## 3 Best Practices

### 3.1 Desired Characteristics of Best Practices

- **3.1.1 Innovation** .............................................. 31
- **3.1.2 Impact** .................................................. 32
- **3.1.3 Sustainable Effect** ................................. 32
- **3.1.4 Replication** ............................................ 33

### 3.2 Best Practices in Education .................................. 33

- **3.2.1 Existing Best Practices in Education** ........ 34

### 3.3 Best Practices in a CiC .................................... 35

- **3.3.1 Definition: What are Best Practices in a CiC?** .... 36
- **3.3.2 Granularity of Best Practices** .................... 37
- **3.3.3 Definition and Validation of Best Practices** .... 40
- **3.3.4 Assumptions** ......................................... 41

## 4 Measurable Recommendations .................................. 43

### 4.1 Relevant information ..................................... 44

### 4.2 Queries ...................................................... 45
CONTENTS

4.2.1 Information Retrieval .................................................. 45
4.2.2 Visual Queries ................................................................. 47
4.2.3 Queries in educational settings ........................................ 47
4.2.4 Use of XML in Education .................................................. 48

4.3 Application-specific and XML-based Queries ............................ 48
4.3.1 Equivalence of Internal Model and XML representation .......... 49
4.3.2 Querying on Application’s Internal Model .............................. 50
4.3.3 Querying on XML Representation ........................................ 51

4.4 Processes to define Best Practices ........................................ 52
4.4.1 FLOSS: Bazaar-style Software Development .......................... 52
4.4.2 Collaborative Content Creation ........................................... 53

4.5 Identification of Situations .................................................. 53

4.6 Products .............................................................................. 54
4.6.1 CiCv2 .............................................................................. 54
4.6.2 Deliverables ...................................................................... 55

5 CiCv2: A Flexible CiC Implementation ........................................ 57
6 Query Model and Implementation

6.1 Relevant Information ................................................. 74

6.2 Abstract Query Model .................................................. 75
   6.2.1 Query .............................................................. 76
   6.2.2 Query Resource Provider ....................................... 77
   6.2.3 Query Connection ................................................ 77
   6.2.4 Query Connection Link .......................................... 78
   6.2.5 SubQuery ........................................................... 79

6.3 Query Context in CiCv2 ................................................. 79
   6.3.1 External data: QueryResourceProvider ......................... 80
   6.3.2 Query Composition: QueryConnection and QueryConnectionLink .. 84

6.4 Basic building blocks for queries .................................... 85
   6.4.1 CiC Query .......................................................... 85
   6.4.2 Constant Query .................................................... 86
   6.4.3 Query Result ...................................................... 86
   6.4.4 File Query ........................................................ 87
   6.4.5 Current Document Query ......................................... 88
CONTENTS

7 Query Usage Scenarios

7.1 Common Queries .................................................. 112
  7.1.1 Classroom Snapshot ............................................. 112
  7.1.2 List Students .................................................... 113
  7.1.3 Current Page from Student .................................... 113
  7.1.4 Specified Page from Student ................................ 114
  7.1.5 Diff Count ...................................................... 114
  7.1.6 Opened File ..................................................... 114
  7.1.7 Student Activity Indicator .................................... 115
  7.1.8 MatchMaker Collaborations list ............................. 115
  7.1.9 MatchMaker Activity Indicator .............................. 115
  7.1.10 Model Complexity ............................................. 116

7.2 Java Programming .................................................. 116
  7.2.1 Java Reference Frame .......................................... 116
  7.2.2 Session Plan ..................................................... 118
  7.2.3 Related Queries ................................................ 118
  7.2.4 Discussion and Results ....................................... 122
**CONTENTS**

7.3 Stochastic Modelling ......................................................... 123

7.3.1 Stochastic Reference Frame .............................................. 123

7.3.2 Session Plan ................................................................. 124

7.3.3 Related Queries .......................................................... 126

7.3.4 Discussion and Results .................................................... 126

7.4 System Dynamics Modelling .................................................. 127

7.4.1 System Dynamics Reference Frame .................................... 128

7.4.2 Session Plan ................................................................. 130

7.4.3 Related Queries .......................................................... 130

7.4.4 Discussion and Results .................................................... 130

8 Discussion and Conclusions ...................................................... 133

8.1 Methodology to Define, Validate and Apply Best Practices ......... 135

8.2 Future Work ........................................................................ 136

8.2.1 Generating repositories Courseware including Best Practices .... 136

8.2.2 Applying the Queries in other Learning Environments .......... 137

8.2.3 Improving the Query Development process ....................... 137
CONTENTS

8.2.4 Defining strict typing for Queries ........................................... 137
8.2.5 Integration of external Processing ............................................ 138
8.2.6 Interactive Queries ............................................................ 138

A Examples ................................................................. 139
A.1 Definition and Validation of Best Practices .............................. 139
A.2 Application of Best Practices ................................................. 142

B Implementation of sample Queries ........................................... 143
B.1 “Classroom Snapshot” Implementation ...................................... 143
B.2 “List Students” Implementation ............................................ 145
   B.2.1 XQuery 1 of part (b) ................................................... 145
   B.2.2 XQuery 2 of part (b) ................................................... 145
B.3 “Current Page from Student” Implementation .......................... 147
   B.3.1 XQuery of part (b) .................................................... 147
   B.3.2 XQuery of part (c) .................................................... 147
B.4 “Specified Page from Student” Implementation ....................... 149
   B.4.1 XQuery of part (b) .................................................... 149
<table>
<thead>
<tr>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.4.2 XQuery of part (c)</td>
</tr>
<tr>
<td>B.5 &quot;Diff Count&quot; Implementation</td>
</tr>
<tr>
<td>B.5.1 XQuery of part (c)</td>
</tr>
<tr>
<td>B.5.2 XQuery of part (d)</td>
</tr>
<tr>
<td>B.5.3 Sample output of part (d)</td>
</tr>
<tr>
<td>B.6 &quot;Opened File&quot; Implementation</td>
</tr>
<tr>
<td>B.6.1 XQuery 1 of part (b)</td>
</tr>
<tr>
<td>B.6.2 XQuery 2 of part (b)</td>
</tr>
<tr>
<td>B.6.3 XQuery of part (c)</td>
</tr>
<tr>
<td>B.7 &quot;Last Activity&quot; Implementation</td>
</tr>
<tr>
<td>B.7.1 XQuery of part 1</td>
</tr>
<tr>
<td>B.7.2 XQuery 1 of part 2</td>
</tr>
<tr>
<td>B.7.3 Sample result for XQuery 1 of part 2</td>
</tr>
<tr>
<td>B.7.4 XQuery 2, part 2</td>
</tr>
<tr>
<td>B.8 &quot;List MatchMaker Collaborations&quot; Implementation</td>
</tr>
<tr>
<td>B.8.1 XQuery 1</td>
</tr>
<tr>
<td>B.8.2 XQuery 2</td>
</tr>
</tbody>
</table>
CONTENTS

B.9 “MatchMaker Activity Indicator” Implementation .............................................. 164

B.9.1 XQuery 1 ........................................................................................................... 164

B.9.2 Sample output of XQuery 1 .............................................................................. 166

B.9.3 XQuery 2 ........................................................................................................... 166

B.9.4 Sample output of XQuery 2 .............................................................................. 167

B.9.5 XQuery 3 ........................................................................................................... 170

B.10 “Model Complexity” Implementation ............................................................... 171

B.10.1 XQuery 1 of part (a) ...................................................................................... 171

B.10.2 XQuery of part (b) ......................................................................................... 174

Bibliography .............................................................................................................. 175
List of Figures

1.1 The Space-Time Matrix .................................................. 2

5.1 Teacher Application GUI ............................................... 62

5.2 Teacher Application GUI showing Discussion Board. ............. 64

5.3 FreeStyler application showing a combination of nodes and hand-written text. . . . 66

5.4 Example of User Actions Log .......................................... 70

5.5 Example of MatchMaker Log ........................................... 71

6.1 Control (red, from left to right) and Data (blue, from right to left) flow ............. 78

6.2 Signature of the method provided by all subclasses of CiCQuery ............... 79

6.3 Query Composition: 4 queries composed using visual connections .......... 84

6.4 Multiplexion Query Example .......................................... 89
LIST OF FIGURES

6.5 Result of MUX Query ................................................. 89
6.6 Complex Query. ...................................................... 98
6.7 Query Palette ......................................................... 103
7.1 Java Palette .......................................................... 117
7.2 Query specific for MatchMaker exercise ................................ 119
7.3 Stochastics Palette .................................................... 125
7.4 System Dynamics Palette ............................................. 129
B.1 “Classroom Snapshot” Implementation .............................. 144
B.2 “List Students” Implementation ...................................... 146
B.3 “Current Page from Student” Implementation ...................... 148
B.4 “Specified Page from Student” Implementation .................. 150
B.5 “Diff Count” Implementation ......................................... 153
B.6 “Opened File” Implementation ....................................... 156
B.7 “Last Activity” Implementation, part 1 ............................. 158
B.8 “Last Activity” Implementation, part 2 ............................. 159
B.9 “List MatchMaker Collaborations” Implementation ............. 163
LIST OF FIGURES

B.10 “MatchMaker Activity Indicator” Implementation . . . . . . . . . . . . . . . . . . . . . . . . . 165

B.11 “Model Complexity” Implementation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 172
LIST OF FIGURES
Chapter 1

Introduction

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. On one hand there is no need for gathering at the same time and place, and even asynchronous activities 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 communication 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...
Chapter 1. Introduction

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.
1.1. Motivation

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.
1.1. Motivation

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 inter-relations, of which the most significant are the following ones:
Chapter 1. Introduction

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 (?), 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.
1.1. Motivation

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 thoroughly, 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
Chapter 1. Introduction

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 with their work is the primary goal in a Computer-integrated Classroom (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 to compare the results to other experiences and using the best of them as guidelines will have an important impact on the learning experiences.

1.2 Hypotheses

The solution being proposed in this thesis is to define 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.

Since the processes of defining, validating and applying best practices is a long-term activity, involving the analysis of a vast number of situations, the task of accessing the information 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 3.3.3.
1.2. Hypotheses

The following hypotheses are proposed:

**Hypothesis 1:** It is possible to identify situations in a classroom, based on the information available within the system.

**Hypothesis 2:** It is possible to reduce the effort to validate Best Practices in a classroom by using a tool that identifies the possible situations within the classroom.

If it is possible to identify situations based on some objective criteria, 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. 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 needed information. 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 prove the hypotheses stated in 1.2:

1. It is possible to identify situations in a classroom, based on the information available within the system.
2. It is possible to reduce the effort to apply Best Practices in a classroom by using a tool that identifies the possible situations within the classroom.

1.4 Specific Objectives

The specific objectives can be enumerated as:

1. provide a querying architecture embedded in a Computer-integrated Classroom (CiC) environment that allows a teacher or tutor to define, validate and apply known best practices in that scenario.
2. perform a series of experiments in order to verify if the hypotheses expressed above do uphold in a real setting.
3. evaluate 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
1.5. Contributions

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, and specially in the educational arena, no systematic approach exists 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 1. Introduction
Chapter 2

Computer-integrated Classrooms

2.1 CSCW and Groupware

Applications have been developed for solitary 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 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\footnote{Although in Grudin (1990b) he criticizes the term user interface and states that instead computer interface would be more suitable.} 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 2.3).

### 2.2 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).

One of the main focus of CSCL is the development of distance learning environments. 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.

The focus of CSCL lies 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 (Lipponen, 2001). 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 (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 tutor-
ing 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).

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
2.3 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).

In the case of this thesis, context and awareness are important concepts that help to better define best practices. It can be the case that the recommendations contained in a best practices set can be useful in one context and less helpful in another.
2.4 Computer-supported Teaching and Learning

Computer support can be applied 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 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.

2.4.1 Distance learning

At present, distance learning scenarios, face-to-face teaching with a strong teacher-student relation are being replaced by the interaction among learners (peers) and learning materials. The interaction between learner and teacher, and the one among student peers are 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,
2.4. Computer-supported Teaching and Learning

(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.

2.4.2 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, 2003), 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.

2.4.3 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.
2.5. Computer-enabled classrooms

2.4.4 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.

2.5 Computer-enabled classrooms

2.5.1 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.
Chapter 2. Computer-integrated Classrooms

2.5.2 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.

2.5.3 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 (e.g. comparing courses taught in several periods, evolution over time, searching for relations among various courses) situations (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.

2.6 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).
2.6. Computer-integrated Classroom (CiC)

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 workspaces. 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
Chapter 2. Computer-integrated Classrooms

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.

2.6.1 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
2.6. Computer-integrated Classroom (CiC)

- open discussion
- guided discussion

2.6.2 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 (T³) (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.

2.6.3 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.
Chapter 2. Computer-integrated Classrooms

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).

2.6.4 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.

2.7 CiCv2: a Domain-independent CiC Implementation

The existing CiC implementations have had specific audiences and objectives in mind. However, in this thesis the subject is to use the CiC in a wider variety of scenarios, requiring to adapt to more general needs, for which the implementations are less suitable. Therefore, it is necessary to create a flexible CiC implementation, which allows its use in various educational domains and classroom situations. This implementation will be called CiCv2 from now on.

Currently, there is no possibility of incorporating new visual elements, rules nor functionalities into the CiC without modifying the application substantially. As in the CoolModes environment
2.7. CiCv2: a Domain-independent CiC Implementation

(Pinkwart, 2003), the CiCv2 will gain flexibility with the introduction of reference frames which provide visual languages, rules that govern them and functionalities related to these languages.

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 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 2. Computer-integrated Classrooms
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, thinking of 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.
Chapter 3. Best Practices

Bretschneider et al. (2001) 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, since it affords the most control over the exemplars to be included in the analysis. 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. 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.
3.2. Best Practices in Education

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 an end in 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 are dealing with general or coarse-grained best practices (see 3.3.2), 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., 2001), 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 environments like the CiC, 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

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
3.3. *Best Practices in a CiC*

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 that 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.

### 3.3 Best Practices in a CiC

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 a CiC?

In a CiC, **Best Practices** will be considered 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 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,
3.3. *Best Practices in a CiC*

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 significant for a number of different scenarios, so it makes sense to keep them available in some library. This has several advantages, of which some are technical and others 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 **Granularity of Best Practices**

The granularity of the recommendations is an important issue, as 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 improving the learning results.

If recommendations are too broad, it results 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 manage a huge set of recommendations, and the main objective, helping the teacher to find more time to dedicate to important tasks, will not be met.

A compromise exists among both possibilities: providing too much detail and number of recommendations versus the difficulty in mapping the recommendations into actual actions to be taken. 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 following examples, where the underlined words are the parameters of the recommendation:

- set a defined order on the planned activities contained on several files on the repository, first
3.3. Best Practices in a CiC

- define groups of min to max 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 numberN nodes and numberE 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 do not see any confusion on the existence of additional parameterized measurements.

- (using the Java Reference Frame (see 7.2.1 on page 116)) 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 7.3.1 on page 123)) find one solution to the Stochastic Model that is not making use of the Lottery coupon filter.

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
Chapter 3. Best Practices

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 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 are necessary to specify the recommendation, and that are 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.

3.3.3 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 them accordingly. These activities form part of the validation of best practices, a requirement before thinking of wide adoption.

Defining the best practices for a given classroom situation is a complex task that requires the coordinated work of a multidisciplinary team. 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,
3.3 Best Practices in a CiC

or if there appears to be no relationship about one characteristic and the result of the experience. Also, rebating candidates can be essential: if it is possible to find a case in which a supposedly good practice was followed, but the result was not as expected, other factors should be considered and the candidates position challenged.

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?

3.3.4 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.
Chapter 3. Best Practices

In order to aid educators in each of these phases, all 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 determine whether the recommendations have been followed.

**Assumption 2:** each recommendation forming part of the Best Practices has a measurable objective.
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. It will supposed that some kind of system is used to manage the courseware material both inside and outside the classroom. Using the information available in such an environment, it should be possible to identify common patterns in both suc-
cessful and unsuccessful experiences and verify the correct application of current best practices in any session.

A specially developed environment will be used, called Computer-integrated Classroom (CiCv2, 2.7). This environment is based on the general idea of a CiC (see 2.6 on page 22) in which the students work, using collaborative activities and being guided by a teacher. 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, 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.
4.2 Queries

In section 3.3.1 (page 36) 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 help her to apply the best practices.

4.2.1 Information Retrieval

Information Retrieval (IR) is a field in computer science that has received a vast amount of attention and can be considered mature, yet there is still a lot of activity and interesting research happening in this area. The most mature area within IR is the computer-centered view, focusing mostly on creating efficient structures and algorithms to index, rank and process queries to increase performance and objective quality. The most active areas in research are mostly the changes in the nature of the information used in IR systems as triggered by the World Wide Web, and the human-centered view, including human needs and perception as well as natural language that have to be considered to effectively satisfy the requests that will be made to the system.

As a result, it is possible to identify on one hand mature and useful models like entity-relation (Chen, 1976) and relational (Codd, 1970), for which the SQL language is the de-facto standard
Chapter 4. Measurable Recommendations

to access the information. On the other hand, there are newer models including Object Oriented, Multimedia and XML-based databases. Information retrieval has evolved starting from searching for exact occurrences of a pattern in text and determining matching documents in a collection to end up with complex queries and multimedia databases. Also, the requirements for various information retrieval systems can be quite varied. If for instance in one case the incorrect inclusion or exclusion of one element within the final result may be interpreted as total failure of the query, in another context that query might be perfectly reasonable (Baeza-Yates and Ribeiro-Neto, 1999). According to Chávez et al. (2001), searching for elements which are close to a given query element under some similarity criterion has its motivation in applications such as Query by Content in Structured Databases, Query by Content in Multimedia Objects, Text Retrieval, Computational Biology, Pattern Recognition and Function Approximation, as well as Audio and Video Compression.

Most query languages use some form of algebraic description of the request, and the result is more or less a direct output of the retrieved information, in some standard format. In the relational database case for example, the result is naturally a table, in some predefined format that enables the user to identify each tuple of the result and the elements of each column. However, some systems offer a possibility to define the queries in other ways, and to return the results in less standard structures. Some of these systems use the concept of query by example (Zloof, 1975), that is regarded as an easier to learn language. In this language, the user forms a pattern, preferably in a visual way, in order to show what the results are expected to look like. This enables a direct interaction with the expected results rather than handling the formal languages and interpreting the results. Boolean queries are traditionally considered in this field, and in Echeverría and Pino (1989) an early example of boolean queries expressed visually can be found.

Normally, a person searching for multimedia information has to do just as if she was searching for textual information (Schauble, 1997). This is related to the fact that pictures seldom appear without some collateral text like captions or other descriptions (Srihari et al., 2000). An alternative is to use e.g. an image and performing a search for similar images or videos, a kind of query by example. However, it is still difficult to go further and be able to extract semantic information from the images, which could be indexed for a more efficient retrieval. Research on Multimedia Information Retrieval has been carried out within several fields of Computer Science, including
4.2. Queries

information retrieval, digital signal processing, database communities as well as artificial intelligence (Meghini et al., 2001). Every approach has its own advantages and can help to improve the results of the queries.

4.2.2 Visual Queries

One important aspect of Information Retrieval, as already mentioned, is the definition and efficient processing of queries. One particular form of defining queries is using some sort of visual language that helps in many situations to define the queries in a comfortable way for the user. This is the case of query by example, which is inherently associated to a graphical display of the query, even if this is not always the case. Recently, some efforts have been made into creating a graphical querying system for XQuery (Braga and Campi, 2003; Augurusa et al., 2003; Comai et al., 2001), in which the familiar tree structure should facilitate the task of defining the query. The result in this case is a text equivalent to the graphically created query.

This kind of queries can provide some help to the author that should provide basic queries that will be later on used by the teacher and combined with other queries, but it is not useful for the visual presentation in the teacher’s case. In the case of the teacher, it should not be necessary for her to know the structure of the XML files that are generated by the application, neither the XML code corresponding only to the domain-specific visual language that this application uses. Therefore, it is necessary to provide another, higher-level visual representation for the queries that do not force the teacher to know about parts of the system that have no relation whatsoever to the purpose of using the system.

4.2.3 Queries in educational settings

As digital educational materials are being deployed widely and a person has access to several sources for accessing them, it becomes necessary to locate material that suits specific activities and needs. This problem is being tackled mainly by providing a standardized way to describe educa-
tional material by way of metadata. Several standards aim to provide a common way to describe learning objects, such as IEEE LOM (IEEE, 2002) and IMS Metadata (IMS Global Learning Consortium, 2004). Based on this information, it is possible to search for material that may be valuable in a particular situation, as is the case in the Edutella project (Nejdl et al., 2002), where relevant information is located through the use of queries on the metadata.

4.2.4 Use of XML in Education

In many educational settings, XML is being used to store the information and meta-information for courses and other data. This is the case of the Learning Objects Metadata (LOM) standard (IEEE, 2002), which intends to provide a way to facilitate the exchange of digital learning material through the provision of meaningful and searchable as well as easily indexable information along with that material.

Another important language being used in education is SMIL (Synchronized Multimedia Integration Language) (Hoschka, 1998), which is used to represent animations. Likewise, the data managed in the Active Documents (Verdejo et al., 2002) are also completely managed in XML.

As more and more educational material and environments use XML as a primary way to exchange information in the form of documents, web services and others, it makes more sense to use that standard as a base to create the tools to use in educational settings. The tools proposed in this thesis build on XML and can therefore be used in a myriad of environments and settings in the future.

4.3 Application-specific and XML-based Queries

The information available inside a computer-supported environment like a CiC can have several formats. In the case of the CiCv2 (2.7), two different instances for the same document can be
4.3. Application-specific and XML-based Queries

found, depending on its status:

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). The document is available in its serialized XML format.

2. Document instance opened by an application: the document is being edited within a classroom session. An outdated version of the document may exist in the repository, but as the interaction is synchronous, the interesting information 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.

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 state. In the first case the document files would be accessed directly, using the XML structure to perform any desired action. In the second case, the internal data model would be used inside the application 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 queries. It is possible to treat documents in both states the same way, just transforming them from one state to the other. This allow us to save a huge amount of work and simplify all queries.

4.3.1 Equivalence of Internal Model and XML representation

Since the XML file that gets saved is basically a serialized version of the application internal model, it can be concluded that they are equivalent. It is possible to create the XML representation starting from the internal model, and vice-versa, regenerate the internal model starting from the written XML file.
However, some portions of the internal model may contain more information. 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 data. In the XML representation, only the corresponding Model is used, and in the application, the View is also available. Moreover, the Model can be accessed in the application only through the reference present inside the View. This view includes data about its size, which could be a useful parameter to be used in queries. But one of the characteristics of the MVC pattern is that in some cases, the view can be changed, so the size that results when instantiating the model does not have an absolute meaning.

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 among both representations is achieved. It will thus be possible to use any of both representations to process the queries with essentially the same results.

The transformation of one representation to the other is not quite equivalent. Due to the processing necessary to set up the View and Controller part of each modelled object, the transformation from the XML representation into the representation used internally by the application is more complex. In tests, the process of parsing a XML Document into FreeStylers internal structure took several orders of magnitude longer than generating the XML structure (as a Document Object Model – DOM – or plain text file) from the application’s model. 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.

### 4.3.2 Querying on Application’s Internal Model

When using the applications internal model, it is possible to take advantage of the objects defined inside the application. These include helpful methods that provide access to aggregate information, execution of methods and some other specific properties. The same information is also available through the XML representation as just presented, but ease of use should be considered.
4.3. Application-specific and XML-based Queries

On the downside, currently there are no existing tools that might be used to directly work on the data as present inside the application. In order to perform any query, all tools would have to be developed from scratch or heavily adapted in order to be usable. When developed, these tools would not be suitable to use outside of the CiC environment, and any interaction with tools from the outside would require some transformation for the communication of data to work.

4.3.3 Querying on XML Representation

Using the XML 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 (4.2.4), 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 attack the problem and it can be adapted to other XML documents relatively easily. Using XML it is possible to interact with the “outside world” in both ways: sending XML documents or fragments to an outside application for processing, and 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, within a distributed scheme (see 6.1) such as the one proposed, it will also be necessary to transport the intermediate results to the final destination. Using XML it is trivial to send e.g. the document fragment, since there are standard ways to transform the Document Object Model into a serialized string that can be sent through a network connection and turned back into an equal Model on the receiving end. Using an internal model of the data would mean to take into account an additional serialization / deserialization process in between. Currently, FreeStyler provides the means to serialize / de-serialize using an XML format, so the easiest way to transport intermediate results would probably based on that format. Hence, directly using the XML format is the most straightforward way to start extracting information, specially when thinking about communication with other tools and resources.
4.4 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., 2001). Before thinking about defining, validating and adapting best practices inside computer-enabled classrooms, the process through which best practices can be reached and re-evaluated need to be defined.

The processes through which Best Practices can be defined and validated are necessarily collaborative processes, because the Best Practices have to be applicable in a wide range of scenarios. The comparison of how several independent applications of the Best Practices is the main motivation of defining them in the first place. 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.

4.4.1 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)
- “Every developer scratches its own itch”
- “Given enough eyeballs, all bugs are shallow”
4.5. Identification of Situations

4.4.2 Collaborative Content Creation

4.5 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 2.3), 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 will provide. If the information needed by the 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 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.
Chapter 4. Measurable Recommendations

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.

4.6 Products

The system to be implemented builds on the concept of a Computer-integrated Classroom (CiCv2) as described in 2.7 on page 26, 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 3.3.3.

4.6.1 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
4.6. Products

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.

4.6.2 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 includes the ability to include domain-specific modeling and is not tied to any teaching environment in particular. The CiCv2 will help the teacher actively to 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.
   - **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.
Chapter 4. Measurable Recommendations

- **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 will improve the proposed best practices until they reach a level of maturity.

It is expected to be able to determine 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 progresses, 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 that produce a new document as their output.
Chapter 5

CiCv2: A Flexible CiC Implementation

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 situation or continue with the next step in the planned schedule.

The same 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. In order to be useful, these queries should focus on the subject- and domain-specific aspects, in order to provide output that 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 using 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 several more uses. 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 as well as more elaborate queries that can be composed, creating new queries (see 6.5). The composition is made visually, just as the rest of the interaction in the CiCv2, using drag and drop and similar constructs, making the interaction with this framework familiar to all of the involved users.

The system to be implemented builds on the concept of a Computer-integrated Classroom described in 2.6 on page 22, 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 evaluate the 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 implementation, CiCv2, consists of three modules that interact with each other. One is the central repository, available from inside and outside the classroom sessions. The other modules are the ones used by teacher and students, respectively.

### 5.1 Repository

CiCv2 uses a repository that manages several tasks:
5.1 Repository

- User Authentication
- Session Directory
- File Repository

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

5.1.1 Security and Authentication

Before any interaction takes place, the users of CiCv2 have to authenticate at the repository. The repository contains information about the users, its 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\textsuperscript{1} is generated. All of 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 user is verified using the X500Principal object and the requested target. As will be seen further on, the target can be of several kind, including the execution of some query, retrieval or uploading of a file, or request for information.

In order to guarantee that users will be granted access only to authorized files and resources, features of the java.lang.SecurityManager, provided Java language, are used. Prior to accessing some resource, the Java libraries (part of the Java Virtual Machine or JVM that executes the

\textsuperscript{1}The complete name is javax.security.auth.X500Principal
Java code) verifies the access restrictions the JVM may be imposing on the user. 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.

- It is necessary for the users 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 she 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.

### 5.1.2 Session Management

When starting a session, the teacher authenticates at the repository, and afterwards creates a session for 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.
5.2. Student and Teacher Applications

5.1.3 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.

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`. This class provides all of 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.

A user starting the Student or Teacher Application will be requested for a username and a password for authentication at the repository. Once the authentication has been successful, the user selects a session or decides to work in remote mode, in which case no session is used. The graphical user interface (GUI) presented to the user once the authentication and session creation/selection has been passed includes several elements as seen in figure 5.1:

- Menubar: several actions are available through this menu, most notably the Transfer, FreeStyler and View menus. The menu titled "Allow" is available only in the Teacher Application.
- Log: underneath the Menubar is the Log Panel, which keeps a record of the latest events inside the classroom session.
- Inbox: to the right 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 respective message is executed.

- Discussion board: visible in figure 5.2, it allows teacher and students to exchange text messages.

- Status bar: at the bottom of the GUI. It contains the last event or output registered. It gives a user feedback that a selected task has been executed successfully or that some error occurred.

The teacher application also maintains a MatchMaker server that is used to share content (see 5.5.3).

5.2.1 File Transfers

The Student and Teacher applications implement several forms of file transfer actions, according to the purpose of the transfer. 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 of the students participating in the active session, and they are notified of the availability of the newly sent file.
5.2. Student and Teacher Applications

- **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 that 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**: 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 executed automatically.

- **Send file(s)**: Similar to the ”Distribute Assignment” action. The teacher selects a source file and any number of students to which the file will be sent.

- **Get file(s)**: Allows the teacher to retrieve files from the students inside a face-to-face classroom session.

In each case, a file transfer needs a selection of one or a group of sources and destinations.
5.3 Discussion Board

The CiCv2 GUI allows text-based teacher-student communication. The students can send messages to the teacher, and the teacher can send messages to one or a group of students.

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). But when the teacher allows this feature, the students can start sessions on their own. This is specially useful when giving the students freedom to choose their own groups.
5.5 FreeStyler

for some collaborative activity using the computers. It is not necessary for the teacher to
define the groups, but the students can organize themselves.

5.5 FreeStyler

FreeStyler (Hoppe and Gaßner, 2002) is a flexible application that combines freehand annotations
with visual language elements and can be used in different 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 a 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 5.3 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.

5.5.1 JGraph

JGraph (Collide, 2003) is 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
Chapter 5. CiCv2: A Flexible CiC Implementation

Figure 5.3: FreeStyler application showing a combination of nodes and hand-written text.
5.5. *FreeStyler*

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 makes the restriction mentioned before less important.

The JGraph allows two basic types of objects to be used, as expected for a graph: Nodes and Edges. The Nodes can have any level of complexity, and are interconnected by the 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 this way it is possible to define complex simulations of following the particular characteristics of a specific domain. As an example, implementations of visual languages for modeling petri nets and robots escaping a maze, among others, 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 if it is 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 of the users in the same MatchMaker session are replicated to the other applications, so the other users can see the changes almost instantly. This is an important resource for collaboration inside a classroom, by sharing all of the 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 and has been integrated into the CiCv2 as the default tool to
5.6. Plug-in Mechanisms in FreeStyler and CiCv2

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.

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

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, in a face-to-face classroom session the interesting data 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 make 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 information requested 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 identify 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.
Chapter 5. CiCv2: A Flexible CiC Implementation

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 later analysis.

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 5.4 presents a simplified example of the data logged in this type of log.

5.7.2 Standard and Error output

The application and any active Plug-in has access to Java's standard output `java.lang.System.out` and standard error `java.lang.System.err` streams. To have access to this valuable information,
5.7. Logging

<?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>
</SyncActions>

Figure 5.5: Example of MatchMaker Log

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.

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. This history contains all relevant actions and changes done on pages that are shared using MatchMaker collaboration. Figure 5.5 shows an example
Chapter 5. CiCv2: A Flexible CiC Implementation
Chapter 6

Query Model and Implementation

In order to prove the hypothesis stated in 1.2, a methodology to model existing best practices has been defined. This methodology follows the assumptions defined in 3.3.4. 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.

To prove this concept, a new version of a Computer-integrated Classroom environment, CiCv2 2.7, has been developed. 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 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.

Using CiCv2 along with the querying system, several best practices can be implemented and later used in other contexts, 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.
Chapter 6. Query Model and Implementation

6.1 Relevant Information

As already has been implicitly stated, the information sources on 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 more.

- **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.

  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, 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.
6.2. Abstract Query Model

From this information it can easily be learned if students are interacting, what 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 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. But 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.

6.2 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 4.3, the information available to the queries is in XML format, and it make 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 of the information sources described in 6.1. Considering these requirements, following concepts are defined: Query, Query Resource Provider, Query Connection, Query Connection Link and SubQuery.

6.2.1 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 6.1. The other part consists of access to several context-dependent resources, as detailed in 6.3.1.

Atomic queries

Atomic queries are those which are not composed of other queries. They take as input one XML stream, perform some modifications to it, acting like a filter in the Unix sense, and finally 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, however. In particular, it is possible to define multiplexer and demultiplexer queries so that all others use these instead of having multiple inputs and outputs. Having done this, almost all queries could be described in terms of a languages like XSLT (Clark, 1999) or XQuery (Boag et al., 2001). In the queries used in this thesis, most of the processing is done using the XQuery language.
6.2. Abstract Query Model

An atomic query can be as complex as wanted, as long as it does not involve other queries. But if it is possible, it should be preferable to separate a query into several sub-queries, since it allows more possibilities for reusing parts of that query in other ones.

6.2.2 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 from another query. All of this information is accessed through a Query Resource Provider.

6.2.3 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 par 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) through to the last (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.

6.2.4 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...
6.3 Query Context in CiCv2

CiCQueryResult execute(QueryConnection conn, QueryResourceProvider prov);

Figure 6.2: Signature of the method provided by all subclasses of CiCQuery

to order several inputs in some repeatable way. But other queries (see for example the Multiplexion Query in 6.4.6) may use this label as if it is part of the input, so the output will depend on its value.

6.2.5 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.

Using the Composite Design Pattern as described in (Gamma et al., 1994), it is possible to compose queries, forming a new query as the result. 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 different 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 Query Context in CiCv2

According to the definition in 6.2.1, all information a query can access is either available as XML input (cf. 6.2.3) or through requests to a Query Resource Provider (cf. 6.2.2). In the CiCv2 implementation, this maps to the definition of two interfaces: QueryConnection (see 6.3.2) and QueryResourceProvider (see 6.3.1). Using these interfaces, a query is executed using the execute method as defined in figure 6.2. This way, every execution of a query has access to all of the relevant context through these defined interfaces.
6.3.1 External data: QueryResourceProvider

To make the relation of a query to its context as clear as possible, a Java interface called QueryResourceProvider is defined. All of the information a query needs to execute and that depend on the external context, are provided through the QueryResourceProvider interface. 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 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 other stored files. Some of these elements do not make sense in all context, in which case the returned object is null.

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

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

Errors are logged and need to get sent to a central location where they can be handled.

public java.util.Vector getStudentHandles();

The Teacher application maintains a list of all students connected to the current CiCv2 session. This method returns a Vector of StudentHandles inside the Teacher application, in the other contexts it just returns the null object.

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

When a query intends to open a file for reading, it does so through this method. 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.


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.
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.

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 6.4.9), which uses a saved JGraph (see 5.5.1) to manage the sub-queries.

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.

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

Takes a query coded in a String in its XML form, sends it to the repository where it is executed, sending the result back. It returns the result of the execution at the repository. In the repository implementation, this method simply executes the query, since it already is located in the repository context.
6.3. Query Context in CiCv2

**public java.io.File getCiCLog();**

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

**public java.io.File getStdLog();**

As the previous method, it returns an opened file to the Standard and Error output of the Teacher or Student application (see 5.7).

**public java.io.File getMmLog(String name);**

For each opened Matchmaker collaboration, a new logfile is opened in which all actions are logged (see 5.7). Since the logging is performed on the Student application, where the MatchMaker server resides, this method returns the `null` object in the Student application and repository.

**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.
6.3.2 Query Composition: QueryConnection and QueryConnectionLink

Using Java interfaces named QueryConnection and QueryConnectionLink, the relationships among queries are defined. 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 seen in figure 6.3.

Consider one example in which the Query Composition is made visually combining queries. In figure 6.3, 4 queries have been connected by QueryEdges. These edges represent the data flow, the query labeled "Student has Opened File" can be identified as the first query. Next comes a query labeled XQuery, followed by another one of the same type, and finally the last query, labeled "ObjectCreation". The execution of the queries follows the other way around, starting from the last query ("Object Creation"), which is the one that displays the Exec button. When this button is pressed, the query to be executed needs to fetch its input, triggering the execution of the preceding query. Thus, the first query to be executed is the one identified as the first above, following the same stack-like execution found in most programming languages.
6.4. Basic building blocks for queries

When connecting two queries using a QueryConnectionLink, optionally a label can be added in order to identify that connection. 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 as to avoid confusion. However, the existence of inbound data connections is not controlled at this point. See 6.5 for details about the visual control and development of queries.

6.4 Basic building blocks for queries

In this section the implementation of several atomic queries is described, following the model presented in 6.2.1. These implementations are independent of the method used to link several subqueries together using a Query Connection (cf. 6.2.3) 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.

6.4.1 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. In itself, this class has no functionality and is not used as a query in itself.

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 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.

### 6.4.2 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

### 6.4.3 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 following constructors:

- public CiCQueryResult(java.lang.String result);
- public CiCQueryResult(org.w3c.dom.Node result);
6.4. Basic building blocks for queries

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

- **public String asString():** 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():** 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.

6.4.4 File Query

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

- Predefined Filename: a name that is defined in advance of 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.

For opening the files, the `getFileReader( ... )` method from the `QueryResourceProvider` (cf. 6.3.1) is used. Some special filenames are defined, that have particular meanings:
6.4.5 Current Document Query

Using the getCurrentDocument() method, an XML representation of the current Document is returned. 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.

6.4.6 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 6.4, the output generated is the one displayed in figure 6.5.
6.4. Basic building blocks for queries

Figure 6.4: Multiplexion Query Example

Figure 6.5: Result of MUX Query
6.4.7 Diff Query

The concept of differences between documents is widely used within the version management arena and Unix community (MacKenzie et al., 2003). The differences are expressed as the smallest amount 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 differences 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 different 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 differences 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.
6.4. Basic building blocks for queries

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.

Implementation

The Diff Query implementation is based on the work described in 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 among 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
  </part>
</file>
```
Chapter 6. Query Model and Implementation

The X-Diff implementation detects 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 \textit{cic-diff-query-delete}, indicating the type (\textit{subtree}) in a parameter. In the following example, the tag "filename" and its subnodes was removed from the original, the result is:

\[
\text{<title>Part 1</title>}
\text{<filename>part1.xml</filename>}
\text{<author>jharding</author>}
\text{<length>6834</length>}
\text{</part>}
\text{</file>}

\[
\text{The X-Diff implementation detects following changes to a XML structure:}
\]

- 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 \textit{<cic-diff-query-insert type="subtree" tag="tagnname"/>} where the tagnname refers to the inserted subtree. When a new subtree, consisting of the XML fragment \textit{<editor>jhp</editor>} after the tag \textit{author}, the result is the following:

\[
\text{<title>Part 1</title>}
\text{<filename>part1.xml</filename>}
\text{<author>jharding</author>}
\text{<length>6834</length>}
\text{</part>}
\text{</file>}
\]
6.4. Basic building blocks for queries

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

```xml
<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"/>
        </editor>
    </part>
</file>
```

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

```xml
<file>
    <part number="1">
        sample text
        <title>Part 1</title>
        <filename>part1.xml</filename>
    </part>
</file>
```
• Modification of a textnode. When changing the text inside the length tag from 6834 to 6835, the result is:

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

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

```xml
<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:
6.4. Basic building blocks for queries

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

```xml
<file>
  <part number="2">
    <cic-diff-query-update name="number" type="attribute">
      <from>1</from>
    </cic-diff-query-update>
  </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 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.
6.4.8 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.

The implementation of the XQuery needs a text that includes the XQuery expression, and a XQuery processing engine. Since the processing engines are still in a development phase, it was chosen to include several implementations that the user can choose from at run-time. 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 of the involved engines.

Currently, the XQuery engines used in the XQuery implementation are the detailed next:

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

6.4.9 Complex Query

A complex query is used to group a sequence of queries linked together by QueryConnectionLinks and hide the complexities of that query from the user. Using this mechanism an arbitrarily complex query can be taken, composed of any number of queries, and it will look as simple as any other 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 6.6
6.4. Basic building blocks for queries

shows an example of a Complex Query, where the sub-queries corresponding to the complex query marked as (a) are being shown in 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 subqueries when the former have not been provided.

When a Complex Query is executed, the first and last query are identified inside the grouped subqueries. 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.

SubQuery

The SubQuery interface corresponds to the definition of SubQuery in 6.2.5. It binds a Query to the notion of its neighbours through additional QueryConnections.

6.4.10 Context Changes

As already mentioned, any query, and complex queries in particular, can be executed in any context. Making use of this, two queries have been defined that, when executed, change the context in which the contained queries are to be executed.

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.
Figure 6.6: Complex Query.
6.4. Basic building blocks for queries

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

```xml
<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, in order to take advantage of this architecture, it is necessary that upon requesting the execution of the query from the first student, the Teacher Application part of the execution does not wait for the result before requesting the execution on the next student, since this approach would be equivalent to a sequential execution, where several processors perform the tasks, but only one at a time.

To overcome 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 waits for the result. The Teacher Application collects the results as they arrive, and 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 issue. It also means that the result of a StudentQuery may not include the details for every student registered in the session.

### 6.4.11 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 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.
6.4. Basic building blocks for queries

6.4.12 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 implemented for the CiCv2 queries has two basic modes of operation, with a default fallback 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, presumably in the form of a FreeStyler XML document.

However, 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 6.1.
6.4.13 Timed Query

This 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 desired interval 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 in the defined interval.

6.5 Query Reference Frame

The abstract model defined in 6.2 has been implemented in two stages. First, the functionality of the query itself was developed, as has been described in 6.4. This corresponds to the Query definition (6.2.1). In the CiCv2, the Query Connection, Query Connection Link and SubQuery constructs have been implemented using the concept of a reference frame (cf. 2.7), as a visual language for constructing queries.

Following the Model-View-Controller pattern as described in 5.5.1, the query 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 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.
6.5. Query Reference Frame

Figure 6.7: Query Palette
In a JGraph, the information about neighbour 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 of giving access to the query implementation and to manage the connections to neighbour queries is fulfilled.

6.5.1 Data types

Although all the data used in the system is represented as 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 representations of each intermediate result. It is recommended to define several data types, 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. In the stage the development is being done in the context of this thesis, 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, it is expected that the benefits of defining such categories will outweigh the costs, specially when non-technical users start using the system daily.
6.5. Query Reference Frame

6.5.2 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 of the capabilities 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.

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 different 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 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 optionally a text description.
Chapter 6. Query Model and Implementation

The icon is defined for the queries that simply group subqueries together and also for the ones that change the context for the subqueries Student Query and Repository Query (6.4.10). This way, the user is always aware of the context in which the subqueries 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 subqueries, 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.

6.6 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
6.6. Example

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. To determine whether the activities are being carried out successfully or not, based on the information available in the classroom environment following aspects can be considered:

6.6.1 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.
6.6.2 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 determine 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 of 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 compared with the other possibilities. 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
6.6. Example

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.

6.6.3 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.

6.6.4 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 in a very important aspect: the teacher has no additional information than the documents delivered 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.
Chapter 7

Query Usage Scenarios

In order to validate or refute the hypotheses stated in 1.2, several experiments were carried out. The experiments were conducted at the Universidad de Chile, with students of an undergraduate course called “Computación para el Trabajo Grupal” (Group-Work Computing). The participating students were all either following a computing engineering (Ingeniería Civil en Computación) curriculum or in graduate programs.

Before starting with the described sessions, a preliminary session was conducted to introduce necessary concepts and basic knowledge about the FreeStyler (cf. 5.5) and CiCv2 (cf. 2.7). The classroom sessions were performed in a 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, as well as the logfiles of teacher, students and repository. Additionally, using a query described in the next section, a periodical snapshot of the state in the system was taken, including all of the students currently opened documents.
Chapter 7. Query Usage Scenarios

7.1 Common Queries

Some of the queries used in the classroom experiments were used in several sessions. These queries are not domain-specific, so they could be applied in several settings where the domain of the subject differs from each other.

7.1.1 Classroom Snapshot

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

The name of the file into which the document is saved has the form of 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. 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 period after which the query is repeated will define the accuracy or resolution of the simulation. Using the data gathered during the three experiments described further on in this chapter, it was found that using a period of 30 to 90 seconds, depending on the task, is enough to give the necessary information without accumulating an excessive number of identical documents on the repository.
7.1. Common Queries

7.1.2 List Students

Even when the teacher has a list of connected students available directly 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

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 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 B.2 on page 145.

7.1.3 Current Page from Student

The teacher can retrieve the currently opened page from a predefined student. In order to define which student to query, a ConstantQuery is used to give that input, which 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. Generally however, the teacher will use other queries to gather information and, based on this information, choose to take a closer look at the work of certain students.
Chapter 7. Query Usage Scenarios

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, allowing her to modify it without the changes propagating to the student since it is a local copy, displaying it for the whole class and taking it as an example, etc. Implementation details for this query are available in B.3 on page 147.

7.1.4 Specified Page from Student

Like the previous query, this 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 B.4 on page 149.

7.1.5 Diff Count

This query calculates the differences found between a specified file in the repository and the current document for each student. Then the number of changes among both documents are counted, separated in deletions, updates and inserts. Implementation details are described in B.5 on page 151.

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.
7.1. Common Queries

Implementation details are described in B.6 on page 155.

7.1.7 Student Activity Indicator

This query presents the activities seen by each student during the last couple of minutes. The considered time interval can be configured, and in these experiments 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 actions (through the use of the menu bar and buttons) performed. Implementation details and sample output are available in B.7 on page 157.

7.1.8 MatchMaker Collaborations list

Using this query the teacher gets a list of all active MatchMaker collaborations within the current CiCv2 classroom session. For details about the implementation and a sample output of this query, see B.8 on page 162.

7.1.9 MatchMaker Activity Indicator

By providing the name of a specific session, this query creates a table that shows the students that are actively participating in the session, and a summary of the total number of interventions, as well as a breakdown by type in ActionExecuted, ObjectCreated and ObjectChanged. For implementation details, see B.9 on page 164.
Chapter 7. Query Usage Scenarios

### 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 B.10 on page 171.

### 7.2 Java Programming

In this, the first experiment of the series, the objective was 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 teachers computer.

#### 7.2.1 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:

- **Java Node**: contains Java source code and has buttons that allow a user to compile and execute

- **Input Node**: a text node that is used to inject as standard input into the execution of a Java Node connected through an edge.

- **Output Node**: a text node that receives the output during the execution of a Java Node connected to it through an edge.

- **Import Node**: a node that modifies the classpath of the JVM executing the Java Node it is connected to.
7.2. Java Programming

Figure 7.1: Java Palette.
7.2.2 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.

7.2.3 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 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 advance of the student is classified into several possible alternatives. As can be seen in the example in figure 7.2 part (d), the query creates a table in which each student gets its pages classified into one of several alternatives 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, has met the minimal requirement of creating the MatchMaker session, or additionally to creating the session also verifies that the session was successfully created.
7.2. Java Programming

Figure 7.2: Query specific for MatchMaker exercise
• 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”.

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

```xml
<AnnotationNode FontSize="10" creationtime="1110487750862" creator="jhp"
    lastModifier="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 4</th><th>Page 5</th></tr>",
    "<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>
```
7.2. Java Programming

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"
    else "none"

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

7.2.4 Discussion and Results

As the teacher has an overview of value-added information, it is possible to identify when the
students are showing adequate progress on their tasks. Students that are progressing at a different
pace 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 in her peers, the teacher might want to give 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 helped to guide the class in a natural way.
7.3. Stochastic Modelling

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 much profounder as the query presented above, the result of which can be updated every couple of seconds.

7.3 Stochastic Modelling

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

7.3.1 Stochastic Reference Frame

To allow the representation and execution of the random experiments, the reference frame provides a series of elements that are classified into following categories:

- Urns: nodes that generate random elements. Examples 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 on the previous ones.
Chapter 7. Query Usage Scenarios

- **Drawing Nodes**: to fetch some value from an urn, the urn is connected to a drawing node using a drawing edge. Depending on the type of the edge, the values drawn are immediately returned back into the urn or not. 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.

### 7.3.2 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:
7.3. Stochastic Modelling

Figure 7.3: Stochastics Palette.
Chapter 7. Query Usage Scenarios

- Kino 5: 7 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: 1 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 more 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.

7.3.3 Related Queries

In a similar way to the previous scenario, it is possible to create a specific query that helps to identify whether a solution is being provided successfully. Moreover, in this case it is possible to solve the “Toto 3” problem 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.

7.3.4 Discussion and Results

In this exercise, the following two aspects are relevant: 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.
7.4. System Dynamics Modelling

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. By direct observation in the classroom and the video sequences, it can be seen 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 mayor 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 was to be expected, several users made the mistake to use a 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.

7.4 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, medicine (spread of diseases) and others.
7.4.1 System Dynamics Reference Frame

As can be seen in figure 7.4, the System Dynamics reference frame has two basic types of nodes: Stock nodes and Rate nodes. The Stock nodes maintain the current value of a variable inside the model. The Rate nodes specify the amount that is to be transferred 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 calculate 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-alike shape. The second has an unlimited stock, whereas the first has a limit.

Rate nodes specify the amount of stock that is to be transferred 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 calculate 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. In order for the rate node to get the information from stock nodes, constant nodes of clock nodes, it is necessary to connect an Info Edge from the information source to the rate node. For each iteration, the rate of the stock flow for the next iteration is calculated as a function of all the nodes connected through info edges to this rate node.

In order to transfer stock from one stock node to another, the source stock node has to be connected to a rate node, using a Flow Edge that is available on the same palette. This rate node specifies the amount of stock to be transferred in the next iteration and is connected to the second stock node that will receive the stock to be transferred, 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. By setting the Step Control slider it is possible to control the
7.4. System Dynamics Modelling

Figure 7.4: System Dynamics Palette.

amount of time units that are processed each time the button on its left is pressed. 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 of the iterations. This Graph Display node is defined in a separate reference frame, called Function Reference Frame.
7.4.2 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 have they 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 crossed dependencies between one species (rabbits), its food source and a species of predators (wolfs) do exist. The objective is to find the necessary relations and initial values as to find a cyclical behaviour of the system.

7.4.3 Related Queries

In this scenario, the queries used 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

7.4.4 Discussion and Results

In this case, the students had more freedom to model the situation, without any 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 identifies potentially interesting student documents.
7.4. System Dynamics Modelling

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 of the cases where the difference was above a reasonable amount, about twice as many nodes and edges as expected or more, a closer 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 was not satisfactory, but by creating the new model on another page, or delete 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 actions.

In order to not force each teacher to make their own interpretations of possible causes for the results of a query, part of the courseware should include the reasoning and possible interpretations. A teacher should expect the courseware to include the queries that are specific to this classroom situation or activity, and some suggestions of how to interpret and react 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, 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. Using these hints, a teacher may interpret the results accordingly verify or alternatively disambiguate the interpretation and decide what action to take.
Chapter 7. Query Usage Scenarios
Chapter 8

Discussion and Conclusions

The hypotheses stated in 1.2 are following:

**Hypothesis 1:** It is possible to identify situations in a classroom, based on the information available within the system.

**Hypothesis 2:** It is possible to reduce the effort to apply Best Practices in a classroom by using a tool that identifies the possible situations within the classroom.

By looking at the experiences in the scenarios described in 7, 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. Based on the identification of situations, it is then
feasible to not only define in the Best Practices that include recommendations to be applied in the identified situations.

When a recommendation can be classified as scenario-specific (see 3.3.2), the possible measures (4) expected as a result of applying the recommendations is also scenario-specific. However, general or domain-specific recommendations do not take into account the different characteristics of scenarios. These characteristics can be associated to the people participating in the activities, including age, previous knowledge, background and motivation. Or they can be associated with the subject matter or domain that is treated in the scenario. In these cases, it might be necessary to specify or adapt the recommendations, specially the values of the measures and how the recommendations apply in each case.

In order to validate a best practice, the teacher needs to compare the outcome of the activities with the outcomes expected by the proposed best practices. But not only is it necessary to analyze 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
8.1. Methodology to Define, Validate and Apply Best Practices

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 surface 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.

8.1 Methodology to Define, Validate and Apply Best Practices

The definition of Best Practices is relatively straight forward 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 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 also 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 their 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 learn how to best learn a given subject in several different situations, as to improve their understanding of the process.

8.2 Future Work

8.2.1 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.
8.2. Future Work

8.2.2 Applying the Queries in other Learning Environments

Certain parts of the Query Reference Implementation are dependent on the Computer-integrated Classroom V2 (CiCv2, see 2.7), 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 (6.3.1) and a way to connect queries using the QueryConnection, QueryConnectionLink and thus SubQuery interfaces (6.3.2).

8.2.3 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.

8.2.4 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.
8.2.5 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.

8.2.6 Interactive Queries

During the usage of the queries, it was possible to face some limitations that can be overcome by including “interactive 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 comfort of the teacher, 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

Examples

A.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 6.6, 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
Chapter A. Examples

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 student. 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.
A.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 B

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 6.1.

B.1 “Classroom Snapshot” Implementation

The Classroom Snapshot query consists of two visible queries (as seen in figure B.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.
Chapter B. Implementation of sample Queries

Figure B.1: “Classroom Snapshot” Implementation
B.2. “List Students” Implementation

The first query, a StudentQuery and thus a ComplexQuery, consists of two subqueries (see figure B.1 (b)). The first of them is a CurrentDocumentQuery, and its output is sent to a SaveQuery 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.

B.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 B.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.

B.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>

B.2.2 XQuery 2 of part (b)

<EllipseNode FontSize="10" creationtime="1110487750862" creator="jhp"
lastModificator="none" package="info.collide.plugins.mindmap"
Chapter B. Implementation of sample Queries

Figure B.2: “List Students” Implementation
B.3. “Current Page from Student” Implementation

Figure B.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).

B.3.1 XQuery of part (b)

```xml
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
```

B.3.2 XQuery of part (c)

```xml
let $input := input()
let $sessiondata := $input//SessionData
let $student :=
```

queries.tex Rev. 1.3 (02/08/2005 18:30) - jhp
Figure B.3: “Current Page from Student” Implementation
B.4 “Specified Page from Student” Implementation

$\text{input}[/\text{CiCQuery[@label="Student"]}/\text{CiCQueryResult/child::text()}$

let $alt := \text{sessiondata}@alt\_classname$

for $\text{workspace in sessiondata//Workspace}$

let $\text{current := workspace/@currentPage}$

where $\text{current = "yes"}$

return $<\text{SessionData}>\{alt\}$

$<\text{Workspaces}>\{workspace\}</\text{Workspaces}>$

$</\text{SessionData}>$

B.4 “Specified Page from Student” Implementation

Figure B.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).

B.4.1 XQuery of part (b)

for $\text{student in input()//Student}$

let $\text{username := fn:string(student/@username)}$

let $\text{wantedun := student//CiCQuery[@label = "Student"]}/\text{CiCQueryResult/child::text()}$

where $\text{username = wantedun}$

return $\text{student//SessionData}$
Chapter B. Implementation of sample Queries

Figure B.4: “Specified Page from Student” Implementation
B.5. “Diff Count” Implementation

B.4.2 XQuery of part (c)

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

B.5 “Diff Count” Implementation

The “Diff Count” query, as presented in figure B.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).

B.5.1 XQuery of part (c)

```xml
<AnnotationNode FontSize="10" creationtime="1110487750862" creator="jhp"
lastModifier="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>
```
B.5. “Diff Count” Implementation

Figure B.5: “Diff Count” Implementation
B.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>

B.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>
B.6 "Opened File" Implementation

The "Opened File" query shows a list of students that have opened a specified file. In figure B.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).

B.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>

B.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"
    sticky="false" uiLocked="false"
Chapter B. Implementation of sample Queries

Figure B.6: “Opened File” Implementation
B.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>

B.7 “Last Activity” Implementation

Figure B.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 B.8 we see the detail of the subqueries in the query presented in figure B.7 (part (a)) and the detail of the query “Activity in last X Minutes” (part (b)).

B.7.1 XQuery of part 1
Figure B.7: “Last Activity” Implementation, part 1

```xml
<th>Strokes</th><th>Actions</th>
</tr>
<tr>
  <td>javi</td>
  <td>3</td>
  <td>0</td>
  <td>1</td>
</tr>
<tr>
  <td>pedro</td>
  <td>1</td>
  <td>8</td>
  <td>4</td>
</tr>
<tr>
  <td>javier</td>
  <td>0</td>
  <td>0</td>
  <td>7</td>
</tr>
<tr>
  <td>miguel</td>
  <td>2</td>
  <td>1</td>
  <td>0</td>
</tr>
```

<Figure>

```xml
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
}
</Content>
</AnnotationNode>

Chapter B. Implementation of sample Queries
B.7. “Last Activity” Implementation

Figure B.8: “Last Activity” Implementation, part 2
Chapter B. Implementation of sample Queries

B.7.2 XQuery 1 of part 2

\begin{verbatim}
<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")
\end{verbatim}
B.7. “Last Activity” Implementation

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>

B.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>
B.7.4 XQuery 2, part 2

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

B.8 “List MatchMaker Collaborations” Implementation

Figure B.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).

B.8.1 XQuery 1

```xml
<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>
```
B.8. “List MatchMaker Collaborations” Implementation

Figure B.9: “List MatchMaker Collaborations” Implementation
**Chapter B. Implementation of sample Queries**

### B.8.2 XQuery 2

```xml
<EllipseNode FontSize="10" creationtime="1110487750862" creator="jhp"
  lastModificator="none" package="info.collide.plugins.mindmap"
  sticky="false" uiLocked="false">
  <Content>{input()//text()}</Content>
</EllipseNode>
```

### B.9 "MatchMaker Activity Indicator" Implementation

Figure B.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.

#### B.9.1 XQuery 1

```xml
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>
```
B.9. “MatchMaker Activity Indicator” Implementation

Figure B.10: “MatchMaker Activity Indicator” Implementation
Chapter B. Implementation of sample Queries

B.9.2 Sample output of XQuery 1

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

B.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>

B.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>

Chapter B. Implementation of sample Queries

<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="objectChanged"
  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"/>
B.9. “MatchMaker Activity Indicator” Implementation

<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"
Chapter B. Implementation of sample Queries

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>

B.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>

B.10 “Model Complexity” Implementation

Figure B.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).

B.10.1 XQuery 1 of part (a)
Figure B.11: “Model Complexity” Implementation
B.10. “Model Complexity” Implementation

<Content>
{let $students := input()//Student/result
let $counts :=
for $student in $students
return count($student/page)
let $npages := max($counts)
let $sth :=
    for $page in (1 to $npages)
        return concat("<td>Title P", $page, ", Nodes P", $page,
            "<td>Edges P", $page, "<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>", $nodes, "<td>", $edges, "<td>", $strokes, "</td>"
)
    return $ph
let $tot :=
    concat("<tr><td>", string($student/@username), "</td>", $phtml, "</tr>"
)return $tot
}
"</table></HTML>"} </Content>
</AnnotationNode>
Chapter B. Implementation of sample Queries

B.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


BIBLIOGRAPHY


Baloian, N., Hoppe, H. U., and Luther, W. (2002b). Teaching strategies to automatically configure course material in undergraduate lectures. 20


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY

Stallman, R. M. (2002). Free software, free society: selected essays of richard m. stallman. GNU Press, Boston, Massachusetts. 52


BIBLIOGRAPHY


