An XML-based Query Mechanism to Augment Awareness in Computer-integrated Classrooms

Jens HARDINGS
DCC - Universidad de Chile
jharding@dcc.uchile.cl

Abstract. "Awareness" in a distant learning scenario is understood as the degree of achieving the same experience as in a face to face situation. I present a mechanism to obtain information in a Computer-integrated Classroom (CiC), based on access to several XML data sources, that will improve awareness within a face to face classroom situation. This mechanism should enable participants of the CiC, particularly the teacher, to increase their ability to manage the sessions efficiently, since they have help to access detailed information at the right moment.

1 Introduction

Dourish and Belotti [1] define awareness as an *understanding of the activities of others, which provides a context for your own activity*. In groupware, the concept of achieving "awareness" is generally associated with trying to provide the same experience as is possible in a face to face scenario, adding tools that mimic many of the information channels that get lost because of the lack of presence. However, little work has been done in trying to increase the context information in face to face scenarios to increase the natural awareness found in these situations.

I try to extend the notion of awareness in a way that a person can improve her ability to manage face-to-face situations, in particular a classroom scenario. Each person has a limited capability of attention, and by using tools that help us to handle a greater amount of information in a more efficient way, we can be better connected to the reality than we would on our own. Teachers could be aware of more details to determine if students are unchallenged or overstrained.

To achieve such increased awareness, it is necessary to have tools that allow the teacher or other participant to have access to information that is not directly available. The mechanism that will enable increased awareness is a querying system, including a library of queries and a distributed system to perform these queries, gathering the result and displaying or storing it. The advantage of a querying system is that it is possible to access all information at any time, without having to be consciously of that information except when needed. A teacher may want to know if students have been participating, taking notes or have done the assignments, even if she does not annotate those events on a regular basis. Other data may not be available under normal circumstances, like the order in which actions have taken place.

2 Architecture

The system I implement builds on the concept of a computer-integrated classroom as described in [2], adding a querying system that allows to get access to available data in a usable form. I will assume that the teacher is the one that performs queries, although it will be possible to extend the system in order to allow other participants to perform queries, probably with access restrictions that would impede unwanted access to some information.

Within the CiC, each participant can have her own instance of a Classroom Module running, to access shared documents or perform local annotations, construct models or other work. In most cases, the annotations and modelling will take place using a tool called FreeStyler [3], which stores its documents in XML format and has been integrated into the CiC as the default tool to handle documents.

In order to achieve useful results within the context of the CiC, i.e. the face-to-face classroom session, it is necessary to have access to the information as it is being produced on each separate Classroom Module. The data that is handled within each Classroom Module has be made available to one location where it is gathered, possibly taking advantage of the distributed nature to make parallel processing as it is 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 as we will see in 2.2.

2.1 Abstraction level for queries

We see that it is possible to process the queries either centered on an approach that uses the application's internal model directly or to use the XML files that are the persistent form of the same model. Evidently it is useful to have only one implementation instead of developing two independent applications that are identical in their final outcome. And since the performance costs of converting from one format to another would be affordable in this case, I have decided to implement the system accessing the XML data instead of loading it and manipulating the internal application data.

One advantage of using XML over the internal data is that it is possible to apply the same or similar procedures to other XML-based data. In particular, it is necessary to access other data than the documents by the presented scheme, so it would have been necessary to manipulate that XML data anyway. Additionally, within a distributed scheme (see 2.2) 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, whereas using an internal model of the data would mean to take into account an additional serialization / deserialization process inbetween. The availability to manage external data is also of great importance, since new queries can be added that manage ad-hoc XML data without further using that data in the applications.

The drawback is that all documents that are being edited have to be saved first in order to perform the queries on current data. Since it is not necessary to receive the results of a query instantly, that cost is affordable considering the benefits.

2.2 Information sources

As already implicitly stated, the information sources on which the queries will draw upon are various and of different structure. Besides the information that is stored directly (through the user's modeling and note taking) and indirectly (as metadata that the application adds) within the documents, the CiC system keeps track of much more information, like access to files or interactions among participants.

The system can access information from different sources, performing specific queries on distributed data and aggregating the parts to form a unique result in the form of an XML document or fragment which may then be incorporated into an existing document.

- FreeStyler files: written in XML, the FreeStyler documents represent the persistent serialized version of the application's data. The explicit information of all users is to be found in these documents.
- Metadata: the system will keep metadata information that is not explicitly created by the users. In the case of FreeStyler, the metainformation could be managed internally, but to allow the use of external tools whose file format is unknown it is preferable to maintain the metadata in a separate file.
- CiC Logfiles: the Computer integrated Classroom 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.
- MatchMaker logs: during the classroom sessions, it is possible for the applications to interact using a server called MatchMaker [4] which enables the underlying model of FreeStyler documents to be shared in real-time among several participants, in a way to see current modifications as they occur and provide a copy of the model to each of the involved parties. This application generates its own log of events in several XML files which can be used to obtain valuable information in this system.

2.3 Distribution of queries and sources

In order to coordinate the different information sources, it is necessary to have a directory or index in which all possible sources are listed and which describes how to access those sources. Additionally, since the nature of the system is to have distributed information sources, it makes great sense to perform at least a part of the queries in a distributed form, even more if the results are expected to be much smaller in size than the complete information source. Each module processes locally the most part of a query that is possible, passing the result of the query to a central querying engine that gathers all needed results and raw data to continue with the query processing as needed. The results are then available at the central querying engine, which may include them in a document or otherwise display the results.

2.4 Visualization of results

The immediate result of a query will be either an XML fragment, which could be saved as a new document or parsed to be integrated into the FreeStyler application to visualize it. But in other cases, the result should be best saved into a new document, or maybe even fragmented

into several documents for later use. The preferred way to save the results is in the FreeStyler document structure, since that way it would integrate best into the CiC system. This would be the case if the result is a type of report, aggregation of information available in documents or other data that may be processed further.

However, in some situations it may not be desirable to display the result at all, but the query may be started and the result analyzed by a software module that monitors the development of the classroom sessions and triggers actions according to the results.

2.5 Query library

One central part of the system is a collection of predefined queries that will be available as a library, similar as the concept of a view in Databases. These queries can be parameterized and include both atomic and complex queries (see 3), that can be used directly or within new complex queries. It is possible to have document-specific queries, that for example allow to automate some processing of multiple choice test, be it for assessing or self helping purposes. With most of the queries it makes sense to combine them, raising the possibilities without necessarily having exponential numbers of different queries to choose from. The flexibility of the library is therefor more important than its completeness, since an unstructured numerous amount of queries does not help improve the classroom experience.

3 Query composition

Using the Composite Design Pattern as described in [5], 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 which is mentioned in 2.3.

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. It can be the case that 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 language like XSLT [6] or XQuery [7].

An atomic query can be as complex as we want it to, as long as they do not involve other queries. But if it is possible, it should be preferred to separate a query into several, since that way we allow further reuse of parts that may have other future uses.

It is possible to distinguish different possible query types, based on the type of processing that is performed. Interesting possibilities include following aspects:

- comparison of XML documents: using a comparing mechanism like the described in [8, 9] and meeting certain conditions, it is possible to calculate the editing distance [10] between two documents efficiently. The documents are considered as trees by using the DOM (Document Object Model).

- filtering queries: several of the typical aggregate operations that are available on any database system like counting, filtering, grouping, sorting, etc. All of these queries should be implemented using either XSLT or XQuery. However, most of these queries are suitable to be used as complex queries, in the sense that the input may as well be the result of a previous query.

- content specific queries: taking advantage out of knowing the document structure and semantics, it is possible to identify certain properties. For example, the FreeStyler tool allows the use of freehand writing and drawing, so if a query should be able to distinguish the intent of some lines that are represented in the document, it would fall into this category of document specific query, since it is not possible to accomplish without a precise knowledge of the semantic within the document.

3.2 Complex queries

There are several ways of combining atomic queries, and each complex query that has been defined can from that point on be used in any other complex query.

One important point to consider is that different atomic queries will deliver rather incompatible results, the fact of both being XML notwithstanding. For example, one query might return the number of ocurrences of a certain element within a document, whereas another query will put out a listing of differences among two documents. It is clear that both results are not interchangeble, moreover if we are to use the details within the results. Therefore, it may be necessary to restrict the composition of queries so that the output of one fits with the input of the next. This can be done by using typing the streams, in which case errors would be checked at the construction time of the query, or verifying them at run-time.

A query will have the structure of a tree in which the leafs are atomic queries. All complex queries have one or more "children" queries from which they receive the input in form of an XML stream. The construction of such a tree will initially be done in a low-level language, but it should be possible to create a graphical interface that manages the composition in order to create new queries on demand.

4 Scenario description

It is possible to define rather simple queries, such as to determine the assistance or checking for homework delivery, or complex ones as to calculate which of the sections of the course notes have most student annotations, which would possibly mean that it is either an interesting point or it needs more work to clarify. Knowing if all students are reading a specific page in the notes or in which order they have worked on the assignments proposed can be helpful for a teacher to determine the improvements and problems of her students. This information can be available to the teacher either on demand or by repeating the queries automatically at a given time interval.

The most interesting uses seem to be within the classroom session, since the information available cannot otherwise be analyzed quickly nor throughly enough to achieve a useful result. But it is also valuable to be able to analyze past sessions and even past courses, to determine the best way to improve notes or classroom planification.

5 Conclusions and future plans

The queries can allow a teacher to improve her ability to manage the information available in a classroom. It is possible to analize part of the information with greater detail, which means that it is possible to manage more complex models without loosing overview of the big picture.

Information about the queries, in the form of metadata, may be made available to the system, in order to be able to determine the importance of the query in a particular context. This way, it will be possible to aim at the proposed goal of the CiC to provide most needed actions in a comfortable way, and do it dynamically as the context changes.

Depending on the scenario, the last events or last actions of a particular user, it will be possible to determine the level of competence for each of the queries, creating an ordered list that can serve as a shortcut for the most obvious next actions within the current context. Additionally, the same data can be used to suggest potentially interesting compositions in order to further elaborate the queries that just have been executed.

In order to determine what information is of importance in a classroom situation, the work will be done in close relation with teachers actually using the tools. This interaction will provide the necessary experience to get the really needed information and allow the corresponding evaluation of the system in an empirical setting.

References

[1] Paul Dourish and Victoria Bellotti. Awareness and Coordination in Shared Workspaces. In *Proceedings of the ACM conference on computer supported cooperative work (CSCW'92)*, pages 107–114, Toronto, Ontario, 1992. ACM Press.

[2] Nelson Baloian, Alexander Berges, Stephan Buschmann, Katrin Gaßner, Jens Hardings, H. Ulrich Hoppe, and Wolfram Luther. Document management in a computer integrated classroom. In Joerg M. Haake and Jose A. Pino, editors, *Proceedings of CRIWG 2002, 8th International Workshop on Groupware*, LNCS, pages 35–46. Springer, September 2002.

[3] H. U. Hoppe and K. Gaßner. Integrating collaborative concept mapping tools with group memory and retrieval functions. In Gerry Stahl, editor, *Proceedings of CSCL 2002*, pages 716–725. Lawrence Erlbaum Associates, January 2002.

[4] Marc Jansen, Niels Pinkwart, and Frank Tewissen. MatchMaker - Flexible Synchronisation von Java-Anwendungen. Technical Report Forschungsbericht 763, Universität Dortmund Fachbereich Informatik, October 2001.

[5] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[6] W3C. XSL Transformations (XSLT). http://www.w3.org/TR/xslt/, November 1999. Version 1.0.

[7] W3C. XQuery 1.0: An Xml Query Language. http://www.w3.org/TR/xquery/, April 2001. Working Draft.

[8] Yuan Wang, David J. DeWitt, and Jin-Yi Cai. X-diff: a fast change detection algorithm for xml documents. Submitted for publication, 2001.

[9] Grégory Cobéna, Serge Abiteboul, and Amélie Marian. Detecting Changes in XML Documents. In *Proceedings of International Conference on Data Engineering (ICDE) 2002*, San Jose, California, USA, February 2002.

[10] K. C. Tai. The tree-to-tree correction problem. *Journal of the ACM*, 26(3):422–433, 1979.