Journal of Web Engineering, Vol. 6, No.3 (2007) 196-221
© Rinton Press

INTEGRATING INTERACTION DESIGN AND LOG ANALYSIS: BRIDGING THE GAP WITH UML, XML AND XMI

GHEORGHE MURESAN

School of Communication, Information and Library Studies
Rutgers University, New Brunswick, USA
muresan@scils.rutgers.edu[a]

Received January 15, 2007
Revised March 23, 2007

In this paper, we describe and discuss a formal methodology that integrates the conceptual design of the user interaction for interactive systems with the analysis of the interaction logs. It is based on (i) formalizing, via UML state diagrams, the functionality that is supported by a system and the valid interactions that can take place; (ii) deriving XML schemas for capturing the interactions in activity logs; (iii) deriving log parsers that reveal the system states and the state transitions that took place during the interaction; and (iv) analyzing the state activities and the state transitions in order to describe the user interaction or to test some research hypotheses. While this approach is rather general and can be applied in studying a variety of interactive systems, it has been devised and applied in research work on exploratory information retrieval, where the focus is on studying the interaction and on finding interaction patterns. The details of the methodology are discussed and exemplified for a mediated retrieval experiment.

***Key words*:** Methodology, interaction design, log analysis, UML, XML, DTD, XMI, search process, mediated information retrieval

1 Introduction

1.1 Motivation

While much of the research work in **Information Retrieval** (IR) has focused on the systemic approach of developing and evaluating models and algorithms for identifying documents relevant to a well-defined information need, there is increasing consensus that such work should be placed in an **Information Seeking** framework, in which a searcher's context, task, and personal characteristics and preferences should be taken into account (Ingwersen and Jarvelin, 2005).

Since Robertson and Hancock-Beaulieu (1992) described the cognitive, relevance and interactive "revolutions" expected to take place in IR evaluation, the focus in interactive IR experimentation has shifted to exploring the dynamic information need that evolves during the search process, the situational context that influences the relevance judgments and the strategies and tactics adopted by information seekers in satisfying their information need. This paradigm shift to a cognitive approach to exploring search interactions and to studying **Human Information Behavior** has generated a large

[a] The author's new affiliation is: Live Search, Microsoft Corp, Redmond, WA, gmuresan@acm.org.

number of theories that attempt to model the search interaction and to predict the user's behavior in different contexts and at different stages of the interaction (Fisher et al, 2005).

Of particular interest to this author are models of the search interaction process and empirical work to validate such models by observing consistent patterns of user behavior (Ellis, 1989; Kuhlthau, 1991; Belkin et al, 1995; Saracevic, 1996; Xie, 2000; Vakkari, 1999, 2001; Olah, 2005). The interest is not simply in validating theoretical models, but also in designing systems that better respond to user needs, that can adapt to support various search strategies, and that offer different functionality in different stages of the information seeking process.

We are interested in methodologies for running interactive IR experiments, and especially in the practical aspect of client-side logging of the interactions and analyzing the logs in such a way that as many meaningful details as possible are captured. The analysis of the logs can subsequently be used to observe patterns of behavior, to build a model of the interaction and possibly to predict user behavior in certain contexts, or simply to test the usability of a user interface. Although no systematic study has investigated the methodologies used for this kind of experiments, there is plenty of anectodal evidence to suggest that much of the investigation is manual and ad-hoc: the researchers examine interaction transcripts or videos, and assign codes to significant actions that take place and to shifts in interaction stages. This process is slow, expensive, and error prone. Logs of interactions are sometimes employed to address this issue: events and actions are logged during the search interaction, and the logs can be analyzed afterwards. However, in our experience, there is usually little or no formal process in designing the logs, the logging process, and the log analysis, in order for the states of the system and the stages of the interaction to be captured.

A number of software tools have appeared on the market to support interaction analysis: their typical functionality is to capture the screen, to film the subject during the interaction, and to log keyboard and mouse events[b,c,d]. The researchers can subsequently examine the interaction, interpret what is happening, insert annotations or mark significant events. Unfortunately, these tools are generic, rather than targeted at a certain type of interaction. Therefore, while they can be helpful, the bulk of the work is still the manual-intellectual annotation done by the researcher. Moreover, the format used for the logs is usually proprietary, which forces the researchers to buy proprietary analysis software that is not customizable.

A second motivation for the proposed methodology comes from observations of a number of interactive IR experiments where the systems had clear usability issues. Such situations are common and not at all surprising: these are experimental systems (as opposed to commercial systems), built for studying some aspects of the interaction, so little or no resources are available for high-quality design and usability testing. Unfortunately, this can potentially lead to compromised research results, as the usability of the interface can potentially affect the searchers' behavior.

Figure 1 captures the typical experimental procedure employed in interactive IR: the baseline and experimental systems are specified, designed and built based on the research questions or hypotheses investigated, code for logging events and actions is inserted in the appropriate places, and the logs are analyzed after the experiment is completed in order to address the research questions. Most often the logging code is added informally, as an afterthought. Therefore, when analyzing the logs, it is difficult to relate the captured events to the states of the system or stages of the interaction.

[b] Morae: http://www.techsmith.com/morae.asp
[c] TaskTracer: http://eecs.oregonstate.edu/TaskTracer/
[d] uLog: http://www.noldus.com/site/doc200603005

Our rather unfortunate experience in Interactive TREC 2002 (Belkin et al, 2002; Hersh, 2002, for the track description) is a case in point. First, our user interfaces were coded without appropriate state diagram design, which created a number of usability issues: “Save”, “Bookmark” or “View” buttons active even when no documents were selected, or even before a search was conducted, “Search” button active even when no query was specified, or while a search was already being conducted (which allowed queries to be submitted multiple times), “Back” button when no document was yet in the history stack, etc. This can potentially bring into question the validity of our research results and conclusions, as the usability of the interface could potentially affect the searchers’ behavior.

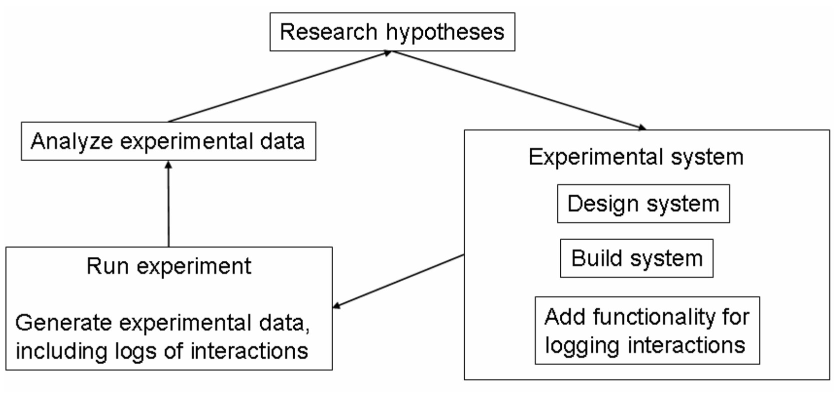

Research hypotheses
Analyze experimental data
Experimental system
Design system
Build system
Add functionality for logging interactions
Run experiment
Generate experimental data, including logs of interactions

Figure 1 A typical experimental procedure in interactive IR

Second, we only captured in the logs the data that we anticipated would answer our research questions (e.g. number of queries submitted, number of documents viewed or saved, etc). Moreover, the logging code tended to summarize the data rather than log all the details of the interaction; for example, it logged the number of viewed documents rather than the documents themselves. Figure 2 depicts an example of such a log.

The consequences were that: (i) we were unable to refine our hypotheses and to do more detailed analysis of the interaction, in light of the initial results, because no extra data was captured in the logs (e.g. no record of viewed documents was kept); (ii) in the occasions when the system crashed, the data accumulated in memory, necessary for building summaries (e.g. the number of viewed documents), was lost, so the logs were useless.

```
TREC-2002 START: 2002-08-15 17:58:57
QUERY: geneticly engineered foods safety
QUERY: geneticly engineered foods safety
SAVE DOCUMENT: [G13-84-2041245] Food Safety and Biotechnology: Are They Related?
QUERY: problems genetically engineered foods
SAVE DOCUMENT: [G40-01-0459199] International Information Programs, U.S. Department of
State, Economic Perspectives, October 1999
FINALLY SAVED DOCUMENTS:   [G40-01-0459199] International Information Programs, U.S.
Department of State, Economic Perspectives, October 1999;        [G13-84-2041245] Food Safety and
Biotechnology: Are They Related?
NUMBER OF VIEWED DOCUMENTS: 12
NUMBER OF UNIQUE VIEWED DOCUMENTS: 8
TREC-2002 STOP: 2002-08-15 18:03:50
```

Figure 2 A sample extracted from Interactive TREC 2002 logs

1.2 Vision

What we propose in this paper is a formal procedure that integrates the modeling of the interaction, the logging process and the log analysis, so that (i) the user interface accurately implements the conceptual model of the interaction intended to be supported; (ii) the stages of the interaction and the states of the system are captured accurately in the logs; and (iii) the logs can be analyzed in a systematic and at the same time flexible way. When applied to a particular kind of interaction (such as interactive information retrieval), the proposed procedure can be used to investigate user behavior or to test the usability of a user interface. The optimal situation is when the researchers design the user interface and build the experimental system, so that the design of the logging can be easily integrated, and the semantic events, defined in the design stage, can be easily detected and logged. If user interaction with a third party system is studied (e.g. accessing a commercial search engine via a web browser), then the procedure can still be applied, but more work is needed to recognize significant, semantic events and actions among the keyboard and mouse events that take place during the interaction.

We propose integrating the design of the interactive system, the design of the logger, and the design of the log analyzer, by requiring the development of a **conceptual model** of the interaction, which unifies these designs, as depicted in Figure 3. While this means more work at the onset, and may seem un-necessary when the experimental schedule is tight, it pays off in the long run. Moreover, the entire research team can participate in the conceptual design, with the advantages that some mistakes and omissions may be avoided, the team members have a better understanding of the underlying interaction model, and the work can be more easily shared. This contrasts with the common situation when the designated programmers build the system and other members of the research team do the log analysis, with insufficient collaboration.

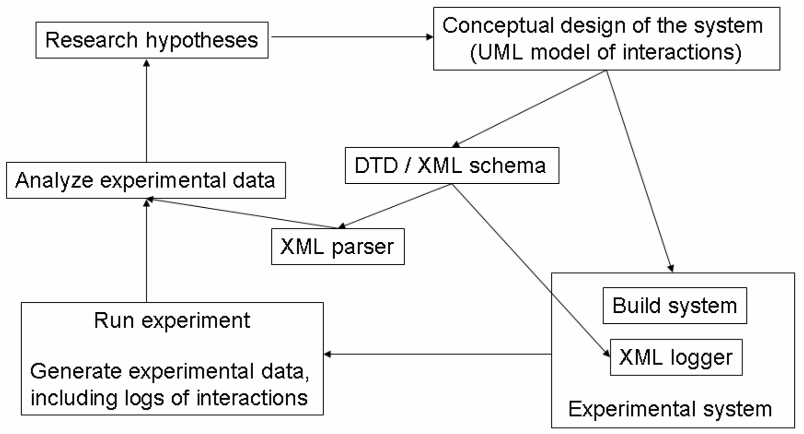

Research hypotheses
Conceptual design of the system (UML model of interactions)
DTD / XML schema
Analyze experimental data
XML parser
Build system
XML logger
Experimental system
Run experiment
Generate experimental data, including logs of interactions

Figure 3 Integrated approach to design, logging and analysis

Our formal approach is based on statecharts (Harel, 1988) or, in the more modern UML (Unified Modeling Language)[e] terminology, on state diagrams. These are extensions of finite state diagrams[f], in which the use of memory and of conditional transitions makes it practical to describe system behavior in reasonably compact diagrams. Such a model of a system describes; (i) a finite number of existence

[e] http://www.uml.org/
[f] http://en.wikipedia.org/wiki/Finite_state_machine

conditions, called **states**; (ii) the **events** accepted by the system in each state; (iii) the **transitions** from one state to another, triggered by an event; (iv) the **actions** associated with an event and/or state transition (Douglass, 1999; Fowler, 2004). Such diagrams have the advantage that they describe in detail the behavior of the system and, being relatively easy to learn and use, allow the participation of the entire research team in developing the conceptual model of the IR system to be employed in an experiment. It also makes it easier for the designated programmers to implement and test the system, as the logic is captured in the model.

While UML is well suited to design the interaction supported by a user interfaces, XML is an excellent choice of format for logging user actions and state transitions. The Extensible Markup Language (XML[g]) is a World Wide Web Consortium (W3C[h]) endorsed standard for document markup that offers the possibility of cross-platform, long-term data storage and interchange. XML is more than a mark-up language: it is a meta-markup language, in the sense that it can define the tags and elements that are valid for a document or set of documents. For our purposes, it has the advantage that it is non-proprietary and it can be examined with any text editor or open-source XML editor. Also, there are plenty of XML parsers available, written in various programming languages, so processing the logs and extracting relevant information is easy. Moreover, it allows a variety of access modes: (i) sequential access to each event in the log (via SAX[i]); (ii) random access to certain kind of events, relevant for a certain research hypothesis (via XPATH[j]); and (iii) complex visiting patterns (via DOM[k]).

Closely related to XML are two other standards, Document Type Definitions (DTD) and the W3C XML Schema Language, which are used to describe the vocabulary and language of an XML document. A DTD or an XML Schema, (or simply "schema", to refer to either) can be used by a human to understand or to impose the format of an XML document, or by a machine to validate the correctness of an XML document. Moreover, it can be used by an increasingly number of tools (such as the open-source NetBeans) to generate parsers for such XML documents.

While in principle both DTD and XML Schema can be used, there are some differences between the two. DTD's have the advantage that are easier to write and to interpret by a human and, as they have been around for longer, there are more tools to process them for XML validation and code generation (most commonly into Java or C++). The newer XML schemas allow more specificity in defining types of elements and attributes, but that comes at the cost of reduced readability and more human effort. It is envisaged that the two will co-exist in the future, and that a pragmatic choice can always be made according to the context as to which is more appropriate to use.

UML is ideally suited to support the design of systems, and XML for recording the activity logs. The problem is bridging the gap between the two. One approach fully supported by existing technology is to use the Java Architecture for Data Binding (JAXB[l]) specification to derive Java classes (or rather skeletons of Java classes, specifying name, attributes and method prototypes) from UML diagrams, and then XML DTDs or XML schemas from the Java classes. This approach has the advantage that the skeletons of the Java classes can be expanded with code either for implementing the user interface, or for processing the logs.

[g] http://www.w3.org/XML/
[h] http://www.w3.org
[i] http://www.saxproject.org/
[j] http://www.w3.org/TR/xpath
[k] http://www.w3.org/DOM/
[l] http://java.sun.com/webservices/jaxb/

An alternative solution is to use the Object Management Group's (OMG) XML Metadata Interchange (XMI) specification[m]. Initially created as an open source specification that allowed modeling tools from different vendors (such as Rational Rose, TogetherJ) to export/import design models, XMI has grown to wider applicability by supporting the production of XML vocabularies and languages that enable the integration of many e-business applications (Carlson, 2001, 2006). XMI specifies a set of mapping rules between UML and XML in terms of elements, attributes and relationships. It must be noted that mapping UML to XMI is not an exact science, and different levels of strictness can be applied, and tradeoffs between a number of mapping decisions can be specified. For example, attributes specified in a UML class diagram can be converted to either XML elements or XML attributes. Carlson (2001) discusses at length such tradeoffs, as well as the use of XPath, XPointer[n] and XLink[o] in implementing more complex relationships from UML diagrams, such as inheritance, association or composition.

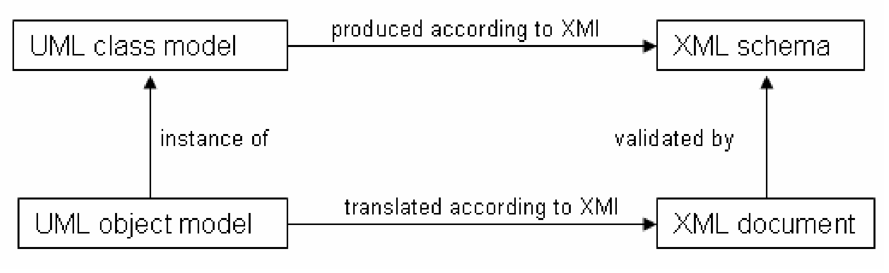


Figure 4 Mapping UML models to XML schemas and documents

Figure 4 captures this approach. UML class diagrams provide the blueprints for UML object diagrams, and XML schemas provide the template for XML documents. XMI specifies the translation of UML class models into XML schemas and of UML object models into XML documents. The obvious and direct application of this approach to logging the interaction appears to be the following: (i) derive UML class diagrams from state diagrams (this is trivial, as the states at different levels of granularity correspond to classes); (ii) use XMI to derive XML schemas from the UML class diagrams; and (iii) capture in XML logs the successive states of the user interfaces, after each event or user action. The problem with such an approach, and the modified approach that we have adopted, are discussed in section 3, after we introduce a case study to exemplify our methodology.

In summary, the expected gains of this vision are:

- generating user interfaces that accurately implement a certain interaction model;
- client-side logs that accurately capture user interactions, such as a search session;
- support for building user models that capture usability problems as well as user preferences. This in turn can contribute to building better interfaces, and to building personalized systems that adapt to the user's needs and preferences.

[m] http://www.omg.org/technology/documents/formal/xmi.htm
[n] http://www.w3.org/TR/WD-xptr
[o] http://www.w3.org/TR/xlink/

1.3 Situating our work among related approaches

A clear distinction needs to be made between different stages of creating interactive systems when discussing and comparing approaches, methodologies or techniques, as these are different for (i) specifying the requirements of the system; (ii) designing the user interface; and (iii) designing and implementing the software. The actual stage of designing the user interface (Tidwell, 2006), although essential for building usable and ultimately successful interfaces, is not one of the concerns of our work. We are interested in linking the system specification to the software design; therefore, we are only going to discuss work relevant to this activity.

Most often, the specification of an interactive system is in the designer's natural language, such as English, accompanied by a set of the sketches of the interface at different stages of the interaction. Unfortunately, natural-language specifications tend to be lengthy, vague and ambiguous, and therefore are often difficult to prove complete, consistent and correct (Shneiderman and Plaisant, 2004). Use cases use a graphical notation to describe user goals, but the emphasis is more on the user-system interaction than in the task itself (Sharp et al, 2007). Task analysis provides a more concise and systematic way to describe and analyze the underlying rationale and purpose of what people are doing: what they are trying to achieve, why they are trying to achieve it and how they are going about it. Task analysis produces models of the world and of the work or activities to be performed in it: it describes the entities in the world, at different levels of abstraction, and the relationship between them, either conceptual or communicative (Diaper and Stanton, 2004). Actually "task analysis" is a rather generic term, an umbrella for a set of related methodologies such as Hierarchical Task Analysis (HTA), Goals, Operators, Methods and Selection rules (GOMS), Groupware Task Analysis (GTA), etc; Limbourg and Vanderdonckt (2004) provide a description of these, as well as a syntactic and semantic comparison.

The specifications above are in general at a high level of abstraction and task granularity. While useful in guiding the design of the system, they do not provide sufficient support for automatic processing in order to prove completeness or correctness of a system, or for code generation. A possible exception's is Paterno's work (2001, 2004) on graphical representation of task specification. He proposes the use of ConcurTaskTrees (CTT) and discusses a variety of ways to integrate task models, which describe the activities that should be performed in order to reach users' goals, with UML diagrams, created for supporting object-oriented software design, but biased towards the internal parts of the software system. Possible approaches are: (i) to represent CCT models with existing UML notation, e.g. with class diagrams; (ii) to develop automatic converters between UML and task models; (iii) to extend UML / building a new type of diagram. Paterno favors the latter approach, proposing a notation for tasks similar to the existing UML activity diagrams, but that also capture hierarchic relationships between tasks. These are used, is proposed, together with other UML diagrams such as use cases (which define pieces of coherent user behavior without revealing the details of the interactions with the system) and sequence diagrams, which reveal details of the interactions for a certain task or sub-task.

Paterno's work is related to ours in the sense that he also tries to bridge the gap between different levels of abstraction, moving from user tasks towards software implementation. Apart from the application of our methodology being rather different, the difference is that we are looking at a more detailed level of the interaction, which connects keystrokes and mouse events to semantic actions, in the context of solving a certain task.

Shneiderman and Plaisant (2004) also discuss more specific and formal approaches such as grammars, transition diagrams or statecharts, which provide a more fine-grained view of the human system interaction and provide support for automatic processing and a connection to software design.

For example, Winckler and Palangue propose a formal description technique based on statecharts, dedicated to modeling navigation in web application (2003). That work is indeed related to ours, but they focus and limit their attention to modeling the interaction, with no interest in logging and further analyzing it.

More closely related to our goal and approach is Trætteberg's work on DiaMODL (2003), a dialog modeling hybrid language that combines a dataflow-oriented notation with statecharts that focus on behavior. That work is complementary to ours: rather than proposing a new notation or language, our intent is to use and integrate existing notations and languages in order to combine the advantages that they offer. In that direction, we were inspired by Carlson's work on linking UML and XML (2001, 2006), which we already mentioned in the previous sub-section. However, his view is data-centric, with application in transferring data between applications, while we are mainly interested in modeling, representing, logging and analyzing user-system interactions. Similarly, Crawle and Hole propose (2003) an Interface Specification Meta-Language (ISML) which appears to be related but more generic than our Interaction Modeling Language, plus they also restrict their focus to modeling, rather than logging, the interaction.

2 Case Study: Mediated Information Retrieval

In order to help the reader more easily understand the proposed methodology, we are going to describe its application on our **MIR** (Mediated Information Retrieval) project. The focus of this paper is the experimental methodology that we designed and employed, rather than the actual research questions and the experimental results of that project. Therefore, the description of the project will be limited to the minimum necessary. A more complete description of the project and a comprehensive analysis of the results appear elsewhere (Lee, 2006).

2.1 The mediated retrieval model

We proposed the concept of **mediated information retrieval** (or access) in previous work (Muresan and Harper, 2001, 2004; Muresan, 2002), as a way to address the problem of exploratory searches, when the searcher may be unfamiliar with a problem domain, uncertain of what information may be useful for solving a particular task, or what query terms would be helpful in retrieving relevant information. The idea is to emulate the function of the librarian or intermediary searcher, who interacts with the information seeker, elicits more information and helps the searcher refine, clarify and formulate her information need. Our reification of the mediation interaction model is based on so-called **source collections**, specialized collections of abstracts or documents that cover the searcher's problem domain. These collections, which emulate the librarian's knowledge of a certain domain, are either manually structured (based on some ontology that describes that domain) or are automatically clustered in order to reveal the concepts and structure of the domain, in order to inform and educate the searcher.

The interaction model is captured in Figure 5. In the first stage the searcher interacts with the *source collection* so that (i) she becomes more familiar with the terminology, concepts and structure of the problem domain, and better able to convey her information need; and (ii) the system monitors the user's interaction and her selection of documents, and learns the type of documents that she is interested in. Following the mediation stage, the search target moves to the Web or any other *target collection* where the user hopes to find new information to satisfy her need and complete some task. At this point the system is able to support the searcher by suggesting query terms; also, the user is expected to be more familiar with the problem domain, and able to formulate better queries than before the mediation.

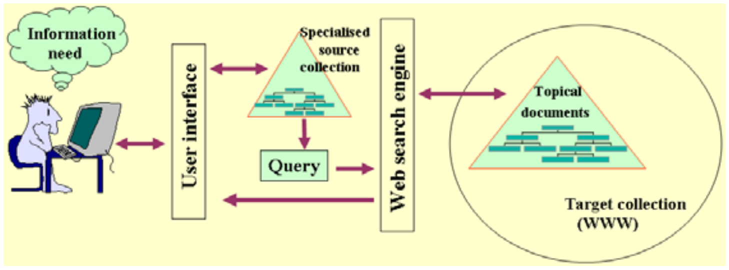
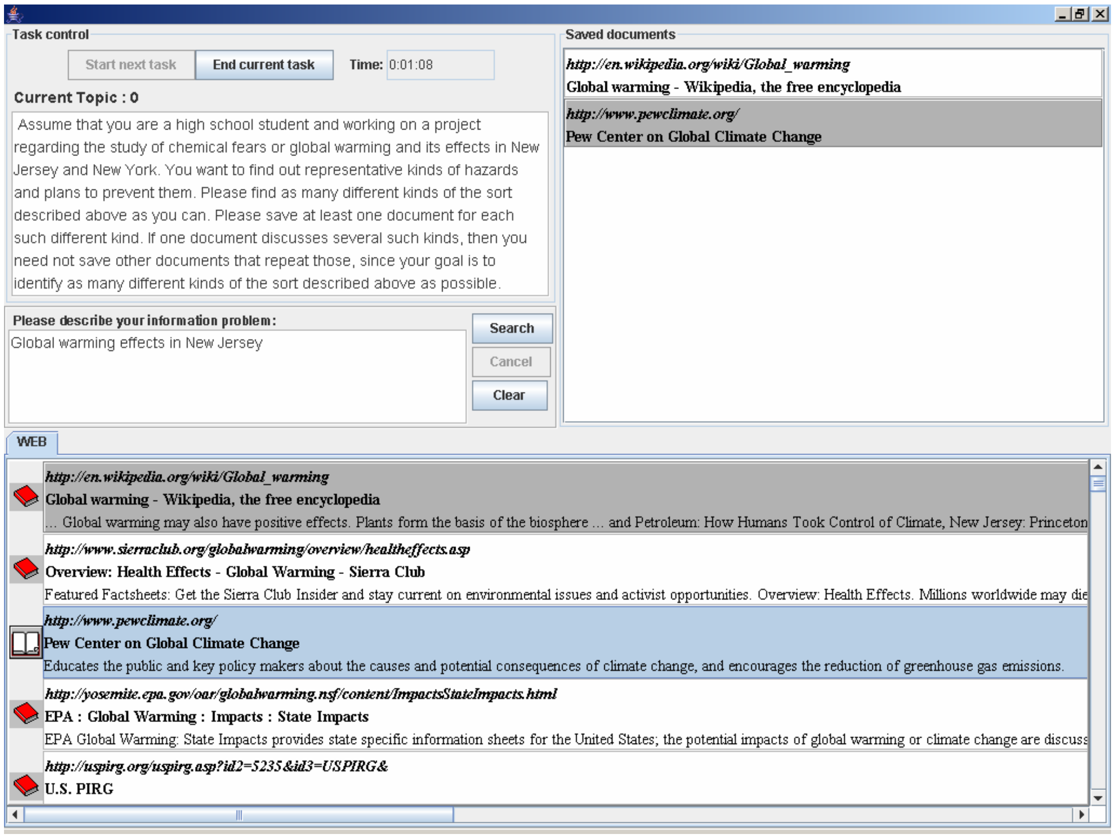


Figure 5 The interaction model in mediated information retrieval

2.2 The MIR project

In previous work we demonstrated the *potential* effectiveness of mediation through pilot studies and user simulations. In the MIR project, yet to be completed, we run formal user studies to verify if mediation can indeed improve retrieval effectiveness. Moreover, we are interested in observing patterns of interaction, which could help us design better interfaces.



Time: 0:01:08

Current Topic : 0

Assume that you are a high school student and working on a project regarding the study of chemical fears or global warming and its effects in New Jersey and New York. You want to find out representative kinds of hazards and plans to prevent them. Please find as many different kinds of the sort described above as you can. Please save at least one document for each such different kind. If one document discusses several such kinds, then you need not save other documents that repeat those, since your goal is to identify as many different kinds of the sort described above as possible.

Please describe your information problem: Global warming effects in New Jersey

Saved documents

http://en.wikipedia.org/wiki/Global_warming
Global warming - Wikipedia, the free encyclopedia

http://www.pewclimate.org/
Pew Center on Global Climate Change

http://en.wikipedia.org/wiki/Global_warming
Global warming - Wikipedia, the free encyclopedia
... Global warming may also have positive effects. Plants form the basis of the biosphere ... and Petroleum: How Humans Took Control of Climate, New Jersey: Princeton

http://www.sierraclub.org/globalwarming/overview/healtheffects.asp
Overview: Health Effects - Global Warming - Sierra Club
Featured Factsheets: Get the Sierra Club Insider and stay current on environmental issues and activist opportunities. Overview: Health Effects. Millions worldwide may die

http://www.pewclimate.org/
Pew Center on Global Climate Change
Educates the public and key policy makers about the causes and potential consequences of climate change, and encourages the reduction of greenhouse gas emissions.

http://yosemite.epa.gov/oar/globalwarming.nsf/content/ImpactsStateImpacts.html
EPA : Global Warming : Impacts : State Impacts
EPA Global Warming: State Impacts provides state specific information sheets for the United States; the potential impacts of global warming or climate change are discuss

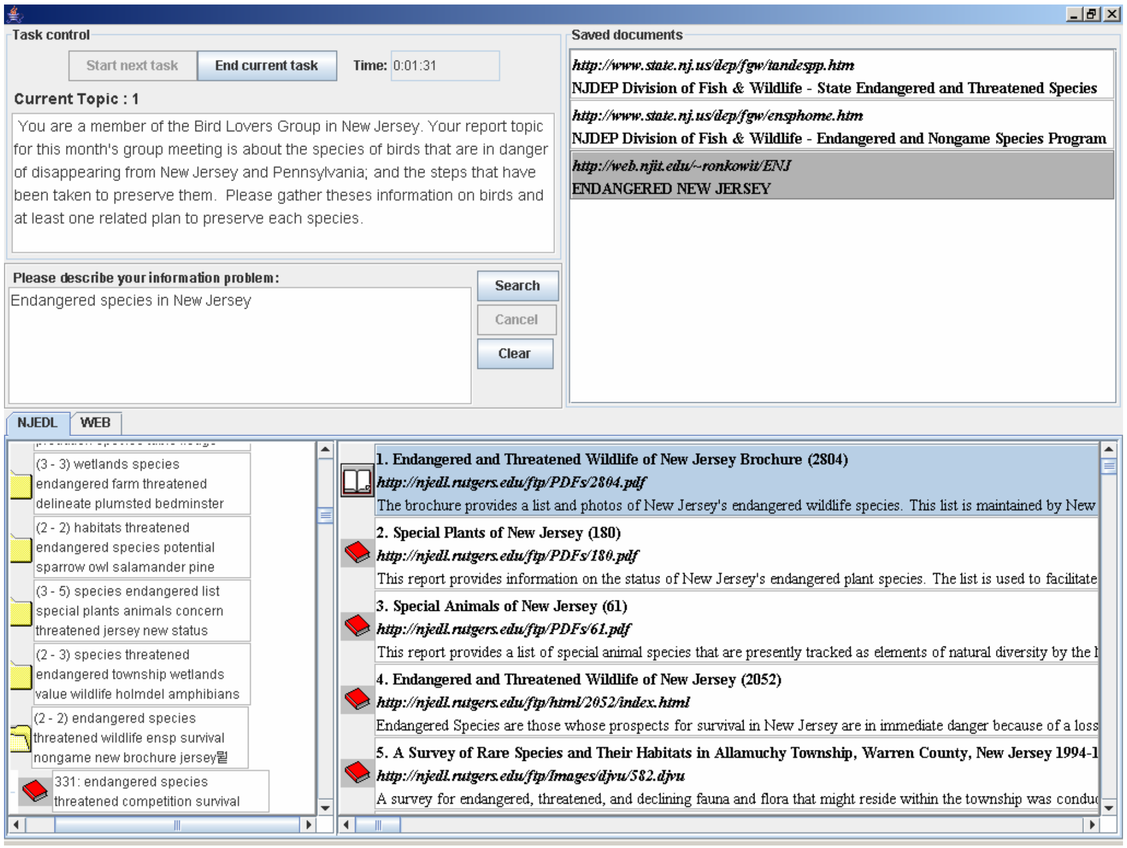
http://uspirg.org/uspirg.asp?id2=5235&id3=USPIRG&
U.S. PIRG

Figure 6 The baseline MIR interface (no mediation)

In the first stage of the project, which we have completed, the human searcher did not get any support from the system in formulating their queries to be submitted to the Web search engine. The mediation consisted in the user exploring the source collection in order to better understand the topic investigated, and to enrich her vocabulary. In a future stage of our investigation, the system will suggest a "mediated query" and the searcher will be able to edit it before submitting it to the search engine.

From among the candidate source collections that we were able to obtain, we selected the New Jersey Environmental Digital Library (NJEDL) collection because: (i) with approximately 1,300 documents, it is relatively small so, once clustered, it can be searched and browsed relatively easily in a reasonable amount of time; (ii) it provides a good coverage of environmental issues; (iii) we were able to generate a number of training and test topics for the experiment. A good test topic is one for which there are relevant documents in the target collection (the Web), but finding them requires good queries.

Our experimental design was inspired by work in Interactive TREC (Dumais and Belkin, 2005). We compared a baseline system, with no mediation, against the experimental system, based on mediation. Each of 16 subjects was randomly assigned a condition that specified the systems to be used and the topics to be investigated, two with the first system and another two with the second system. The systems and the queries were rotated in a Latin square design, in order to avoid any order effect. Figures 6 and 7 depict the user interfaces for the baseline and experimental systems.



Current Topic : 1

You are a member of the Bird Lovers Group in New Jersey. Your report topic for this month's group meeting is about the species of birds that are in danger of disappearing from New Jersey and Pennsylvania; and the steps that have been taken to preserve them. Please gather theses information on birds and at least one related plan to preserve each species.

Please describe your information problem:

Endangered species in New Jersey

http://www.state.nj.us/dep/fgw/tandespp.htm
NJDEP Division of Fish & Wildlife - State Endangered and Threatened Species

http://www.state.nj.us/dep/fgw/ensphome.htm
NJDEP Division of Fish & Wildlife - Endangered and Nongame Species Program

http://web.njit.edu/~ronkowit/ENJ
ENDANGERED NEW JERSEY

(3 - 3) wetlands species endangered farm threatened delineate plumsted bedminster

(2 - 2) habitats threatened endangered species potential sparrow owl salamander pine

(3 - 5) species endangered list special plants animals concern threatened jersey new status

(2 - 3) species threatened endangered township wetlands value wildlife holmdel amphibians

(2 - 2) endangered species threatened wildlife ensp survival nongame new brochure jersey뫱

331: endangered species threatened competition survival

1. Endangered and Threatened Wildlife of New Jersey Brochure (2804)
http://njedl.rutgers.edu/ftp/PDFs/2804.pdf
The brochure provides a list and photos of New Jersey's endangered wildlife species. This list is maintained by New

2. Special Plants of New Jersey (180)
http://njedl.rutgers.edu/ftp/PDFs/180.pdf
This report provides information on the status of New Jersey's endangered plant species. The list is used to facilitate

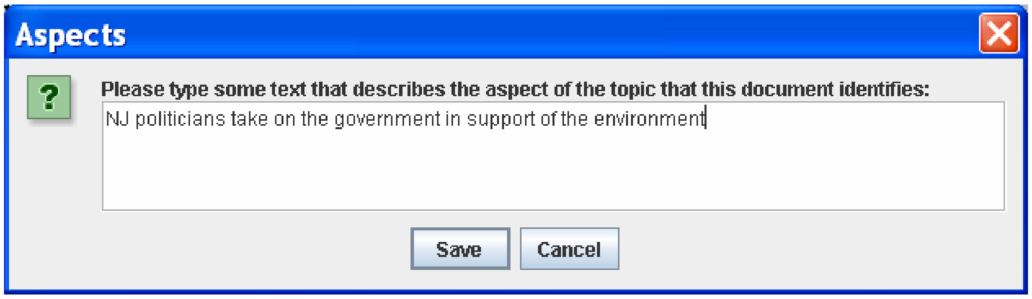
3. Special Animals of New Jersey (61)
http://njedl.rutgers.edu/ftp/PDFs/61.pdf
This report provides a list of special animal species that are presently tracked as elements of natural diversity by the l

4. Endangered and Threatened Wildlife of New Jersey (2052)
http://njedl.rutgers.edu/ftp/html/2052/index.html
Endangered Species are those whose prospects for survival in New Jersey are in immediate danger because of a loss

5. A Survey of Rare Species and Their Habitats in Allamuchy Township, Warren County, New Jersey 1994-1
http://njedl.rutgers.edu/ftp/Images/djvu/582.djvu
A survey for endangered, threatened, and declining fauna and flora that might reside within the township was conduc

Figure 7 The experimental system (with mediation)

An effort was made to make the systems identical, with the exception of the mediation functionality, so that any differences in results can be attributed to mediation. Each interface has a *Task control* panel where the task is displayed, and where the subject can formulate their information needs and submit them as queries. Search results from the target collection are shown in the "WEB" tab of the *Search results* panel. When a document is selected, it is displayed in the web browser. The subject can use the right mouse button to save a document from the hit list; the document snippet will be shown in the *Saved documents* panel. When a document is saved, the searcher is asked to specify the aspects of the topic that the document deals with, as shown in Figure 8. Retrieval effectiveness is measured both by recall (the number of relevant Web documents saved by the searcher, relative to the total number of relevant documents known by the researchers to be relevant) and aspectual recall (the number of distinct topical aspects identified by the searcher, relative to the total number of aspects found by the researchers). In order to identify relevant documents and aspects, we employed a **pooling** procedure similar to what has become a standard procedure in such IR experiments (Voorhees and Harman, 2005): we judged the relevance of the documents saved by all the subjects, and of the candidate documents identified by ourselves when exploring candidate test topics.


Aspects

Please type some text that describes the aspect of the topic that this document identifies:

NJ politicians take on the government in support of the environment

Figure 8 Capturing the aspects covered by a saved document

The experimental system has an additional tab, "NJEDL", which supports the exploration of the full source collection. The source collection is clustered, and the subjects can use a combination of searching and browsing for its exploration. On the one hand, searching can provide starting points for browsing: when a document snippet in the result list is selected, not only is the full document shown in the web browser, but the cluster hierarchy is expanded and scrolled automatically, so that the user can investigate the neighborhood of the selected document. On the other hand, browsing the clusters and documents of the source collection is expected to reveal serendipitous relevant information and to suggest new query terms.

At the beginning of the experiment subjects are given a tutorial, and the experimental system is demonstrated to subjects through the prescribed mediation interaction: after seeing the current topic, the searcher explores the source collection, available in the NJEDL tab, in order to understand the topic and its context better, and to grasp its terminology. Then, the interaction moves to the WWW tab, where a query can be submitted to the Web search engine, like in the baseline system. In the experiment the user is not forced to adopt this interaction model: if the topic is familiar and formulating a good query is perceived as easy, she may choose to go straight to the WWW tab and search the Web. However, the source collection is always available, and the searcher can always explore it; this may happen if the Web search is perceived as unsuccessful, and when more ideas for query terms are sought.

3 State-Based Design of Interaction and Logging

3.1 From UML to XML via XMI

The Unified Modeling Language (UML) defines a standard language and modeling notation for creating models of business and technical systems. It has the advantage that it can be easily understood not only by software engineers building systems, but also by non-technical people specifying the requirements of a system; in our context, the entire research team can collaborate to define the intended functionality of an interactive system, and in particular of an information retrieval system. UML is most often used as a blueprint for a system, facilitating communication between designers, and guiding the programmers building the system. It specifies not only the concepts and vocabulary of a system, but also the relationship between the concepts and the language that describes the functionality of the system.

UML can be used to define several types of diagrams that capture different aspects of a design (Fowler, 2004). *Class diagrams* describe the types of objects in the system and the various relationships between them. *Interaction diagrams* describe how groups of objects collaborate in order to implement a certain function. Of particular interest to us are *state diagrams*, used to describe the behavior of systems. When used in the design of user interfaces, state diagrams capture the stages of the interaction, define valid user actions at each stage, and specify the transitions brought about by various user actions.

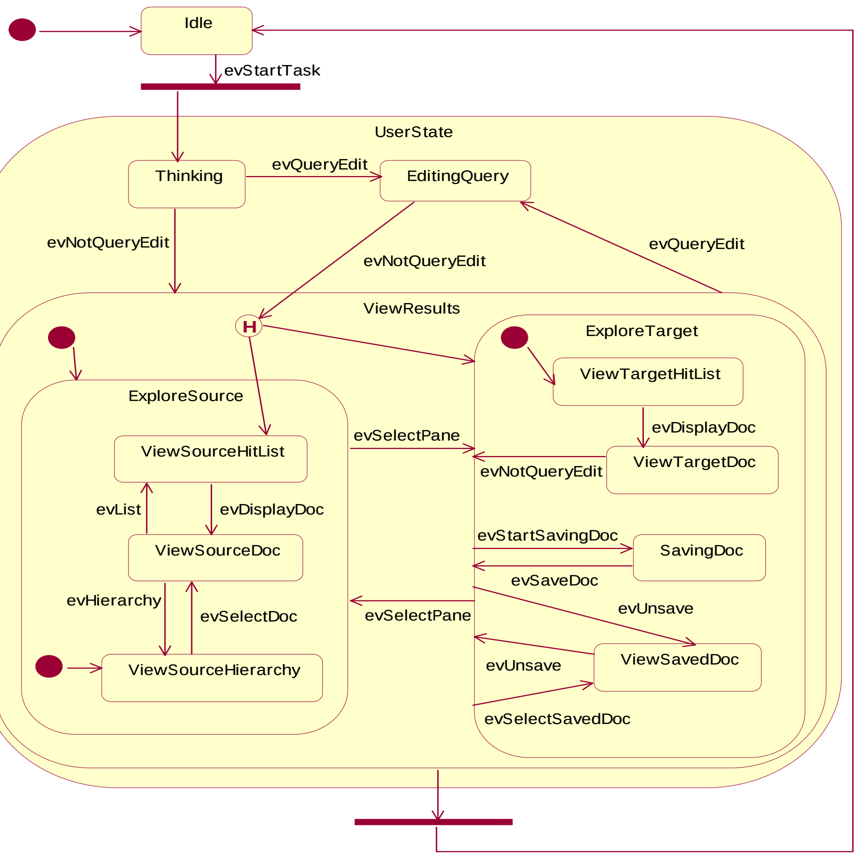


Figure 9 State diagram for the MIR project

To exemplify, Figure 9 shows the state diagram that depicts the system states during the MIR interaction. We believe that such a diagram is fairly easy to understand or design even for a researcher not trained in software engineering. In the Idle state between search sessions, the user may perform related activities such as filling in questionnaires required by the experiment. When the session starts, triggered by an evStartTask event, the system displays the current search task and enters the Thinking state, in which the subject reads the task description and thinks of appropriate queries (or alternative actions) to be used. If the user starts typing a query (marked by an evQueryEdit event), there is a transition into the EditQuery state. On the other hand, in the case of using the mediation system, the user has the choice of starting to browse the source collection first (marked by expanding the cluster hierarchy or selecting a cluster, i.e. an event different from query editing). While the user is editing the query (i.e. typing or using copy-and-paste), the system stays in the EditQuery state. When the ”Search” button is pressed, the history (H) pseudo-state will indicate which of the collections was being explored prior to editing the query; thus the query is submitted to the appropriate collection, the search results are displayed in the *Search results* panel of the appropriate tab, and the system enters the ViewResult state. This is a “superstate”, which has a number of “substates”: the ExploreSource state corresponds to the exploration of the source collection (NJEDL), while the ExploreTarget state corresponds to the exploration of the target collection (the Web). The searcher may choose between the two collections (and therefore between the two sub-states) by selecting one of the tabs, or the sub-state may be set automatically by the history mechanism.

The granularity of the states depends on the desired precision of modeling the interaction; it is typically dictated by the intended functionality of the system, but it may also be informed by the type of research hypotheses under study. For the MIR project, we considered a second level of substates. When exploring the source collection, the user may be browsing the cluster hierarchy (ViewSourceHierarchy) or scanning the list of search results (ViewSourceHitList), or selecting and viewing one of the documents (ViewSourceDoc). When exploring the target collection, the user may be scanning the list of hits (ViewTargetHitList), or may be viewing a selected hit (ViewTargetDoc) or may be in the process of saving a search result judged relevant, and typing in the aspects covered by that document (SavingDoc), or may have second thoughts and look again at a saved document trying to decide whether to unsave it (ViewSavedDoc). The searcher can shift focus of the exploration between the source and target collections (the evSelectPane event triggers this shift). This choice is captured by the history pseudo-state (H), which dictates if future transactions from EditQuery should go to ExploreSource or ExploreTarget following a query submission to the search engine.

Not depicted in this diagram are the orthogonal (or parallel) states, corresponding to different components of the system such as the *Task control* panel and the *Search results* panel. These states can also be modeled at different levels of granularity in order to support the design and implementation of the system. For example, the Query panel can be in a Valid state, when a query can be submitted, or an Invalid state, when there is no query, or a query has just been submitted and the search results are expected from the search engine. These system states, parallel to the user states (and hence the two synchronizations bars in the diagram), are essential in designing the functionality of the system. However, they are omitted here for space reasons.

A couple of clarifications are in order:

- Although think-aloud protocols can help, it is not possible to have a perfect image of the searcher’s cognitive process. Therefore, what is represented in the diagrams is not user cognitive states, but system states. However, the user’s actions and the sequence of system states do reflect the decisions taken by the user, and can therefore be used in modeling user behavior.

- The labels assigned to system states reflect the researchers' understanding of the interaction, and specify their interpretation of what is going on. Like variable names in programming, these labels should convey the semantics of the interaction; however, a perfectly accurate depiction of the user's cognitive process is not necessary. In the example, the label "Thinking" was assigned to the state in which the searcher was instructed to read the assigned task and to think of a search query to submit. There is no guarantee that the user follows the instructions and is indeed thinking; conversely, it does not mean that this is the only state in which the user has to think. The label simply attempts to depict the researcher's best description of what is going on.

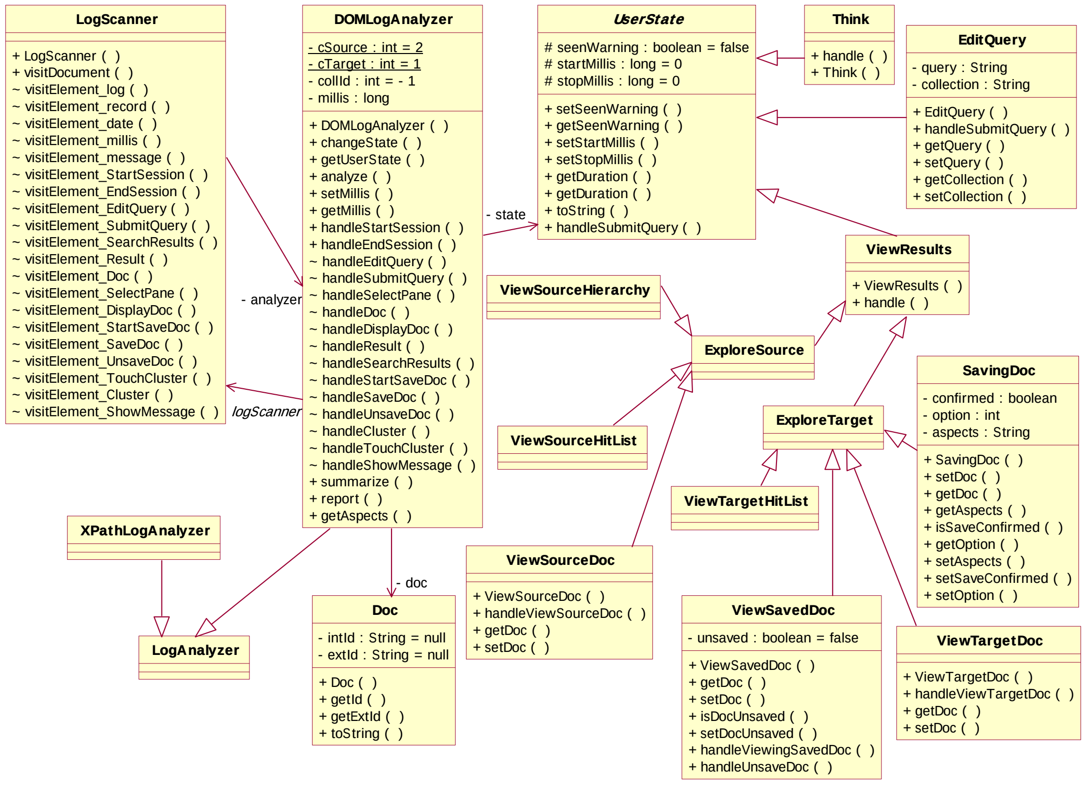


Figure 10 State classes used in the MIR log analyzer

3.2 *Explicit vs. implicit logging of states*

At first sight, explicitly logging the system states appears natural, so that someone examining the logs can clearly see what happened while the system was in a certain state, and when a state transition occurred. However, logs are usually so large and contain so many details, that the researcher is unlikely to gain much knowledge from examining them visually. Rather, the logs should be processed automatically and the information pertinent to a certain research question should be summarized, and possibly visualized, so that it can be interpreted by the researcher. Therefore, *explicitly* capturing the states in the logs is not necessary, as long as they can be re-created at analysis time, based on the events and actions captured in the logs, and on the model captured by the state diagrams.

One problem with capturing snapshots of system states is the issue of capturing attributes of the state transitions. For example, the user submitting a query generates a state transition. The attributes of this event, such as the text of query, the targeted search engine or the number of hits requested, are usually important for the research hypotheses investigated. However, these attributes of an event are not captured in the UML diagrams, are not translated into the XML schema and, in a naïve application of the method discussed, are not captured in the logs. It is apparent that it is the events or user actions that define the interaction, and they should be captured in interaction logs. Moreover, even if they were not logged, the states of the system could be re-created based on the state diagram and on logged events, which uniquely determine the state transitions. While the argument in favor of logging events is compelling, the logging of the actual system states appears to be optional.

There are, however, several arguments in favor of not capturing the states explicitly, and in having the log analyzer infer them. First, complex systems such as the user interface of a search engine are likely to have complex states, with nested sub-states, and often have concurrent orthogonal states. For example, not depicted in Figure 9 are orthogonal states that describe the connectivity of the system with the Internet. If the system detects a drop in connectivity that would affect the normal running of an experiment, then the normal functionality of the user interface would be over-ridden. Attempting to log the parallel states and the transitions is likely to produce nesting that cannot be captured in a well-formed XML document.

Another advantage of capturing only events and actions in the log and re-creating the states via the log analyzer is that other interaction logs, obtained from previous experiments, or from experiments run by other researchers, can be analyzed based on the same approach, as long as the state diagram is known.

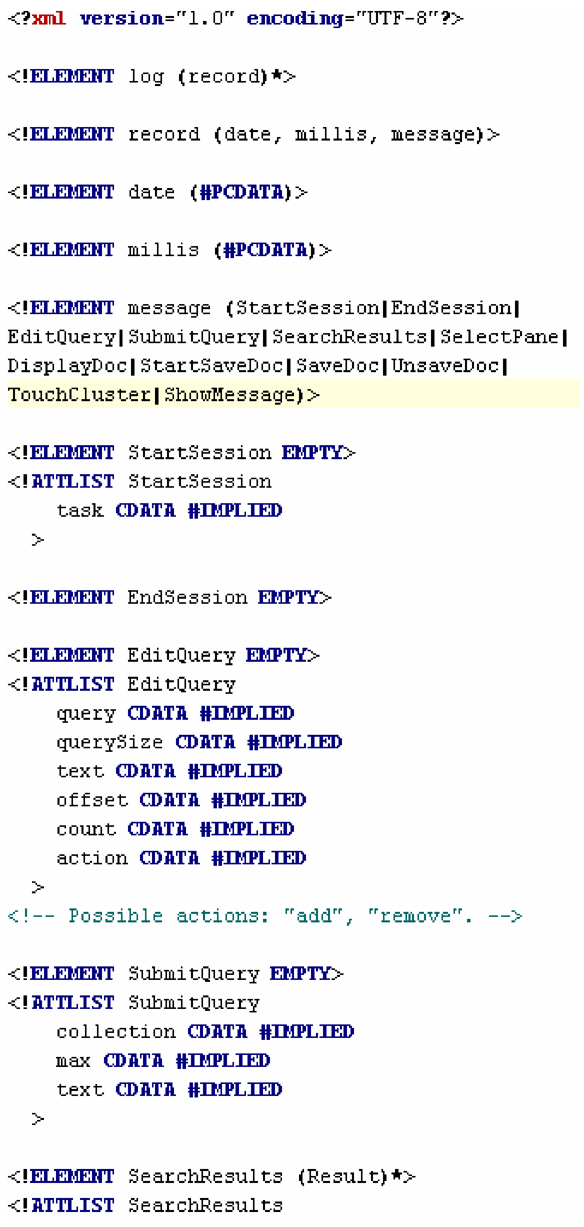
3.3 Our approach to integrating interaction design and log analysis

The above analysis and the decision to focus on logging state transitions suggest an extra step to the direct approach discussed in section 1.2. From the state diagram (exemplified in Figure 9), we derive a class diagram that captures the events that determine state transitions: each state transition corresponds to a class, and the attributes of the class describe the attribute of the event (see Figure 10). Therefore, the new UML class diagram captures the **Interaction Modeling Language** (**IML**) for the user interface. It is from this intermediary diagram that the DTD and/or XML schema are derived via the XMI mapping.

Figure 11 presents a sample of the DTD that describes the MIR interaction, and Figure 12 depicts a sample extracted from a MIR log. It is apparent that the attributes of the events, such as the editing or submission of a query, are captured in the logs and can be used to address the research hypotheses. Moreover, as the IML captures just the interactions, and the log records just the events, not the states, we are able to produce well-formed XML documents (because events do not overlap, as orthogonal states may do). However, based on the state diagram, the states can be re-created while the logs are parsed and the events interpreted. This supports research in analyzing state transitions and modeling user behavior.

Apart from being the source of the XML schema, the interaction diagram supports the automatic code generation for two software modules: (i) the logger that records each valid event and action that takes place, and each state transition undergone by the system, recording in the log the time stamp and the attributes of these events; (ii) a log analyzer that uses an XML parser and identifies events, actions and state transitions, and analyzes the data according to the research hypotheses being investigated.

This can be done with existing open-source tools (such as NetBeans[p]) that can automatically generate code, given the adopted DTD or XML schema.

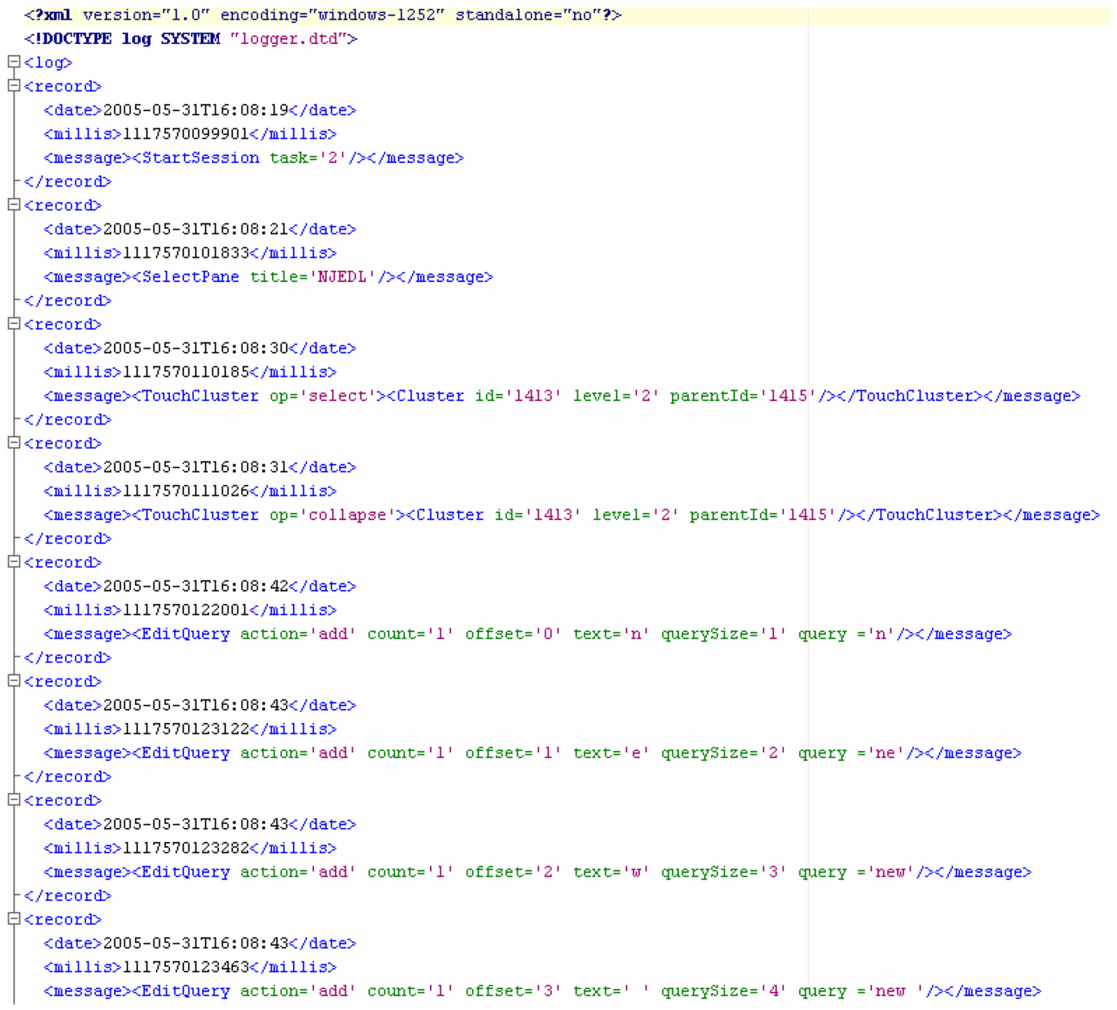


```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT log (record)*>

<!ELEMENT record (date, millis, message)>

<!ELEMENT date (#PCDATA)>

<!ELEMENT millis (#PCDATA)>

<!ELEMENT message (StartSession|EndSession|
EditQuery|SubmitQuery|SearchResults|SelectPane|
DisplayDoc|StartSaveDoc|SaveDoc|UnsaveDoc|
TouchCluster|ShowMessage)>

<!ELEMENT StartSession EMPTY>
<!ATTLIST StartSession
    task CDATA #IMPLIED
  >

<!ELEMENT EndSession EMPTY>

<!ELEMENT EditQuery EMPTY>
<!ATTLIST EditQuery
    query CDATA #IMPLIED
    querySize CDATA #IMPLIED
    text CDATA #IMPLIED
    offset CDATA #IMPLIED
    count CDATA #IMPLIED
    action CDATA #IMPLIED
  >
<!-- Possible actions: "add", "remove". -->

<!ELEMENT SubmitQuery EMPTY>
<!ATTLIST SubmitQuery
    collection CDATA #IMPLIED
    max CDATA #IMPLIED
    text CDATA #IMPLIED
  >

<!ELEMENT SearchResults (Result)*>
<!ATTLIST SearchResults
```

Figure 11 Sample from the MIR interaction DTD

[p] http://www.netbeans.org/

Note that the code generated is just a skeleton, and the research team needs to fill in the class methods with actual code that writes or reads data into or from a file. However, such code is trivial after the design of classes and methods has been generated. For writing, if Java is the implementation language, then the standard logging package[q] makes it extremely simple to output logs in XML: a Logger object uses XML by default to write logs into a file, adds a timestamp automatically, and displays as content of a “**message”** element the text passed to it for logging (see Figure 12).



```
<?xml version="1.0" encoding="windows-1252" standalone="no"?>
<!DOCTYPE log SYSTEM "logger.dtd">
<log>
<record>
  <date>2005-05-31T16:08:19</date>
  <millis>1117570099901</millis>
  <message><StartSession task='2'/></message>
</record>
<record>
  <date>2005-05-31T16:08:21</date>
  <millis>1117570101833</millis>
  <message><SelectPane title='NJEDL'/></message>
</record>
<record>
  <date>2005-05-31T16:08:30</date>
  <millis>1117570110185</millis>
  <message><TouchCluster op='select'><Cluster id='1413' level='2' parentId='1415'/></TouchCluster></message>
</record>
<record>
  <date>2005-05-31T16:08:31</date>
  <millis>1117570111026</millis>
  <message><TouchCluster op='collapse'><Cluster id='1413' level='2' parentId='1415'/></TouchCluster></message>
</record>
<record>
  <date>2005-05-31T16:08:42</date>
  <millis>1117570122001</millis>
  <message><EditQuery action='add' count='1' offset='0' text='n' querySize='1' query ='n'/></message>
</record>
<record>
  <date>2005-05-31T16:08:43</date>
  <millis>1117570123122</millis>
  <message><EditQuery action='add' count='1' offset='1' text='e' querySize='2' query ='ne'/></message>
</record>
<record>
  <date>2005-05-31T16:08:43</date>
  <millis>1117570123282</millis>
  <message><EditQuery action='add' count='1' offset='2' text='w' querySize='3' query ='new'/></message>
</record>
<record>
  <date>2005-05-31T16:08:43</date>
  <millis>1117570123463</millis>
  <message><EditQuery action='add' count='1' offset='3' text=' ' querySize='4' query ='new '/></message>
```

Figure 12 Sample from a MIR log

Even if not used directly in generating the XML schema of the interaction and subsequently the code for log recording and parsing, the original state diagram describing the states of the system

[q] http://java.sun.com/javase/6/docs/technotes/guides/logging/index.html

(Figure 9) can be used for automatically generating code for modeling state transitions and, for example, building a Markov model of user behavior. Note that the classes depicted in Figure 12 are not simply used as an intermediary step to derive the interaction schema. They represent the actual classes (in an object-oriented programming language such as Java) of the log analyzer, and of the software for state modeling; quite obviously, the classes correspond to the states of the interaction. State objects can capture events that took place for the duration of that state, and additional data structures can capture the sequence of states in chronological order.

4 Discussion and evaluation

The effectiveness of a methodology is best demonstrated by its flexibility as well as its ability to solve the problem it was designed for. In this section we discuss a number of design decisions that can be taken to customize the methodology, and demonstrate its power based on anecdotal evidence from our experiments, as exemplified by the kind of data analysis and research hypotheses investigation that it supports.

4.1 Design patterns in the log analyzer

Parsing XML has become routine due to the multitude of open-source parsers and parser generators available for a variety of programming languages. For extremely large logs, unlikely to fit in the computer memory for the analysis, a SAX (Simple API for XML) parser is needed. This type of parser identifies the beginning and end of various elements found in the log, and processes them based on the callback methods provided by the programmer/researcher. The more desirable approach, although restricted to logs of reasonable size, which fit in the random-access memory, is to use a DOM (Document Object Model) parser, which builds a log tree model, in which each XML node corresponds to a certain event or action, and allows the programmer to visit it in whatever order makes sense for investigating a certain research hypothesis. For example, if the research hypothesis being investigated is related solely to the documents saved by the searcher, it is possible and easy to visit just the nodes capturing document saving.

It is common for XML parsers generated automatically based on DTD (such as the one produced by NetBeans) to implement the **Visitor** software design pattern, which allows flexibility in specifying which elements of the log tree should be visited and in what order, in order to collect, process and summarize information. From our experience, we suggest combining that with the **State** design pattern, where different classes correspond to states in the state diagram. This allows the state objects to accumulate, summarize and report information in a simple and flexible fashion (Gamma et al, 1995).

The parsing code generated automatically is just a skeleton providing functionality for visiting the log tree, and the research team needs to provide code for extracting the required data and for conducting the intended analysis, according to the research hypotheses. Therefore, some familiarity with the style of programming specific for building user interfaces and for analyzing user interface interaction is recommended.

Unfortunately, the software engineering or interface design literature that would support this style of design and implementation is rather weak and inconsistent. Some authors do a good job of explaining the use of statecharts when designing user interfaces, but the implementations proposed are procedural, rather than class-based (Horrocks, 1999). Apart from being inelegant and difficult to maintain, such solutions do not support state inheritance, and do not support the accumulation of state information. Other authors do indeed recommend a design solution based on the State design pattern, but they only consider simple statechart examples, with no sub-states and no orthogonal states (Fowler, 2004).

We recommend the use of class inheritance to implement sub-states and the use of composition for capturing concurrent orthogonal states. For example, in the MIR experiment scanning the list of search results is one way to explore the target collection, so it makes sense to make ViewTargetHitList a subclass of ExploreTarget. Figure 10 captures the entire set of classes used to model the user states in MIR, described by the state diagram in Figure 9.

Other decisions are less obvious. When implementing the State pattern, a decision needs to be made as to who controls the state transitions: the class representing the context (the log analyzer in our project) or the classes representing states (Gamma, 1995). While much of the literature seems to suggest that one of the two options should be adopted and applied consistently, a pragmatic combination can be employed in practice. Some events always trigger transition to a particular state; in MIR, for example, evStartTask always triggers a transition to Thinking, and evQueryEdit to EditQuery. In such cases, the context can control the transition and the setting of the new state. Other transitions and effects of the transitions are more complex and may depend on the current state; in such situation the “cleaner” solution is for the handling of the event to be delegated to the state itself (e.g. SavingDoc handles evSaveDoc).

4.2 Singletons vs. multiple objects for states

Another essential decision is how the state objects are created and stored when analyzing the logs. One popular solution is to apply the **Singleton** design pattern (Gamma, 1995), so that a unique (singleton) object is created for each state. This is typically the preferred solution when an application has a small number of states and a large number of state transitions: state objects can be reused rather than new objects created, which makes the application more efficient. Also, a state object can accumulate information over multiple occurrences of the same conceptual state. While in most situation using the singletons is the better solution, for our specific application that solution is not appropriate, due to the level of detail that we want to capture. For example, we want to analyze not only how many queries were edited and submitted overall, but also how much time was spent formulating each of them, if words were typed or pasted into the query box, the number of corrections that were made on the query etc. For capturing specific information for each instance of a state, we adopted the solution of creating a new state object every time a state transition occurs; for example, each EditQuery object captures the interaction related to a different query, rather than accumulating information about all the queries.

Think	4
EditQuery	9
ViewTargetHitList	15
ViewTargetDoc	78
SavingDoc	16
ViewTargetHitList	6
ViewTargetDoc	31
ViewTargetDoc	9
ViewTargetDoc	35
SavingDoc	11
ViewTargetHitList	3
ViewTargetDoc	173
SavingDoc	16
ViewTargetHitList	14
EditQuery	7
ViewTargetHitList	4
ViewTargetDoc	17
ViewTargetDoc	59
ViewTargetDoc	51
ViewTargetDoc	39
EditQuery	13
ViewTargetHitList	25
ViewTargetDoc	38
SavingDoc	15
…	

When the log is analyzed, a number of state objects are created and stored in a list. There is a lot of flexibility on how these objects are subsequently processed. For answering a certain research hypothesis, the list of state objects can be filtered so that only objects of a certain class are kept, and the information stored by them can be summarized and analyzed.

4.3 Log Analysis Results

The focus of this paper is on the methodology for analyzing the interaction rather than on the actual results. Therefore, no comprehensive analysis of the MIR logs is included.[r] The purpose of this section is to demonstrate the kind of analysis supported by our methodology.

First of all, let us distinguish between two fundamentally different approaches to analyzing the logs. The "atemporal" approach can be applied when the interest is in processing information about a certain kind of event, with no regard to state transitions, or to the order of the states in the logs. Examples of such situations are: getting the list of all the documents viewed or saved by the user, getting the list of all queries submitted to the search engine, etc. In such situations, probably the most efficient solution is to implement an XPathLogAnalyzer, which uses XPath to visit only the XML nodes in the log tree that are of interest (for example, the SaveDoc events can be visited by specifying "/log/record/message/SaveDoc" as the path to the nodes of interest).

If the time factor is essential in answering a certain research hypothesis or in getting a certain kind of information, then a DOMLogAnalyzer[s] can be employed instead, which will traverse and process the nodes of the log tree (in XML format) in the desired order. For more flexibility, the task of actually traversing the log tree can be delegated to a separate class (LogScanner in Figure 12), so that the function of traversing the log is decoupled from the function of taking action for each node. An even more flexible solution is to apply the **Strategy** design pattern (Gamma, 1995), by making LogScanner an abstract class and having different visiting strategies implemented by its concrete subclasses.

Let us now have a look at a sample of results obtained by applying this methodology in MIR. The inset text-box shows a sample report obtained by listing the class names for each state object inferred from a log file, together with the duration of that state (in seconds). Subsequent processing could consist, for example, in building a transition matrix by compiling the states from all the log files in order to (i) observe patterns of behavior and be able to predict the next state at a given point; or to (ii) find what are the most common states and most common transitions, and optimize the use of the interface for those situations; or to (iii) detect and correct usability problems (e.g. detecting transitions that never happen, because some functions are not sufficiently visible in the user interface).

An essential piece of analysis for the MIR project regards the effectiveness of retrieval; we are interested to see whether mediation improves effectiveness. The computation of recall and aspectual recall requires relevance judgments. Even without those, a simple extraction and comparison of data from the logs can give us an idea of how well our expectations were met. Note that in previous experiments, run as part of Interactive TREC, a high correlation was observed between recall and the raw number of documents saved by the subjects (Belkin et al, 2001). Moreover, in the current experiment, the subjects were asked to support their decision to save each document by stating the aspects addressed by the document; therefore, one can expect most saved documents to be relevant, and a higher than usual correlation between recall and number of documents saved. Obtaining from the logs the number of saved documents and the number of queries submitted is trivial.

For the sake of exemplifying some of the statistical analysis supported by our approach, let us report that a set of ANOVA tests shows that most differences between the non-mediated and the mediated conditions are not statistically significant. Surprisingly, slightly more documents were saved on average in the non-mediated condition (m = 3.94, sd = 1.76) than in the mediated condition (m =

[r] Detailed results and conclusions from the MIR project are reported in Lee's PhD dissertation (2006), supervised by this author.

[s] Names such as XPathLogAnalyzer or DOMLogAnalyzer are by no means standard names. They were chosen in the MIR project to indicate that the scanning of the logs was based on XPath, respectively on traversing the DOM tree.

3.13, sd = 1.62) despite visibly more effort in the mediation condition. While spending roughly the same total amount of time in the overall search session (m = 1166.16, sd = 185.98 compared to m = 1190.91, sd = 168.91)[t], the mediation subjects submitted significantly more queries (m = 8.69, sd = 4.90 compared to 5.69, sd = 3.22; F = 8.377, p = 0.005). In the mediation condition, subjects submitted an average of 2.22 queries to the source collection, and an average of 6.47 queries to the target collection.

Unfortunately, this is a bad result for the mediation hypothesis. Possible explanations are that (i) the subject could not find relevant documents in the source collection; or (ii) the subjects did not have time to read the identified source documents in order to improve their understanding of the topic or to enhance their terminological vocabulary in order to submit better queries. In order to answer these questions, our next steps are to examine the source documents viewed by the users (captured in the interaction logs) and to judge their relevance to the test topics. This will allow us to check if the statistical language models of the queries submitted following mediation show any significant difference. This shows the power and flexibility of our methodology – the accurate logging of all semantic events, even those not related to the research hypotheses, affords the extension of the original hypotheses, and extra analysis not planned at the outset.

5 Contributions and Future Work

5.1 Contributions and limitations of the proposed methodology

The proposed methodology is a novel and significant contribution to experimental research in interactive systems, with applications in areas such as Human Computer Interaction or Information Seeking and Retrieval. It is particularly suitable for studying exploratory searching, where the research questions are usually related to understanding patterns of behavior in different stages of the interaction. This approach has been successfully applied in Interactive TREC work and in the Mediated Information Retrieval project.

One interesting issue to consider is the generality of our approach. What kind of systems can it be applied to ? Is it not rather limiting to restrict logging to semantic events ? Is it possible to log everything that happens during the interaction ? We will start addressing these issues by re-iterating the purpose of our work. We intended to integrate the design of the user-system interaction (and implicitly of the user interface) with the design of the logger and of the log analyzer. This means the following:

- The user should be limited to performing actions judged by the system designer to be valid in a certain context; e.g. the user cannot submit an empty query, or save a document repeatedly etc. It means that only valid actions should be recorded in the logs. During testing, assertions in the log parsing software can help make sure that the XML documents perfectly match the interaction specification (the XML schema), and that all the recorded events and state transitions are valid.

- It is debatable whether user attempts to perform invalid actions (e.g. the attempt to re-submit a query while the search is active), or events ignored by the system (erratic moves of the mouse) should be logged. On the one hand, only lack of imagination can limit the system designer's as to what should be logged, so the danger of recording too much irrelevant data is real (e.g. if a dedicated thread records the state of the system second by second). On the other hand, recording data that is judged irrelevant at the onset may be valuable if the relevance judgment is reconsidered, for example if new research hypotheses are proposed following the initial analysis of an experiment's logs. While recommending a

[t] The subjects were told that they had 20 minutes (or 1200 seconds) for investigating each topic.

balance between the extremes, we have addressed this issue by including a special action called *ShowMessage* (see the DTD in Figure 11), which records “other” events, i.e. events not included among the valid semantic events in the interaction design. In our own research experiment, we used this capability to record when the task panel’s timer alerts the subject that just two minutes are left for completing the task; this is an event that does not affect the state of the system and can be ignored by the user. However, recording that event allows us to determine if the reminder affected the user’s subsequent search behavior.

- On a related note, the designers need to decide the granularity of the events to be logged. For example, should the system log each keystroke used to edit a query, or just the final query ? Our recommendation is to let the research hypotheses under investigation inform the decision. For example, we were interested in the effect of topic familiarity on the searcher’s query formulation behavior (copying and pasting vs. typing, number of corrections made, etc), so we logged all keystrokes. On the other hand, we only logged the mouse events that had semantic interpretation (selection, cluster expansion, etc).

- Similarly, the system designer needs to decide whether orthogonal events (e.g. the search thread becoming active, or the Internet connection being lost etc) are worth logging, at the expense of more design and implementation time. Our approach is applicable in two ways: (i) the state diagrams are built separately, and the logging is done in separate files; synchronization of logs, based on time-stamps may be required at analysis time; (ii) more complex state diagrams are used, with parallel swim-lanes, and all the events are logged into the same file; the disadvantage is the increased complexity of the software.

Our proposed approach is appropriate for client-side logging, especially when the research team design and implement both the user interface software (which includes the conceptual interaction design) and the log analyzer. In this situation, the same class hierarchy, representing system states, can be used for implementing both the interaction with the user (keystrokes and mouse actions are interpreted in terms of semantic actions according to the state of the system) and the log analyzer (logged events are interpreted in order to re-create the system states). The proposed approach can be adapted in the following situations, with gradually increasing levels of difficulty:

- For adding logging and analysis functionality to existing code. The state diagram of the interaction needs to be reverse-engineered based on the code and on observing the functionality of the system. While the benefits of an integrated design are lost, the logging of the events and analysis of the logs works well.

- For analyzing existing logs produced by a different system. The success of our state-based approach depends on the quality of the user interface that generated the logs (whether it allows or not invalid events to take place and to be logged) and the amount of events logged (whether the sequence of events can unambiguously predict the sequence of system states).

- For server-side logging, our approach is only feasible if the logged information is sufficient to determine the client that generated each event, and if the states of the client can be predicted based on the logged events. It is not appropriate, for example, for analyzing weblogs of HTTP requests.

5.2 Future research directions

One issue that we are currently investigating is an extension of this methodology to studying patterns of behavior by building Hidden Markov Models (HMM) based on the analysis of state transitions recorded in the logs (Jurafsky, 2000). One decision in building such models regards the computation of the transition probabilities. The two potential approaches are based on: (i) macro statistics – the transitions are counted and the probabilities are computed for each individual user, then the probabilities are averaged over the users; and (ii) micro statistics – the transitions are counted and the probabilities computed over all the user logs. The former approach is expected to highlight the differences between individual subjects, and the latter to show common behavior. Both approaches should probably be used so that together they paint a better picture of what is happening. Moreover, where the difference between individual and common behavior is significant, correlations with individual factors (such as familiarity with the topic) should be sought.

Considering the hierarchical structure of states, it is obvious that another issue to consider is state granularity. Taking into account just the top levels may give too coarse a view of the interaction and may not provide sufficient details to answer research questions. On the other hand, the leaf states may provide too much detail and may hide patterns in higher levels; moreover, due to the limited amount of data generated in a lab user experiment, some of the leaf states may appear infrequently, so drawing conclusions from such sparse data may be dangerous. It is probably better to repeat the analysis for different levels of granularity or to smooth detailed interaction models with models built for transitions between high granularity states.

Actually, the analysis described above may prove that, for complex interactions such as information seeking, pure Markov Models may prove inappropriate, and that more complex extensions should be considered. It may be the case that state transitions are not determined just by the current state and certain events, but also by some parameters of the state, such as the amount of time spent, or the number of documents examined.

A very different research direction is to investigate ways to automatically generate graphical diagrams that show the frequency of each state transition and thus give a visual display of user behavior. So far we have extracted transition frequencies with the log analyzer, but have built the diagrams manually.

Also, we subscribe to efforts for standardization of log formats in certain types of applications, such as user interfaces for digital libraries (Gonçalves et al, 2003; Klas et al, 2006). Moreover, we suggest that our approach of deriving logging formats from user interface design should help the effort: the functionality provided by such user interfaces should be first standardized in UML format, and then standardization of the log formats can be achieved as an immediate consequence.

Finally, we intend to investigate a number of IR user interfaces and to compare their state diagrams, trying to identify common patterns. This would allow us to provide support, in the form of reusable toolkits of frameworks, for researchers designing and evaluating user interfaces for Information Retrieval.

Acknowledgements

The methodology proposed and discussed here was successfully used in designing the user interfaces and analyzing the logs for the Mediated Information Retrieval (MIR) project, conducted in the School of communication, Information and Library Studies, Rutgers University. This author thanks Hyuk-Jin Lee, who conducted the user experiment, Nicholas J. Belkin and Dan O'Connor, supervisors, and David J. Harper from the School of Computing, The Robert Gordon University, UK, external advisor.

References

1. Belkin, N.J., Cool, C., Stein, A., Thiel, U. (1995) Cases, scripts, and information-seeking strategies: on the design of interactive information retrieval systems. *Expert Systems with Applications, 9*(3), 379-395.

2. Belkin, N. J., Cool, C., Kelly, D., Lin, S.-j., Park, S., Perez-Carballo, J. and Sikora, C. (2001) Iterative exploration, design and evaluation of support for query reformulation in interactive information retrieval, *Information Processing & Management*, 37(3): 403-434.

3. Belkin, N.J., Cool, C., Kelly, D., Kim, G., Kim, J.-Y., Lee, H.-J., Muresan, G., Tang, M.-C., Yuan X.-J. (2002) Rutgers Interactive Track at TREC 2002, in *Proceedings of TREC 2002*, Gaithersburg, November 2002.

4. Carlson, D. (2001) *Modeling XML applications with UML: Practical e-Business applications*, Addison-Wesley, ISBN: 0-201-70915-5.

5. Carlson D. (2006) Semantic models for XML schema with UML tooling. *Proceedings of the 2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE)*, Nov 2006, Athens, GA..

6. Crowle, S. and Hole, L. (2003) ISML: An interface specification meta-language, *10th International Workshop on Design, Specification and Verification of Interactive Systems*, Madeira.

7. Diaper, D. and Stanton, N. (2004) *The handbook of task analysis of Human-Computer Interaction*, Lawrence Erlbaum Associates, ISBN 0-8058-4433-3.

8. Douglass, B. P. (1999) Doing hard time: Developing real-time systems with UML, objects, frameworks, and patterns, Addison-Wesley, Reading, MA.

9. Dumais, S. T. and Belkin, N. J. (2005) The TREC Interactive tracks: Putting the user into search, in *TREC – Experiment and evaluation in Information Retrieval*, eds. Voorhees, E. M. and Harman, D. K., MIT Press, Cambridge, MA, ISBN 0-262-22073-3.

10. Ellis, D. (1989) A behavioral approach to information retrieval system design. *The Journal of Documentation*, 45(3), 171-212.

11. Fisher, K. E., Erdelez S. and McKechnie, L. (2005) *Theories of Information Behavior*, Information Today, Medford, NJ.

12. Fowler, Martin (2004) *UML distilled: A brief guide to the standard object modeling language*, 3rd ed, Addison-Wesley/Pearson Education.

13. Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995) *Design Patterns – Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, MA.

14. Gonçalves, M. A., Panchanathan, G., Ravindranathan, U., Krowne, A., Fox, E. A., Jagodzinski, F. and Cassel, L. (2003) The XML Log Standard for Digital Libraries: Analysis, Evolution, and Deployment. *The Third Joint Conference in Digital Libraries (JCDL)*, Houston, Texas, May 2003.

15. Harel, D. (1988) On visual formalisms, *Communications of the ACM*, 31 (5).

16. Hersh, W. (2002) TREC 2002 Interactive Track Report, *Proceedings of TREC 2002*, Gaithersburg, Nov 2002.

17. Horrocks, I. (1999) Constructing the User Interface with Statecharts, Addison-Wesley, ISBN 0-210-34278-2.

18. Ingwersen, P. and Jarvelin, K. (2005) *The Turn – Integration of Information Seeking and Retrieval in Context*. Springer.

19. Jurafsky, D. and Martin, James H. (2000) *Speech and language processing*, Prentice-Hall, ISBN 0-13-095069-6.

20. Klas, C.-P., Albrechtsen, H., Fuhr, N., Hansen, P., Kapidakis, S., Kovacs, L., Krievel, S., Micsik, A., Papatheodorou, C., Tsakonas, G. and Jacob, E. (2006) A Logging Scheme for Comparative Digital Library Evaluation, in Proceedings of the 10th European conference on research and advanced technology for digital libraries (ECDL 2006), Alicante.

21. Kuhlthau, C. (1991) Inside the search process: information seeking from the user's perspective. *Journal of the American Society for Information Science, 42*(5), 361-371.

22. Lee, H.-J. (2006) Mediated Information Retrieval for the Web Environment, Ph.D. dissertation, School of Communication, Information and Library Studies, Rutgers University, New Brunswick, NJ, May 2006.

23. Limbourg, Q. and Vanderdonckt, J. (2004) Comparing task models for user interface design, in Diaper, D. and Stanton, N. (eds.) *The handbook of task analysis of Human-Computer Interaction*, Lawrence Erlbaum Associates, ISBN 0-8058-4433-3.

24. Muresan, G. (2002) Using Document Clustering and Language Modelling in Mediated Information Retrieval, Ph.D. dissertation, School of Computing, The Robert Gordon University, Aberdeen, Scotland, January 2002.

25. Muresan, G. and Harper, D. J. (2001) Document Clustering and Language Models for System-Mediated Information Access in *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries*, Darmstadt, 4-9 September 2001, 438-449, ISBN 3-540-42537-3.

26. Muresan, G. and Harper, D. J. (2004) Topic Modelling for Mediated Access to Very Large Document Collections, *JASIST* 55 (10): 892-910, Special Topics Issue: Document Search Interface Design for Large-Scale Collections and Intelligent Access, August 2004.

27. Olah, J. (2005) Shifts Between Search Stages During Task-Performance in Mediated Information-Seeking Interaction, *Proceedings of the 68th Annual Meeting of the American Society for Information Science (ASIST),* 42, Charlotte, NC.

28. Paterno, F. (2001) Towards a UML for Interactive Systems, *Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction*, Toronto, May 2001, 7-8, ISBN:3-540-43044-X.

29. Paterno, F. (2004) ConcurTaskTrees: an engineered notation for task models, in Diaper, D. and Stanton, N. (eds.) *The handbook of task analysis of Human-Computer Interaction*, Lawrence Erlbaum Associates, ISBN 0-8058-4433-3.

30. Robertson, S.E., Hancock-Beaulieu, M.M. (1992) On the evaluation of IR systems. *Information Processing and Management*, 28(4), 457-466.

31. Saracevic, T. (1996) Interactive models in information retrieval (IR). A review and proposal. *Proceedings of the 59th Annual Meeting of the American Society for Information Science (ASIST), 33*, 3-9.

32. Sharp, H., Rogers, Y. and Preece, J. (2007) *Interaction design*, Wiley, ISBN: 978-0-470-01866-8.

33. Shneiderman, B. and Plaisant, C. (2005) Section 5.2: Specification Methods, in *Designing the User Interface*, Addison-Wesley / Pearson Education, p.175-183.

34. Tidwell, J. (2006) *Designing interfaces*, O'Reilly, ISBN 0-596-00803-1.

35. Trætteberg, H. (2003) Dialog modelling with interactors and UML Statecharts - A hybrid approach, *10th International Workshop on Design, Specification and Verification of Interactive Systems*, Madeira.

36. Vakkari, P. (1999) Task complexity, problem structure and information actions, integrated studies on information seeking and retrieval. *Information Processing and Management, 35*, 819-837.

37. Vakkari, P. (2001) Changes in search tactics and relevance judgments when preparing a research proposal: a summary and generalization of a longitudinal study. *Journal of Documentation, 57*(1), 44-60.

38. Voorhees, E. M. and Harman (2005) *TREC – Experiment and Evaluation in Information Retrieval*, MIT Press, Cambridge, MA, ISBN 0-262-22073-3.

39. Winckler, M. and Palanque, P. (2003) StateWebCharts: a Formal Description Technique Dedicated to Navigation Modelling of Web Applications, *10th International Workshop on Design, Specification and Verification of Interactive Systems*, Madeira.

40. Xie, H. (2000) Shifts of interactive intentions and information-seeking strategies in interactive information retrieval. *Journal of the American Society for Information Science, 51*(9), 841-857.