Exploring Interactive Information Retrieval: An Integrated Approach to Interface Design and Interaction Analysis

Gheorghe Muresan
School of Communication, Information and Library Studies
Rutgers University
4 Huntington St., New Brunswick, NJ 08901, USA
muresan@scils.rutgers.edu

Bing Bai
Division of Computer and Information Sciences
Rutgers University
110 Frelinghuysen Road, Piscataway, NJ 08854, USA
bbai@cs.rutgers.edu

Abstract

In this paper, we describe a novel methodology that integrates the design of the (i) user interface; (ii) interaction logger; and (iii) log analyzer. It is based on formalizing, via UML state diagrams, the functionality that is supported by an interactive system, deriving XML schemas for capturing the interactions in activity logs and deriving log parsers that reveal the system states and the state transitions that took place during the interaction. Subsequent analysis of state activities and state transitions captured in the logs can be used to study the user-system interaction or to test some research hypothesis. 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.

Introduction and Motivation

In recent years the focus in Interactive Information Retrieval evaluation has shifted toward 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 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).

We are interested in methodologies for running interactive IR experiments, and especially in the practical aspect of logging 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 explore interaction models, or simply to test the usability of a user interface.

Anecdotal evidence suggests that in such interactive IR user studies, there is usually little or no formal process in designing the log format, the logging system, and the log analyzer, in order for the states of the system and the stages of the interaction to be captured. Logging often consists of print-to-file statements added in an ad-hoc, un-systematic fashion after the implementation of the system has been completed. Therefore, it is hard to insure completeness in capturing the interactions, and capturing the right level of abstraction and granularity may be hard to reach. For example, logging just the queries submitted by the user in a research experiment may be insufficient for answering some research hypothesis: the time taken to type the query, the number of corrections made and other such attributes may be relevant when cognitive processes are investigated. On the other hand, logging "everything", down to tracking the mouse movements, may be a waste of effort and resources. Ideally, the logs should capture the semantic events, deemed relevant for the interaction in the design stage.

State-Based Design of Interaction and Logging

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** that unifies these designs, as depicted in Figure 1. This model will subsequently inform the design of the user interface and will determine the schema for the interaction, the format of the log file, the tasks of the logger, and the functionality of the log analyzer. Our formal approach is based on statecharts (Harel, 1988; Horrocks, 1999) or, in the UML (Unified Modeling Language)[1] terminology, on state diagrams. Such diagrams describe: a finite number of existence conditions, called **states**; the **events** accepted by the system in each state; the **transitions** from one state to another, triggered by an event; the **actions** associated with an event and/or state transition (Douglass, 1999; Fowler, 2004).

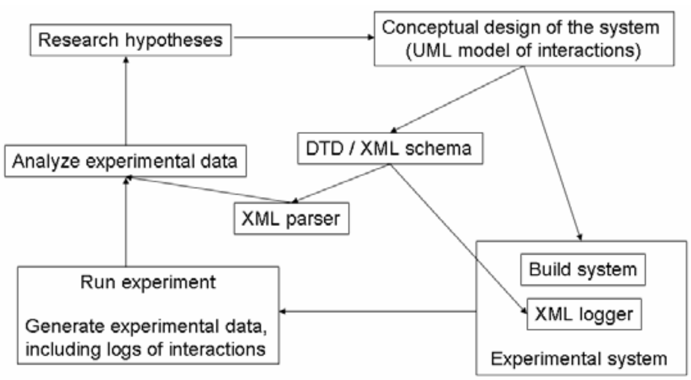


Figure 1: Integrated approach to design, logging and analysis

While UML is well suited for designing the user-system interaction, the Extensible Markup Language (XML)[2] is an excellent choice of format for logging user actions and state transitions: it allows the definition of tags for recording the states of the system, and/or the events that generate state transitions, together with a set of attributes. Moreover, it allows a variety of access modes: sequential access to each event in the log (via SAX[3]); random access to certain kind of events, relevant for a certain research hypothesis (via XPath[4]); and complex visiting patterns (via DOM[5]).

The vocabulary and structure of XML documents valid for a certain application are typically specified via a Document Type Definitions (DTD)[6] or an XML Schema[7] (or simply "schema", to refer to either). A schema 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 or to generate parsers for such XML documents.

Bridging the gap between UML (used for the design of the interaction and that of the log analyzer) and XML (used for logging the interaction) is crucial for our integrated

[1] http://www.uml.org/
[2] http://www.w3.org/XML/
[3] http://www.saxproject.org/
[4] http://www.w3.org/TR/xpath
[5] http://www.w3.org/DOM/
[6] http://www.w3.org/XML/1998/06/xmlspec-report.htm
[7] http://www.w3.org/XML/Schema

methodology. Our solution is to use the Object Management Group's (OMG) XML Metadata Interchange (XMI) specification[8], which specifies a set of mapping rules between UML and XML in terms of elements, attributes and relationships. Carlson (2001) provides a thorough discussion of tradeoffs between a number of mapping decisions.

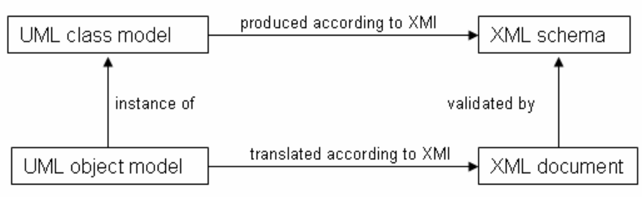


Figure 2: Mapping UML models to XML schemas

Figure 2 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 application of this approach to logging the interaction is 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, and the events or user actions that caused the state transitions.

Case Study: Mediated Information Retrieval

In order to help the reader understand the proposed methodology, we are going to describe the interaction model and one of the experimental user interfaces used in the **MIR** (Mediated Information Retrieval) project (Lee, 2006), as well as the application of our methodology. **MIR** addresses 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 (Muresan and Harper, 2001, 2004).

In the first stage of the interaction the searcher explores a so called *source collection,* a specialized collection of abstracts or documents that cover the her problem domain and emulates a librarian's knowledge of that domain, so that she becomes more familiar with the terminology, concepts and structure of the problem domain, and better able to convey her information need. She can then move to the second stage, in which she searches the Web or any other *target collection* where she hopes to find new information to satisfy her need and complete some task. The tasks in our experiment were related to environmental issues, and the content of the New Jersey Environmental Digital Library was used as the source collection for mediation.

The user interface depicted in Figure 3 has a *Task control* panel where the task is displayed, and where the subject can formulate and submit queries. If a better understanding of the topic is needed, the user can explore the source collection, by submitting queries or by browsing the cluster hierarchy in the NJEDL tab. At any time the searcher can move to the WEB tab and search the target collection; hits are displayed in the *Search results* panel. When a document is selected, it is displayed in the web browser. The subject can use the mouse to save a document from the hit list; the document snippet will be shown in the *Saved documents* panel.

[8] http://www.omg.org/technology/documents/formal/xmi.htm

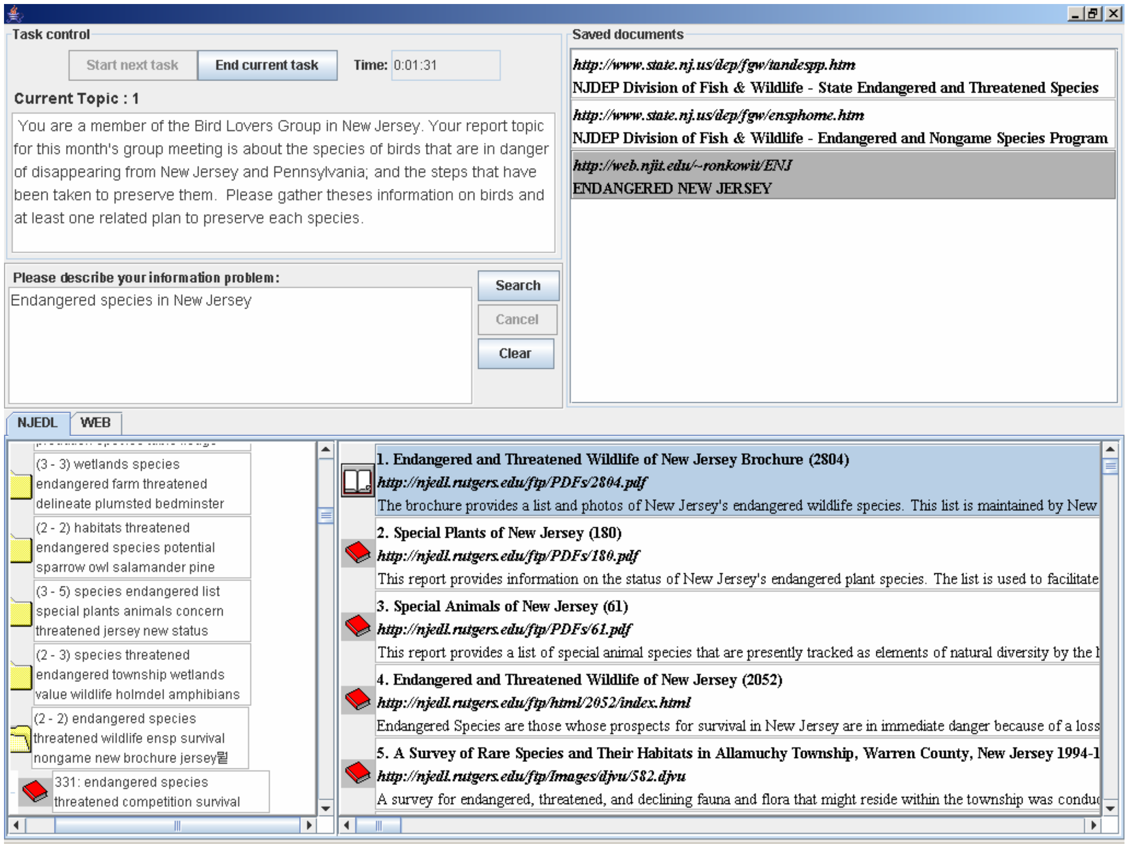
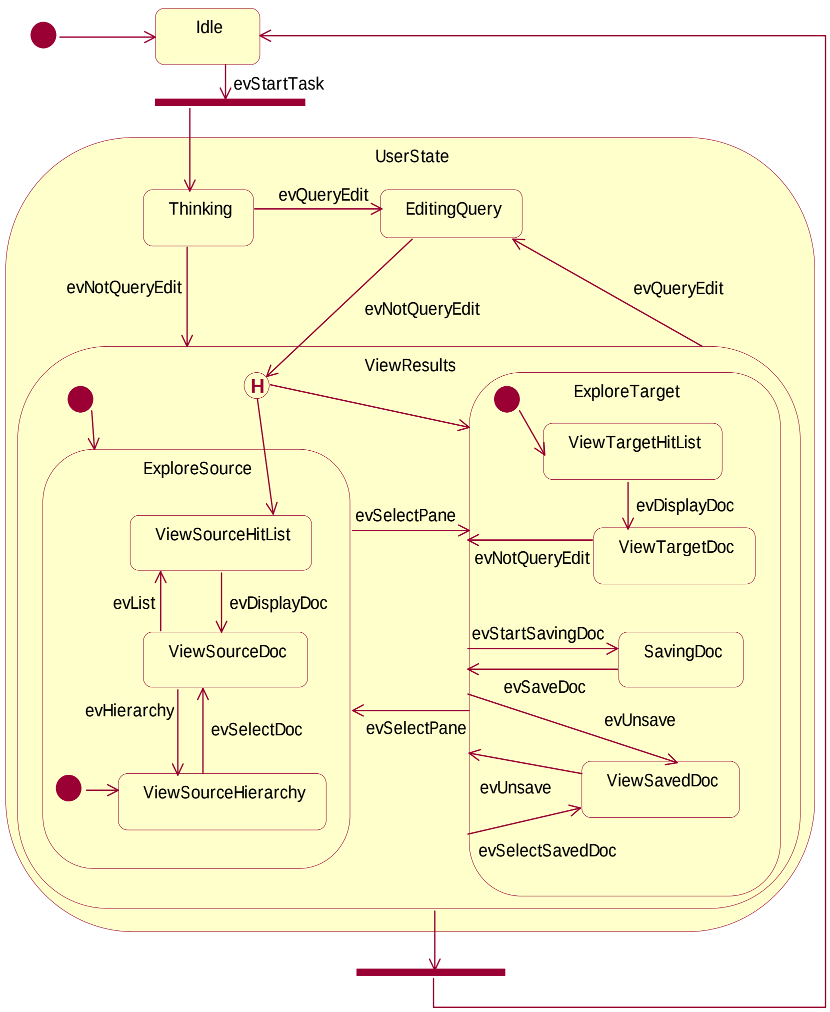


Figure 3: User interface for mediated information retrieval

Figure 4 shows the state diagram that depicts the system states during the MIR interaction. In the Idle state between search sessions, the user may perform related activities such as filling in questionnaires. When the session starts, triggered by an evStartTask event, the system displays the current search task and enters the Thinking state, in which she reads the task description and thinks of appropriate queries or alternative actions. If the user starts typing a query (marked by an evQueryEdit event), there is a transition into the EditQuery state. On the other hand, 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 editing the query (i.e. typing or using copy-and-paste), the system stays in the EditQuery state; when the query is submitted and the search results are displayed in the *Search results* panel, the system enters the ViewResult state. This is a "superstate", which has a number of "substates": the user may choose to explore the source collection (NJEDL), corresponding to the ExploreSource state, or the target collection (the Web), corresponding to the ExploreTarget state.

The granularity of the states depends on the required precision of modeling the interaction. 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 captures 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.



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
…	

Figure 4: State diagram, and a state sequence extracted from logs

The state diagram defines the **interaction model** of the user interface, specifying the valid states of the system, and the valid events and actions taking place during the interaction. The derivation of the state classes for the log analyzer (captured in Figure 5), and of the log schema are relatively straightforward: the state classes correspond to the states of the system, and the elements of the log file's XML schema are derived from the events that trigger state transitions. For simple interaction models, this derivation can be done manually. The automatic approach discussed in section 2, based on XMI, is appropriate for more complex interaction models. The state diagram also suggests the design and implementation of the logging code: whenever a valid event is detected and processed, the task of recording the event and its attributes is delegated to the logger.

Contributions and Future Work

The proposed methodology is a significant contribution to experimental research in Information Seeking and Retrieval, as well as Human Computer Interaction. 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 and can be addressed by studying user logs. This approach has been successfully applied in Interactive TREC work and is being refined in a new project on Mediated Information Retrieval.

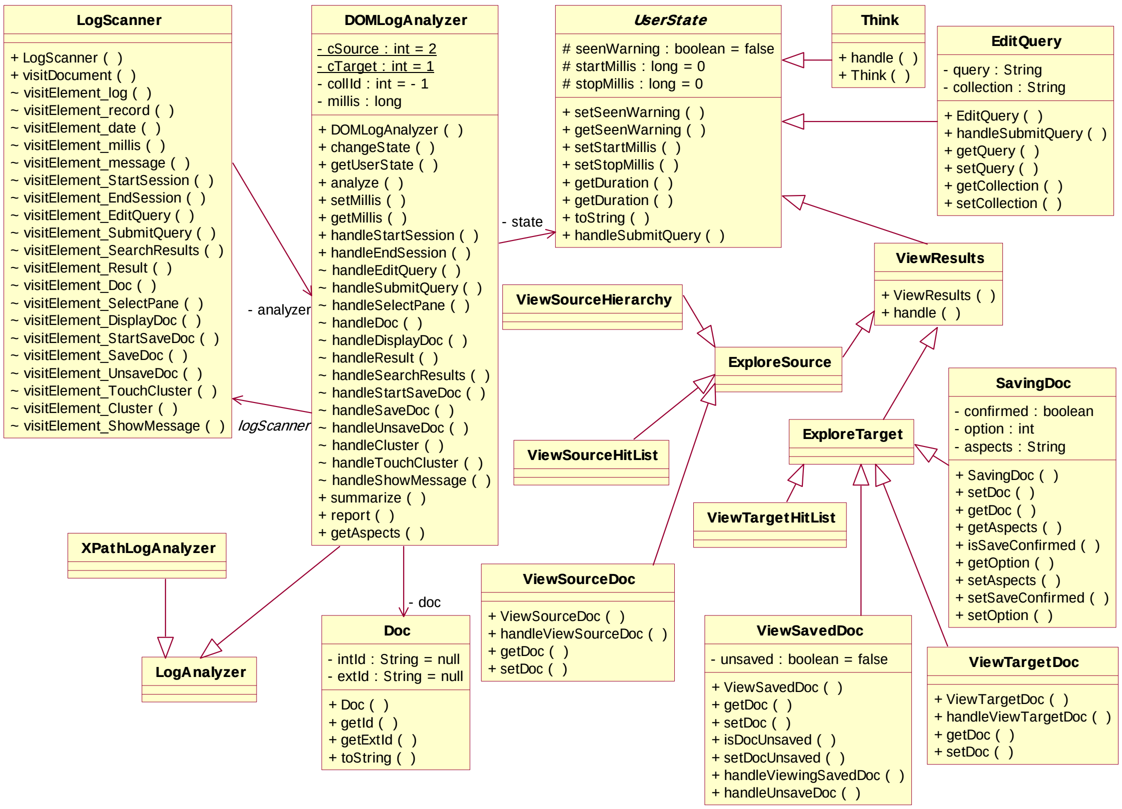


Figure 5: State classes in the log analyzer

The main idea of the methodology is the requirement for a conceptual model for the interaction, based on state diagrams, which subsequently informs the design of the user interface and it determines the interaction log schema, as well as the design of the logging and log parsing software. Moreover, the state diagram provides a mapping between low-level user interface events (keystrokes, mouse moves, etc) and high-level semantic actions (submit query, expand cluster, save document, etc). Therefore, the logs are able to capture the semantics of different events, and the system state that provides context.

We have explored two 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, an XPath-based parser can be used to visit only the XML nodes in the log tree that are of interest (e.g., the list of saved documents can be obtained by specifying "SaveDoc" as the node tags to match).

If the time factor is essential in answering a certain research hypothesis or in getting a certain kind of information, then a DOM-based parser 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 5), 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.

We are currently investigating extensions of this methodology that support the study of user behavior as well as the support afforded by the user interface for various tasks. Firstly, we are looking at analyzing state transitions recorded in the logs for building Hidden Markov Model and predicting future user behavior (Jurafsky, 2000). Secondly, we are building GOMS models and using the sequence of states captured in the logs, together with the durations of the states (Figure 4) in order to study the usability and efficiency of user interfaces (Kieras, 2004).

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.

References

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

Douglass, B. P. (1999) *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns*, Addison-Wesley, Reading, MA, ISBN: 0-201-49837-5.

Fisher, K. E., Erdelez S. and McKechnie, L. (2005) *Theories of Information Behavior*, Information Today, Medford, NJ, ISBN: 1-57387-230-X.

Fowler, Martin (2004) *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd ed, Addison-Wesley/Pearson Education, ISBN: 0-321-19368-7.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995) *Design Patterns – Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, MA, ISBN: 0-201-63361-2.

Harel, D. (1988) On Visual Formalisms, *Communications of the ACM*, 31 (5): 514--530.

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

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

Kieras, D. (2004) GOMS Models for Task Analysis, in Diaper, D. and Stanton, N. (eds.) *The Handbook of Task Analysis for Human-Computer Interaction*, Lawrence Erlbaum Associates, ISBN 0-8058-4433-3.

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.

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 (ECDL)*, Darmstadt, September 2001, 438--449.

Muresan, G. and Harper, D. J. (2004) Topic Modelling for Mediated Access to Very Large Document Collections, *JASIST* 55 (10): 892--910, August 2004.