Process-Aware Knowledge Retrieval

Kurt D. Fenstermacher
Management Information Systems
Eller College of Business and Public Administration
University of Arizona
KurtF@Eller.Arizona.EDU

Abstract

Even with modern information retrieval systems and advanced digital libraries, many people cannot find the information they need when they need it. There are many reasons why information access is difficult: users cannot track all the available information sources, few computer users are skilled in composing queries and “everyday” applications (e.g., word processors and spreadsheets) are not integrated with information access tools. Process-aware retrieval, however, offers a framework that shifts the burden of information access from the user to the computer. By explicitly representing processes, and information about processes, process-aware retrieval enables the computer to make highly targeted suggestions regarding knowledge and information sources, while aiding in the completion of process steps. This paper reviews current strategies for information access, evaluating strengths and weaknesses, and then defines a process-aware framework that builds on the strengths to address the weaknesses.

1. Introduction

If you’ve searched the Web, browsed a set of directories or updated a previously written report, you already know the frustration of accessing the information you need when you need it. Even experts in information access must carefully frame their search queries and identify promising information sources. For knowledge workers, who toil through their computers, the choice is often information feast or famine. The ubiquity of computers and their frequent use in knowledge-intensive tasks is an opportunity for changing the paradigm of information access. Instead of forcing users to monitor the availability of information sources, learn complex query languages and remember which sources and queries were helpful in the past, computers should do the job. If your workstation knew about your activities and the resources you needed, your computer could fetch the information you wanted, perhaps even before you knew you wanted it.

This is the potential of process-aware knowledge retrieval — by representing processes and annotating them with information about knowledge sources, computers could better aid users by proactively suggesting context-sensitive, task-specific knowledge.

Process-aware retrieval offers great promise in a range of applications from high-school students researching term papers to globally distributed workgroups tackling demanding business problems. This paper argues for the promise of process-aware retrieval, and discusses the construction and application of a process-aware framework.

1.1. The Frustration of Information Access: An Example from the Web

Suppose that you’re organizing a birthday party for a friend, and need to make dinner reservations at a local restaurant. You begin by searching restaurant review Web sites for your area, and after reading through a half-dozen reviews, select one of the restaurants — perhaps Crazy Eddie’s Steak and Sushi Bar. The review page does not list Crazy Eddie’s phone number, so you copy the name and paste it into a phone directory search site for your area. You call the listed number, and make the reservation. Next, you’ll need a map showing Crazy Eddie’s location, and driving directions from your workplace. From the phone directory search results, you copy Crazy Eddie’s address, and enter it into an online map service. With the map in hand, you’re ready to announce the party.

Although making a reservation is not difficult, it remains beyond the ken of your computer. Suppose instead of a series of copy-and-paste operations, your computer would recognize when you were making a restaurant reservation and complete the process for you. After entering a restaurant search in a given area, your workstation could suggest alternative review sites with coverage of the same area. After you select a restaurant, your computer could then look up the phone number if needed, dial the restaurant for you, draw a map and print driving directions.

Even the simple task of restaurant reservation illustrates the aforementioned difficulties of accessing information — one must know which restaurant review sites cover the area of interest, how to query such sources and how to integrate information access with
other tasks. Representing even simple processes in the computer, and augmenting existing applications to use these process representations could achieve two important benefits. First, more of the burden of this mundane task would shift from you to the computer. A second benefit of having the computer complete the task is the opportunity for knowledge exchange. If relevant knowledge sources, such as restaurant review sites, were centrally indexed, then new opportunities arise to share knowledge. Sharing centrally indexed resources is not new, but one of the keys in information sharing is knowing when to share. Process approaches answer this question: share when someone else is performing a task that requires this knowledge.

Suppose you wished to make a reservation at an out-of-town restaurant, you might begin with an international restaurant review site, such as the Zagat Survey’s Zagat.com [1]. But many metropolitan areas have local online guides that often include restaurant reviews as well — how would you locate such services? A system that had seen someone else (perhaps a resident of your destination city) use a local review site could propose that site as well, in addition to your more global search. Given the ease of reserving a restaurant table, why must people still perform so much of the task themselves?

Making restaurant reservations might seem a simple task, one that you have probably performed hundreds of times, but the familiarity of the task hides the extensive knowledge needed to do it well. In the above example, you must already know that restaurant review Web sites exist, and know how to find them. You must know how to search the review sites and how to parse the results returned. If you need information missing from a result (e.g., the restaurant’s phone number), you need to know how to use what you do know (the restaurant name, in this case) to complete the missing information (in this example, by entering the name in an online phone directory). For map information, you again need to know that online map services exist, what information they need and how to interpret their results. In the abstract, you must know the process (restaurant reservation), the tasks (restaurant selection, make reservation and generate map/directions) and knowledge sources relevant to each of the tasks and the overall process.

Making reservations at a restaurant is a common task, but clearly not critical to the successful operation of most organizations. However, restaurant reservations are a simple example of a much broader class of computer-based processes that are decomposable into tasks and relevant knowledge sources. For example, a student preparing a term paper must research potential topics, identify sources, review each source, write a rough draft and revise the final paper. A salesperson on a first-time sales call must research the customer’s needs, introduce her organization and identify products and services of interest to the new prospect.

In all of these cases, what’s needed is a contextual representation that the computer can use to aid users. What is the context in which information is accessed? Potentially, the context extends beyond the desktop to the larger organization and even to the world at large. There is clearly no way to encode such wide-ranging contextual information in a computer system. Instead, the question becomes, how little information can be encoded to derive the benefits of context-aware retrieval, without a burdensome knowledge representation task? This paper will argue for the process proxy principle: information about the process and activity a user is performing offers the contextual information needed to more accurately retrieve relevant knowledge, as it is needed by the user.

Process-aware retrieval is a framework that associates relevant knowledge sources with user activities (as shown in Figure 1), enabling process-aware system to make context-sensitive, task-specific suggestions to users. To support process-aware retrieval requires an architecture that builds on existing process representations (e.g., [2]) by including knowledge source annotations, as well as hooks for process recognition and recording.

This paper proceeds by discussing other attempts to address the broader problem of delivering the right information to the right people at the right time — what might be called just-in-time information, by analogy to just-in-time manufacturing [3]. Several fields have addressed the problems of information/knowledge delivery, and although each has made important contribution, challenges remain. After reviewing the work of related fields, the importance of process is addressed, and a process-aware architecture is described.

2. Managing Data, Information and Knowledge

Despite advances in artificial intelligence, computers are still poor thinkers; people continue to make the complex judgments that are commonplace in the knowledge economy. To make such decisions well, however, people need data, information and knowledge. And there’s the rub — as knowledge workers create and store more information in computers, it has become harder to extract just the needed information, just as it’s needed. However, knowledge-intensive processes are often complex undertakings, and require a broad range of skills. Thus, the proposed process-based architecture is a hybrid of work in several fields, although it draws primarily on
work in information retrieval, workflow and work on planning [4] from artificial intelligence.

2.1. Information retrieval

Information retrieval research [5] has primarily focused on the question: given a corpus of text documents, how can one build a system to provide fast access to relevant documents? Much of information retrieval research has revolved around implementations of the vector space model [6], which represents documents as high-dimensional vectors, enabling systems to be built based on linear algebraic operations. The vector space model provides quick retrieval and provides for many complex query forms — including conjunctive, disjunctive, proximity and negative queries. For example, most Web search engines rely on the vector space model to support queries over very large collections (over 1,610,476,000 indexed pages claimed by Google [7]).

Information retrieval (IR) techniques will often work when other techniques will not, because they are predicated on very few assumptions about the indexed documents. To apply IR techniques, the system assumes only a parser that will read the documents and output tokens, which will then be indexed. Wide applicability is a double-edged sword, however, as the simplicity of indexing requires more sophisticated retrieval. In particular, to use most IR systems effectively, users most couch their requests in complex, multi-word queries often incorporating lengthy Boolean expressions.

In past decades, when access to online information sources was through reference librarians and other information specialists, complex query formulation was not a significant barrier. However, with today’s widespread access to online information, few people have the skills needed for effective access. In short, information retrieval systems are a critical component of information access, but new interfaces are needed to shift the burden from the user back to the information system. Short queries are the norm today [8], which in turn lead to ambiguous queries. Is “bank” a financial institution or the side of a river? Without additional search terms — either bank AND finance AND deposits or bank AND river — IR systems cannot distinguish between the two interpretations. The term effective query will be used to distinguish between ambiguous queries and those that can identify relevant documents, either due to the number of terms or their combination.

The parsimonious assumptions of information retrieval ensure broad applicability (at the cost of complex queries), but they make it difficult to incorporate additional information when it is available. Information about document structure can support fielded search, which enables users to link query terms to specific record fields, improving the accuracy of search results. The use of fielded search by Web search engines, which parse primarily HTML documents, is a common example of this. However, most IR systems do not provide the incorporation of rich meta-information at indexing or retrieval time. In this research then, the challenge is to use the power of information retrieval to support quick access to information, while easing the burden of posing effective queries and enabling the incorporation of rich meta-information.

2.2. Workflow management systems

A workflow management system “completely defines, manages and executes ‘workflows’ through the execution of software whose order of execution is driven by a computer representation of the workflow logic” [9]. In this research, the emphasis on process and its explicit representation is shared with the field of workflow systems. Traditionally, the emphasis has been on control and coordination [10] of tasks, rather than leveraging processes as context indicators to support better information access. Workflow systems are an excellent base for process-oriented knowledge retrieval because they are well studied and increasingly common in organizations.

Although there have been some attempts to use process representation as an aid to knowledge retrieval (e.g., in the Kabiria system [11,12]), few workflow systems have attempted to reuse the process information to support knowledge indexing and retrieval. In an overview of document management systems, [13] argue for process representations as the key to regulating the flow of information to users — both to ensure that they are not overloaded, nor that they limited by their own ignorance of relevant sources. They argue for the use of process representation in building intelligent information agents, focusing on opportunities for proactive retrieval, but without describing the representation or its implementation.

2.3. Previous process-oriented work

The idea that process can be a proxy for context, and hence a lever for information access, has been explored in other work. This process proxy principle has several implications; different researchers have emphasized three principles in designing process-aware systems: context-aware, task-specific and proactive assistance.

In the information access literature, those who build process-aware systems often label them context-aware (or context-specific) to denote that information access
depends on what the user is doing and why. Such process representations range from simply the words that a user is currently reading or writing [14] to complex knowledge representations drawn from artificial intelligence research [15,16]. Generally, the label context-aware describes query formulation, and focuses on the use of the process context to generate effective queries.

Although incorporating a notion of context is an important step, users often participate in many contexts simultaneously. Thus, in addition to generating context-specific queries, information systems must respect that users also want information when it is appropriate. In process-aware systems, information is deemed appropriate when it bears on the task at hand for the user. For example, as a Formula 1 racing fan, I would appreciate context-sensitive searches that exclude results from NASCAR, CART and the many other racing leagues operating worldwide. But I also do not wish to be interrupted while writing this paper to be notified that the BMW/Williams racing team and Michelin have settled on the rubber compound to be used for the racecars’ tires during Sunday’s Canadian Grand Prix.

With the proliferation of online information systems, users are confronted with two new problems. First, they must know what information sources exist. Second, users must know what content is available. Past research has explored information filtering, also known as current awareness or selective dissemination of information (SDI) systems [17]. Filtering systems select documents of likely interest from a stream of documents. However, there has been little work done on identifying promising streams themselves. Highly targeted search engines that cover very narrow topics might be appropriate for users, but they must know that they exist, what content they index and how to query them. By associating knowledge sources with tasks, process-aware systems can offer a hybrid push/pull system by recommending sources and queries.

2.4. Process work in artificial intelligence

Abecker, Bernardi et al. [15] describe a more developed approach that supports the three process-aware principles discussed above: context-aware queries, task-specific information and proactive delivery. In particular, they build representations for what they term knowledge-intensive tasks, or KITs. KITs are workflow activities annotated with information about knowledge to begin the activity, as well as processing rules and where task-generated knowledge would be useful. Although there are many similarities to the approach described in this paper, building the KIT representations is a substantial knowledge engineering challenge. Moreover, the authors do not discuss techniques for learning KIT annotations from user behavior — an important source of meta-information. Staab & Schnurr [18] present a very similar approach to Abecker, Bernardi et al. [15], but, as discussed by Abecker, Staab and Schnurr emphasize ontology-based retrieval.

Although workflow research is an obvious home for process-based research, artificial intelligence researchers have long studied planning [4] and such research has connections to the current work. Planning research is largely driven by work on autonomous agents and tries to answer the question, given my goals and the current situation, what should I do next? Many of the ideas present in the workflow literature are seen in the planning literature as well. Planning researchers have developed semantically rich process representations that planning systems can reason about. These computer-reasonable (not simply computer-readable) representations enable planners to operate dynamically, assembling plans to meet changing goal criteria and adapting to a changing world state. Although some work has been done at the intersection of workflow and artificial intelligence [19,20], more investigation is needed. In related work [21], planning techniques are employed to enable dynamic process construction for a workflow management system. For process-aware retrieval, the key lessons from planning are guidelines for developing representations and the potential of generating processes dynamically at runtime.

3. A process-aware architecture

A process-aware retrieval framework requires several components: process representation, knowledge source associations and a platform-specific mechanism for integrating process-aware retrieval with existing tools. The integration mechanism would then read in process descriptions and knowledge sources associations, and then monitor user actions. Based on a combination of the information sources and process descriptions, the monitoring mechanism would attempt to identify instances of known processes. Once a process is recognized, the system can offer to assist the user. The assistance could take two forms: recommending relevant information sources (and perhaps aiding in query construction) and facilitate transitions from one task to the next. Thus, a process-aware architecture will require the following functionalities:

- Process library — A source of process information, preferably in a shared location.
- Knowledge source library — Annotations of process activities that incorporate sources and perhaps query assistance.
Monitoring framework — Observing and recording user actions.

Automation — A proactive framework must be able to take actions on behalf of the user.

Process recognition — Mapping the user actions reported by the monitoring framework to specific processes.

The five primary functions listed above will enable the system to observe user actions, recognize opportunities to offer assistance, aid in transitions and search queries and proactively submit carries or complete tasks. Each of these five functionalities is described in turn in the remainder of this section.

3.1. Process library

The process library is the heart of any process-driven system. Certainly the library must support common operations such as insertion, deletion and modification of existing processes. Because much of the value in process-driven approaches is in the sharing of the process and knowledge sources among many users, the process library is an obvious candidate for network access. In the initial prototype, the process library is stored locally. A natural next step is to adopt a client-server model using HTTP. However, the long-term vision is to adopt a distributed object model, which would enable each user to act as both server and client — both a producer of new knowledge-enabled processes and a consumer of others’ processes.

Although the networking issues are an important aspect of the implementation, the more challenging task is building a suitable process representation. The process representation must support conflicting goals: it must be detailed enough that the computer can map from actions to the process representation to support recognition and yet broadly enough defined a single process description will cover a range of instances.

For the restaurant reservation, the following activities are encoded:
1. Select restaurant.
2. Retrieve restaurant phone number.
3. Retrieve restaurant address.
4. Call restaurant at phone number to make reservation.
   a. If restaurant accepts, note time and date
   b. If restaurant declines, return to step 1
5. Retrieve map of restaurant area.
6. Retrieve directions to restaurant location.
7. Notify other attendees of restaurant, time and date, including map and directions.

For the reservation process, the representation must capture these steps, but more is needed. (A portion of the restaurant reservation process is shown in Listing 1.) Each step must be characterized by the data required and the data it produces. For example, in step 2, retrieving the phone number, the representation must incorporate several facts. First, before a phone number can be retrieved the business name and the location (at the city level) must be known. Second, the primary action (phone number lookup) must be characterized to enable the system to associate information sources appropriate for the task.

The process representation also includes precondition and result attributes. By describing what information is needed to perform each step, as well as the result of performing a step, the process representation can be read by a problem solving or theorem proving system that supports means-end reasoning. With such support, an intelligent reasoner can “backtrack” by matching a desired result against an indexed result, and then continuing back to match the preconditions of the ultimate step to the result of a prior step. In the prototype implementation, only the explicit task ordering given in the files is used, but extending the system with support for logic operations is an area of future research.

3.2. Knowledge source library

Monitoring user actions and matching them against stored process profiles supplies the task context needed to make helpful suggestions for information access, but it leaves two questions unanswered. Which information sources are useful? How are these sources accessed? The knowledge source library

```xml
<Process ProcessID="0001" Name="Restaurant reservation" Owner="kurtf" Description="Find a suitable restaurant, lookup phone number, make a reservation and create a map with driving directions." RevisionDate="11-Aug-2001">
  <TaskGroup Ordering="SequentialLoose">
    <Task>
      <PreCondition>InternetAccess</PreCondition>
      <PreCondition>LocationKnown</PreCondition>
      <Action>Search restaurant reviews</Action>
      <Result>RestaurantSelected</Result>
    </Task>
    <Task>
      <PreCondition>RestaurantSelected</PreCondition>
      <Action>LookupPhoneNumber</Action>
      <Result>KnowRestaurantAddress</Result>
      <Result>KnowRestaurantPhoneNum</Result>
    </Task>
  </TaskGroup>
</Process>
```

Listing 1 — Portion of restaurant reservation process
answers these two questions.

In the prototype implementation, step 2 (retrieve phone number) is labeled \texttt{LookupPhoneNumber}; in the knowledge source library, Qwest’s online phone directory (qwestdex.com) is listed as a source for the \texttt{LookupPhoneNumber} action. Because the \texttt{LookupPhoneNumber} action has standard input and output, the process representation need only state how to map the process-specific information to the standard input and output. In the prototype implementation, the input-output mapping is hard-coded. In future, we will extend the knowledge source representation to include input requirements (city, state and business category or business name or person for QwestDex). For relatively fixed sources (such as directory services, mapping sites and so on) optional output adapters will be added. The adapters will transform the source output into a useable form — translating a QwestDex Web page into a phone number object, for example.

One solution to the source problem is to maintain a central catalog of useful sources. Of course, this does not require a process-aware system — libraries have successfully tracked new information sources for centuries. The challenge is in supporting context-specific retrieval of information sources. Process-aware retrieval answers this question by offering a mapping between user actions and relevant knowledge sources. In the opening restaurant example, visiting any restaurant review site and entering a locale suggests that other review sites that cover the same area would be relevant. With a process-aware system, information sources relevant to follow-on steps are deemed valuable as well. Moreover, the ability to pass needed information from one step to the next (e.g., the name of the restaurant from the review site to the phone directory) enhances sharing of knowledge sources.

In process-aware retrieval, unlike workflow systems, processes are the means rather than the end. In the phone directory example, knowing whether to search for a phone number at QwestDex or British Telecom depends on whether the user is looking in Hawaii or London for a restaurant — an aspect of the process instance, not the process description. Through reification, a designer could simply create different processes — one for United States restaurant reservations, and one for United Kingdom reservations. Creating highly targeted processes scales poorly, however. The rapid pace of information source creation and destruction argues against a strategy that requires the revision of process definitions with each change.

If instead semantic information were available about each source, then information sources could be associated with activities on the fly. Rather than creating complex conditional expressions to handle myriad cases (or creating multiple static processes), process-level goals would be used to construct processes. For example, a user who is interested in selecting a restaurant in a U.S. location would need an American phone directory. A process-aware system that had labeled QwestDex as just such a directory could then assemble a process that fed the results of a restaurant selection to QwestDex. In short, the conditional execution that workflow systems embed in the process description itself would be abstracted away from the process description to a meta-process level.

Work in artificial intelligence planning has always treated actions in this way, beginning with early problem solvers [22]. Moreover, planning researchers have studied abstraction as a thing in itself, beginning with Sacerdoti’s Network of Abstraction Hierarchies (NOAH) system [23]. Later systems, such as HEARSAY II [24] and the model proposed by Hayes-Roth and Hayes-Roth [25], have supported reasoning at multiple levels of abstraction simultaneously. Adapting these techniques to workflow applications is necessary for processing knowledge flows, and could be helpful for other kinds of workflow as well.

3.3. Monitoring framework

For a process-aware retrieval system to be proactive, the system must monitor user actions and attempt to infer process instances from user behavior. Modular system design suggests that these two functions be split: one system monitors user actions and a second identifies instances of known processes. While process recognition is the focus of the next section, this section describes first the requirements for a monitoring framework, and second, the prototype implementation of the framework.

A monitoring framework must serve two constituencies simultaneously: end users and developers. For end users, the framework must support low-overhead, unobtrusive monitoring of user actions. If the monitoring itself is a burden on the user’s computer, or requires frequent user input, most users will work to circumvent the system and defeat the advantages of process-based retrieval. For developers, the framework must enable them to define what actions

```xml
<KnowSource Name="QwestDex" Type="DWeb">
    <URI>http://www.qwestdex.com/</URI>
    <Originator>KDF</Originator>
    <Feedback>
        <Person/>  
        <Comment/>
    </Feedback>
    <AppliesTo>LookupPhoneNumber</AppliesTo>
</KnowSource>
```

Listing 2 — QwestDex knowledge source description
should be noted and what action to take when the actions are observed.

Ideally, a common monitoring framework could be developed for use on many different platforms. Unfortunately, the tight integration required between a user-monitoring framework and user applications precludes a single framework. The widespread adoption of Microsoft Windows and Microsoft’s application suite, Office, however, means that a Windows solution addresses the needs of many framework users.

In addition to substantial market share, Microsoft Windows also offers the technology infrastructure needed to support a monitoring framework in its Component Object Model (COM); COM is a binary reuse framework that supports communication among COM-compliant applications [26]. Most Windows applications expose some, if not all, of their functionality through COM interfaces. Through COM, most applications offer three categories of interaction: properties, methods and events. To monitor applications, a developer can define custom event handlers for the relevant events exposed through a program’s COM interface. Windows will then invoke the appropriate handlers when the event fires in the monitored application.

The current prototype implementation for restaurant reservations is entirely Web-based, and relies solely on Internet Explorer. However, the techniques and code are similar for other applications — particularly Microsoft Office applications. For reservations, the monitoring is focused on the URL of viewed pages, since they serve as keys to index in to the process library. Listing 3 shows the code (in the Python programming language [27]) used to capture the NavigateComplete2 event from Internet Explorer, which fires each time a page is loaded into the browser. The event is passed the URL of the loaded page, which is then checked against a list of URLs known to be used in the restaurant reservation process.

```python
from win32com.client import DispatchWithEvents
class IEEvents:
    def OnNavigateComplete2(self, pDisp, URL):
        kd.getActionFromUri(URL):
            # Create an instance of Internet Explorer, with
            # events handled by the class IEEvents
        ie = DispatchWithEvents("InternetExplorer.Application", IEEvents)
        # Load the page at the given address
        ie.Navigate2("http://www.eller.arizona.edu/")
        ie.Visible = 1 # Shows the browser
```

Listing 3 — Handling COM events for Internet Explorer in Python

### 3.4. Process recognition

A process-aware knowledge retrieval engine requires extensive coordination among user applications and the engine itself. The retrieval engine must monitor user actions within applications, looking for both retrieval and indexing opportunities. The system must infer from user actions that the user is engaged in a process, as well as the task being performed. Recognition could be as simple as a user’s selection of a well-known process from a menu of processes. Alternatively, it might be as complex as inferring the user’s goals and belief from seemingly unrelated actions. The latter, known as plan recognition in the artificial intelligence community [28], is known to be computationally intractable in the general case.

To address these questions, three levels of process support will be defined and discussed: pipelining, workflow management and process-oriented knowledge management. Pipelining is the simplest form of process orientation, in which activities’ input-output mappings are specified and a total ordering of activities is given. Workflow management (with its usual definition) is an extension of pipelining that supports richer process representations and rule-based actions. Process-oriented knowledge management augments workflow’s process representation with knowledge. Each of these approaches is now discussed in further detail.

Process-aware knowledge retrieval builds on workflow management systems to support knowledge management more broadly. The key aspects of a process-aware retrieval are 1) information source suggestions keyed to user activity, and 2) context-sensitive information access. To provide these two services, process-aware systems must recognize when a user is engaged in a known process, annotate process activities with relevant knowledge and build and maintain libraries of annotated processes.

### 3.5. Automation

In addition to recognizing events, a process-driven system must also be able to control other applications. For example, a user who is responding to an emailed request for information might complete a Web search and then email the relevant pages to the questioner, along with a comment in the body of the email. If a process-driven system recognized the mail format as a question form, the engine might extract the query
and submit it to a relevant, Web-based, knowledge source.

Microsoft’s COM specification supports automation, in addition to the event handling mentioned previously. Listing 3 shows automation in action, using Internet Explorer’s COM methods. The next to last line of the listing shows the Internet Explorer instance being directed to the Eller College’s Web site. Note that until the listing’s last line (i.e. Visible = 1), the Internet Explorer instance is not shown to the user, and does not appear among the running programs in Windows task list or taskbar. Controlling visibility enables programmers to use the functions of applications “behind the scenes”. For example, a process-based approach to mortgage shopping could invoke an instance of Microsoft Excel to perform several financial calculations, and then display the result in the application’s own window.

4. Future research

Currently, the process-aware framework is being investigated as a tool for building a virtual public library. In a typical public library application, a citizen interested in learning about the candidates in an upcoming local election might be interested in information about their records with regard to important local issues. If the voter were to simply enter elections into a search engine she would retrieve information about candidates, the election process and democratic principles. A first-time voter might want information about the process of election itself, to better prepare for the casting his vote.

In a university setting, a student researching a paper would likely begin with a broad search to identify a suitable topic. At this stage in the process, a detailed article with a narrow focus would not be appropriate. Thus, an undergraduate student would likely be better served by searching newspapers, magazines and even the general interest publications of professional societies, such as the Communications of the ACM or Archaeology, published by the Archæological Institute of America. With a process-aware framework, voters and students could receive highly targeted assistance.

Computer-based collaboration is another opportunity for additional research. Using a task, role pair to characterize users offers an opportunity to share knowledge sources transparently. Moreover, the process framework can link together the actions of distributed teams. Being able to reason about the processes, and identify gaps in teamwork or areas that require additional emphasis is another possible application.

In addition to many potential applications, several important research questions are still outstanding. The verification of the process proxy principle is one of the key questions, as much of process-aware retrieval depends on it. Using process information to support information access is different from previous uses of process, and thus will likely need different — or at least augmented — process representations. Closely related to process representation is the question of associating information sources with processes. Source association is doubly important in the framework described because it serves as the key to process recognition, as well as information in its own right. As discussed in the review of related work, research in artificial intelligence planning has grappled with many of these same questions in an intelligent agent context. Incorporating planning principles into workflow process representation enables new research into reasoning about processes, and offering a new level of dynamism in workflow systems. Case-based reasoning offers particular promise in the construction and use of intelligent process libraries.

5. Conclusions

Today’s information systems offer excellent performance in skilled hands, but are difficult to navigate for most users. Constructing complex queries, tracking myriad information sources and creating a thread that runs across multiple applications are demanding tasks that should shift from users to their computers. The lack of context in user queries is one of the key hurdles to better information access. Unfortunately, representing the complete context of every user action and goal scales poorly. By encoding process information and observing user interaction with existing application, user processes can serve as proxies for the more complete context. In many environments, the tools exist to enable users to reap the benefits of process-aware retrieval while continuing to use familiar applications.

Process-aware retrieval offers a framework for indexing and retrieving task-specific knowledge. Building a process-aware retrieval system is a challenging integration task, however, that requires a broad system. Drawing on representation work from artificial intelligence planning, process research from workflow and access tools and strategies from information retrieval, process-aware systems can shift the burden of access from the user to the system — where it belongs.

6. References


