Reducing Friction for Knowledge Workers with Task Context

Mik Kersten, Gail C. Murphy

■ *Knowledge workers perform work on many tasks per day and often switch between tasks. When performing work on a task, a knowledge worker must typically search, navigate, and dig through file systems, documents, and emails, all of which introduce friction into the flow of work. This friction can be reduced, and productivity improved, by capturing and modeling the context of a knowledge worker's task based on how the knowledge worker interacts with an information space. Captured task contexts can be used to facilitate switching between tasks, to focus a user interface on just the information needed by a task, and to recommend potentially other useful information. We report on the use of task contexts and the effect of context on productivity for a particular kind of knowledge worker, software developers. We also report on qualitative findings of the use of task contexts by a more general population of knowledge workers.*

Knowledge workers perform work on many different tasks per day, often spending only minutes on any particular task before switching to another (Gonzales and Mark 2004). There is a concrete cost to such task switches; with each switch, workers waste time repeatedly identifying information, such as parts of web pages, documents, and emails, relevant to their current task. The difficulty and cost of making task switches is exacerbated because the knowledge worker must locate the relevant information amongst the vast information space available to him or her through file systems, the web, and other tools.

To recall and identify the information relevant to a particular task, a knowledge worker must currently rely on his or her semantic memory (Snowden 1996). Semantic memory requires multiple exposures to each referent, with each exposure updating the memory. Semantic memory is well suited to working with small information spaces where a knowledge worker can become familiar with a substantial portion of the information space in a reasonable amount of time. However, when working with a large information space, the limitations of semantic memory become apparent. Knowledge workers must spend an inordinate amount of time reminding themselves of the bits and pieces of an information space to complete a task.

To help a knowledge worker perform work on larger information spaces, we have been investigating tools that leverage the knowledge worker's episodic memory, in which only one exposure is required to remember an event. Episodic memory can be seen as a map that ties together semantic memories (Snowden 1996). A common form of episode for knowledge workers are the tasks that a knowledge worker performs, where a task is defined as "a usually assigned piece of work often to be finished within a certain time."[1] Instead of forcing knowledge workers to find and recall information related to a task, we provide a facility to capture and model the information related to a task as it is performed and then provide a facility to recall tasks easily along with with their associated information when a task is revisited. We use the term *task context* to refer to the subset of information relevant to a task. We follow the semantic memory concept of reinforcement (Plotnik 2004) by automatically weighting the pieces of information associated with a task according to how frequently and recently the information is accessed as part of the task. The only burden we impose on a knowledge worker is the need for him or her to indicate the episodes by defining and activating the tasks on which he or she works.

We show how we can use task contexts to reduce the friction and improve the flow of a knowledge worker's work through such operations as focusing the user interface on the information relevant to a task and allowing the exchange of task contexts between knowledge workers. We report on the effect on productivity of introducing task contexts to one particular kind of knowledge worker, software developers. Hundreds of thousands of software developers have access to task contexts daily through the open source Eclipse Mylyn project[2] that we created and continue to evolve. We also report on qualitative data that we have gathered about the use of task contexts for more general knowledge workers.

This article draws on material published earlier about the definition of task contexts and the use of task context to improve productivity for software developers (Kersten and Murphy 2006), adding the application of task contexts to knowledge work, previously presented at a workshop (Kersten and Murphy 2012).

We begin the article with a review of related efforts and a description of our use of the term *task context*. We then describe how we have made use of task contexts in task-focused user interfaces and review the use of those interfaces to improve productivity across three field studies: one of programmers, one of knowledge workers for a short period of time, and one of knowledge workers over several years. We conclude with directions for future research in this area.

Task Management Systems

One way to manage information associated with tasks is to allow a user to have virtual desktops, as is common in modern operating systems. A virtual desktop allows applications to be clustered on separate virtual screens. For some users, each virtual desktop corresponds to a separate coarse-grained task, such as reading email. Henderson and Card's original Rooms system (Card and Henderson 1987) that introduced virtual desktops considered support for quick and semantic access between desktops and for multiple applications with different information to be displayed in different desktops. These characteristics are not found in current operating system implementations.

Other systems have considered a task-centered approach to the management of documents, such as the Placeless and Presto projects that supported a user-driven flexible, but manual, configuration of documents similar to the tagging functionality available in the user interfaces of modern email systems (Dourish et al. 1999). In these systems and in virtual desktops, the structure of tasks must be explicitly defined. More recent work has considered the automation of task definition and contexts; for example, systems have been developed that extract the definitions and contents of tasks from relevant information sources, such as email (Belotti et al. 2003).

The idea that the information associated with a task could be automatically determined based on interaction was introduced by the user-monitoring environment for activities (UMEA) system (Kaptelinin 2003) and expanded upon by the TaskTracer system (Dragunov et al. 2005). Each of these systems monitor a knowledge worker's interactions with information and create a listing of the information interacted with as part of each task, in the case of TaskTracer, and as part of each project, in the case of UMEA. In these systems, the context of the task consists of separate pieces of information with no model of how the information is related. In these systems, the context also increases with the amount of interaction unless a user takes explicit action to delete information from the task, causing the potential for a task context to overwhelm a knowledge worker and reduce the benefits of providing the context.

The context-aware activity display (CAAD) system attempts to provide more automation by inferring both task boundaries and the information associated with a task based on the patterns in which information is accessed together by a knowledge worker (Rattenbury and Canny 2007). Data from a short-term 10-person study of the use of CAAD suggests that while the discovered artifact groupings provide utility for users, the most common edit event applied by a user to a grouping was still to delete an artifact. The need for a knowledge worker to edit the task context introduces friction into the worker's work flow that the context is intended to alleviate.

Event kind	Interaction	Description
selection	direct	Selections through mouse or keyboard
edit	direct	Textual and graphic edits
command	direct	Operations such as saving
propagation	indirect	Propagate to structurally related elements
prediction	indirect	Potential future interactions

Table 1. Kinds of Interaction Events.

To address the automatic trimming of task contexts, a refinement of TaskTracer used machine-learning techniques to predict which web pages accessed as part of a task were likely to be revisited (Lettkeman et al. 2006). While they were able to achieve almost 80 percent accuracy in predicting what web pages a user would want to keep in his or her browsing history, they did find a substantial difference in results between users. This result reinforces a need to have models that can react to the needs of knowledge workers.

This previous work demonstrates the possibilities for helping smooth the activity of knowledge workers through tool support that segments a worker's activity into tasks. This existing work is limited in its treatment of the context of the task, particularly in the lack of modeling the ebb and flow of the importance of particular information artifacts to a task at hand.

Task Context

Recent advances in ubiquitous computing have showcased the need for addressing context explicitly in computing systems as it can be critical for a computing system to know when and to where a computation has moved. The rise of ubiquitous computing has lead to more and broader ideas of context. Our use of contexts for tasks is narrower. We consider context as a representational problem (Dourish 2004), focusing on the information that must be consulted, created, or changed to complete work on the task. The information we consider to form a task context consists of pieces of unique identifiable digital content and the relationships between those pieces. Following in the tradition of UMEA, we determine the task context from the interaction of the knowledge worker with the information. Activity or work performed on the task is thus simultaneously helping to form the context for the task while also causing the work to occur.

A task context consists of a graph of the parts of digital artifacts (elements) and relations between the artifacts. Each element and relation in the graph is given a weighting that defines its relevance, or degree-of-interest (DOI), to the task. The contents and weightings in the graph are defined entirely by an interaction history, which is comprised of a sequence of interaction events that a knowledge worker has with the information in their space (that is, the artifacts) and the indirect interactions that a tool can have on behalf of the knowledge worker for the task of interest. Direct interactions include edits to artifacts and selections of artifacts; indirect interactions include predictions and propagations that cause elements to be added to the interaction history that have not yet been interacted with directly by the knowledge worker. Table 1 summarizes the kinds of interaction events.

Figure 1 shows an example of task context formation where the knowledge worker is a programmer working on source code artifacts. As the programmer works with the source code, interaction history is captured and is used to produce a task context graph. The nodes and edges in this graph reference concrete elements and relations, in this case different kinds of Java source code declarations (M = method, C = class, I = interface). As the interaction history is captured, it is processed and a DOI function assigns a real number weighting to each element and relation, corresponding to the frequency of access to the element or relation, less a decay factor that corresponds to the total number of interaction events processed as part of that task context. Accessing an element increases its weight, while accessing other elements decays the weight of infrequently accessed elements. Value ranges on the DOI (the y-axis on figure 1) specify which elements are interesting and uninteresting. Interesting elements or relationships are those with a positive DOI value. Uninteresting elements or relationships are those with a negative or zero DOI value, which occurs either through decay or because the element or relation has never been the target of an interaction. For a knowledge worker using Word documents and the web, our system will introduce interaction events on the document or parts of the web as

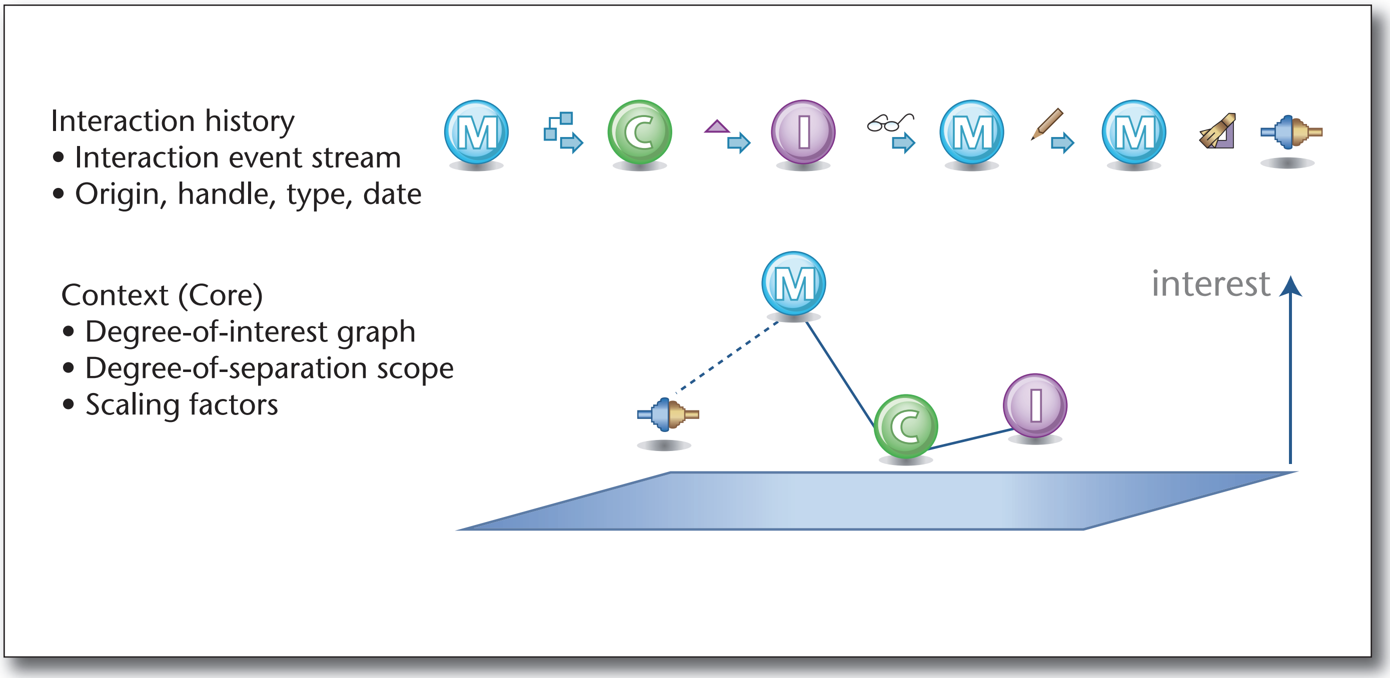

Interaction history
• Interaction event stream
• Origin, handle, type, date
Context (Core)
• Degree-of-interest graph
• Degree-of-separation scope
• Scaling factors
interest

Figure 1. Interaction History to Context with DOI.

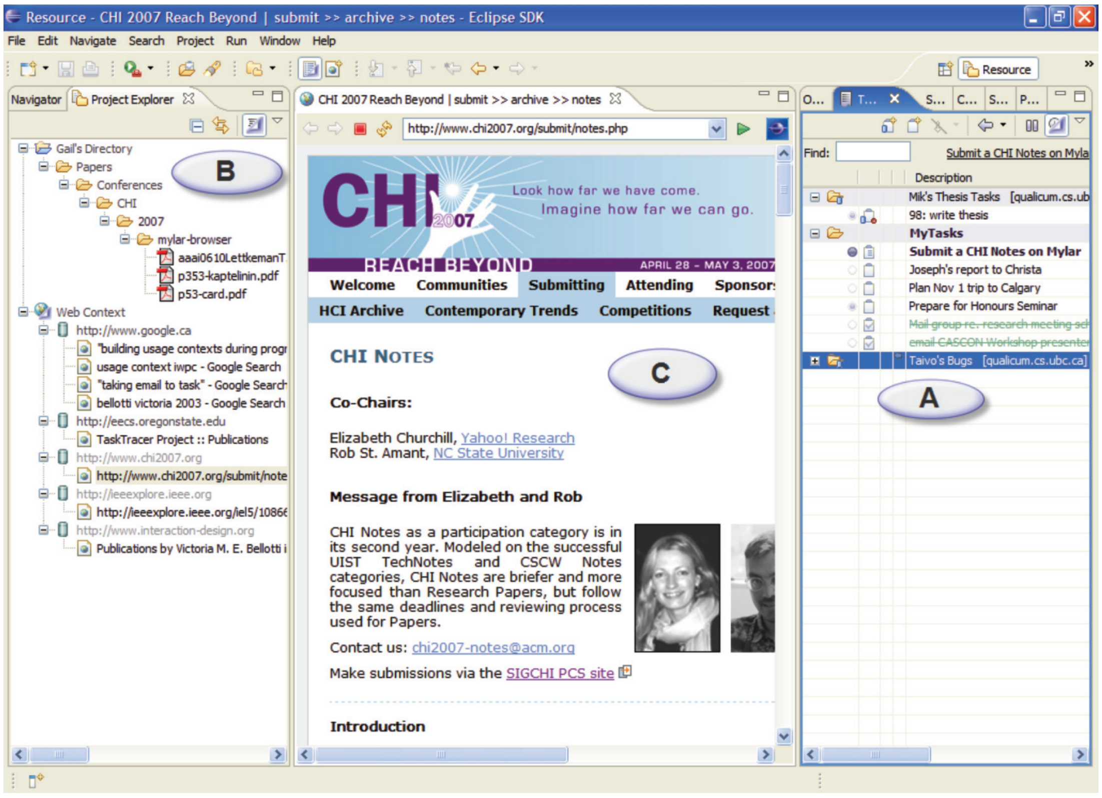
opposed to source code elements. The different kinds of interactions provide different scaled values to the DOI computation. This introduces the possibility of tailoring the DOI computation and the resultant task context to a user. To date, we have not found a need to alter the scalings for different kinds of events per user. More details on the computation of DOI and the scaling factors are available elsewhere (Kersten and Murphy 2006).

Other interaction events, indirect events, are placed into the interaction history by the system creating the task context. Our system currently supports two forms of indirect events. Propagation events introduces interaction events for structurally related elements to an element directly interacted with by the knowledge worker. For instance, if a programmer interacts with a method in a Java class, a propagation event is introduced for all hierarchically related elements, such as the enclosing Java class or package. Similarly, a propagation event may be introduced for a directory enclosing a spreadsheet with which a knowledge worker interacts. The second kind of indirect event is a prediction event, which describes possible future interactions our system anticipates the knowledge worker might perform. For a programmer, a prediction event might be predicting interest in a test case that could exercise a changed programming element. For a knowledge worker, a prediction event might be a web page referenced by a web page with which the knowledge worker has directly interacted that our system predicts will be useful to the knowledge worker. In essence, prediction events are a means of capturing recommendations within a knowledge worker's interaction history.

Representing task contexts as information, namely a sequence of interaction events, makes it possible to transform the context using straightforward operations. For example, interaction histories for two subtasks, T1 and T2, can be composed to form a composite context of an aggregating task, T3, by combining the interaction event sequences of T1 and T2's interaction histories. As another example, a task context may be sliced to meet a particular constraint, such as showing all elements of information on which an edit was performed, simply by restricting the interaction history to the events of interest.

Task-Focused User Interface

Task contexts reduce the friction faced by a knowledge worker performing a task when they are used to facilitate task switches and to readily locate and use information pertaining to a task-at-hand. Figure 2 shows an instance of a task-focused user interface prototype we built. This user interface includes a task list populated by shared tasks queried from a task repository, from email, or created by a user locally (figure 2a); a viewing and editing pane for resources associated with a task (figure 2c); and a tree view that provides access to local files, shared files, and web pages in the task context (figure 2b). In figure 2, a knowledge worker has activated a task labeled "Submit a CHI Notes on Mylar" in the rightmost pane, as indicated by the blue dot. Activating a task is a one-



Resource - CHI 2007 Reach Beyond | submit >> archive >> notes - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Resource
Navigator
Project Explorer
Gail's Directory
Papers
Conferences
CHI
2007
mylar-browser
aaai0610LettkemanT
p353-kaptelinin.pdf
p53-card.pdf
Web Context
http://www.google.ca
"building usage contexts during progr
usage context iwpc - Google Search
"taking email to task" - Google Search
bellotti victoria 2003 - Google Search
http://eecs.oregonstate.edu
TaskTracer Project :: Publications
http://www.chi2007.org
http://www.chi2007.org/submit/note
http://ieeexplore.ieee.org
http://ieeexplore.ieee.org/iel5/10866
http://www.interaction-design.org
Publications by Victoria M. E. Bellotti i
B
CHI 2007 Reach Beyond | submit >> archive >> notes
http://www.chi2007.org/submit/notes.php
CHI 2007
Look how far we have come.
Imagine how far we can go.
REACH BEYOND
APRIL 28 – MAY 3, 2007
Welcome Communities Submitting Attending Sponsor
HCI Archive Contemporary Trends Competitions Request
C
O... T... S... C... S... P...
Find:
Submit a CHI Notes on Myla
Description
Mik's Thesis Tasks [qualicum.cs.ub
98: write thesis
MyTasks
Submit a CHI Notes on Mylar
Joseph's report to Christa
Plan Nov 1 trip to Calgary
Prepare for Honours Seminar
~~Mail group re. research meeting sch~~
~~email CASCON Workshop presenter~~
Taivo's Bugs [qualicum.cs.ubc.ca]
A

CHI NOTES

Co-Chairs:

Elizabeth Churchill, Yahoo! Research
Rob St. Amant, NC State University

Message from Elizabeth and Rob

CHI Notes as a participation category is in its second year. Modeled on the successful UIST TechNotes and CSCW Notes categories, CHI Notes are briefer and more focused than Research Papers, but follow the same deadlines and reviewing process used for Papers.

Contact us: chi2007-notes@acm.org

Make submissions via the SIGCHI PCS site

Introduction

Figure 2. Task-Focused UI for Knowledge Workers.

click operation that triggers the collection of an interaction history to be associated with that task. The leftmost pane in figure 2, shows the documents and web pages that have a positive DOI.

A task-focused user interface makes use of several mechanisms to facilitate the display and management of task contexts. Mechanisms can be defined at the level of an individual view or window that provides access to information in a task context and between views or windows as described in table 2. These mechanisms faciliate multitasking and help a knowledge worker combat information overload. The filtering that has been applied to the Project Explorer pane on the leftmost side of figure 2 enables the relevant documents and web pages to all appear on the screen with other documents and web pages unrelated to the task (that is, with DOI values less than zero) hidden from view. If the knowledge worker is interrupted and switches to work on the "Prepare for Honours Seminar" task, a click of the "Prepare for Honours Seminar" task will close all editors related to the "Submit a CHI Notes on Mylar" task, open editors that had been open the last time the "Prepare for Honours Seminar" task was worked on, and will refilter and focus the leftmost Project Explorer pane on those documents and web pages with positive DOI values for the "Prepare for Honours Seminar" task. This one-click multitasking reduces the friction knowledge workers currently face when trying to switch between tasks.

Other task-focused user interface mechanisms, such as ranking, have more utility in domains with highly structured information, such as programming. In the Eclipse Mylyn open source project, which provides a task-focused user interface for programming, ranking is used to ensure that recommendations of code to use in completion are based on the current task context. Expansion of trees is also

Target	Mechanism	Description
view	Filtering	Elements below a certain DOI threshold are excluded
view	Ranking	Elements are ranked or sorted by DOI
view	Decoration	Foreground or background colour of elements indicates DOI
view	Expansion Management	Auto-expand tree nodes to correspond to a slice of interest. Elide uninteresting text.
window	View Management	Auto-apply focusing mechanisms to view on task context activation
window	Editor Management	Auto-close editors corresponding to uninteresting elements
window	Perspective Management	Auto-restore views associated with a context on activation

Table 2. Task-Focused UI Mechanisms.

critical in programming where source code elements can be deeply nested in hierarchical tree views.

Friction can also be reduced by allowing knowledge workers to collaborate through shared task contexts. When two knowledge workers work on the same task there is often a desire to pass the context of a task back and forth. A task-focused user interface can facilitate this sharing by allowing one knowledge worker to import a task context from another knowledge worker. Once imported, a knowledge worker can use the mechanisms of the interface to see the resources the other worker has considered as they have performed the task and to build upon their colleagues' work.

Using Task Contexts

To understand whether task contexts improve the flow of a knowledge worker's activity, we have been investigating the use in practice of task-focused user interfaces supporting task contexts. We have focused on the use of the concept in practice as the critical questions of interest are the overall impact on a knowledge worker's productivity and whether the contexts capture and model the appropriate relevant information to a task over time.

We have conducted three specific studies of the use of task contexts with different populations. In 2005, we conducted a field study involving 16 programmers to investigate whether the use of task contexts could improve programmer productivity. In 2007, we conducted a field study involving eight knowledge workers to investigate whether task contexts provided utility for knowledge workers dealing with less structured information than the programmers we had previously studied. In 2013, we performed a case study, analyzing the longitudinal use of task contexts by three operational personnel at the software development company with which the authors are principles. Each of these studies has provided different insight into the utility and use of task contexts; we report on each briefly.

Programmer Field Study

The field study of programmers used a within-subject study design. We were interested in how the use of a task-focused interface would affect programmer productivity as measured by whether programmers edited more with the help of a task-focused interface. We used a ratio of the number of edits compared to the number of selections as a measure of productivity. We termed this measure the *edit* ratio. We computed the edit ratio based on edits and selections of programming artifacts so that interaction with task management features provided would not skew the results.

The participants in the study were industry Java programmers who used the open source Eclipse integrated development environment.[3] These participants were recruited at an industry conference. Of the 99 programmers who signed up for the study, the majority of the individuals were industry programmers, with about half working in organizations with more than 50 people and who most identified their industry sector as software manufacturing.

For the study, the programming of a participant was monitored automatically until a certain threshold of work, defined to be 1000 edit events over no less than two weeks. At the end of this period, our study tools sent us an anonymous interaction history for the participant, and the participant was invited to install our task-focused interface for programming, then called the Mylar project.[4] This task-focused user interface was a version of the Eclipse integrated development environment augmented with the same task list as shown in figure 2 and focus capabilities for all programming views, including the editor and a hierarchical view of programming elements. Periodically after starting to use the task-focused interface, the study tools would prompt participants to upload their interaction history to us. Of the 99 participants who started the study, only 16 met the thresholds of activity we set to be considered as a study participant.

For the 16 participants who programmed regularly in both the before and after-task focused interface periods, we compared their edit ratio before and after use of the task-focused user interface using a paired *t*-test. We found that the use of the task-focused interface increased the edit ratio of participants with statistical significance. From the viewpoint of this measure, the task-focused interface did improve programmer productivity.

We also wanted to know if the context of programmers' tasks was captured accurately. A qualitative analysis of the task contexts formed by the programmers showed that 84 percent of the selection events recorded were on elements with a positive DOI, 5 percent of the selection events were of elements with a propagated or predicted interest, and only 2 percent were of elements with a negative DOI. The high rate of selection of elements with a positive DOI and low rate of selection of elements with a negative DOI suggests that the contexts were capturing much of the desired information for working on a task and that the appropriate information was staying modeled in the task context.

Knowledge Worker Field Study

In a 2007 field study of a more general class of knowledge workers, we wanted to gain knowledge about three questions: (1) do knowledge workers find it useful to create contexts for their tasks that mix file and web documents, (2) do these knowledge workers revisit tasks, and (3) does the DOI function serve to keep the relevant information in a knowledge worker's task context?

For this study, given the early nature of the tool encompassing the task-focused interface, we targeted knowledge workers within and related to the University of British Columbia. This recruitment resulted in eight participants who installed the tool shown in figure 2 on their work computer and who answered usage questions through email and in person: P1 (technology assessment), P2 (master's student), P3 (teacher), P4 (instructor), P5 (CEO), P6 (project coordinator), P7 (student coordinator), and P8 (communications coordinator).

Participants were asked to use the tool in their daily work as they saw fit. The tool required them to define tasks using a name for each task and to access their file system and web pages through the tool. The average number of workdays for which the participants used the tool was 10.8 days (± 9.8). The average tasks worked on per day was 3.1 (± 1.54) and the average activations per day was 5.5 (± 3.4). For all but one participant, more tasks were activated per day than tasks worked on per day indicating that participants returned to tasks on which they had worked earlier that day. This data suggests that subjects do revisit tasks and that they use task contexts for the purpose of restoring context when multitasking.

We analyzed the task contexts resulting from the participants interaction with documents and web pages. Six participants had task contexts that mixed document access with web pages. The average number of web pages in a context varied dramatically across users from an average of 3 web pages to an average of 69. The average number of files also varied dramatically from an average of none to an average of 68. Six of the participants generated enough interaction to result in substantial decay in their web documents, while five participants experienced decay in their file documents. The high-values of decay (over 20 percent for most participants) indicates the need to weight the relevance of elements in the context to avoid overloading the user with information. We asked each participant if the task contexts showed too much information (three participants), too little information (two participants), or just the right information as they worked (three participants).

From this study, we found that knowledge workers do want to include both documents and web pages from their information space in a task context, that the knowledge workers revisit tasks, and that the DOI functions does serve to retain relevant information while removing irrelevant information from the view of a task context.

Knowledge Worker Longitudinal Field Study

To provide insight into the nature of task contexts over a longer usage period, we performed a study of the use of task-focused interfaces to run the operations of a software company of which the authors are cofounders. In this organization, a shared task repository is used to communicate and collaborate about many aspects of business operations, including aspects of sales, business development, finance, human resources, marketing, and general administration. At the time of analysis, the shared task repository was operational for three years and eight months and hosted 8402 tasks of which 5707 (70.9 percent) were marked as completed. The average number of comments left on tasks as people collaborated was 8.14 ± 13.63. In this shared task repository, 2404 (28.6 percent) of the tasks had some sharing of files and web documents between users; this percentage does not include artifacts captured as contexts personally by individuals.

We selected a convenience sample of three individuals using this shared task repository, including one author of this article. These individuals were using a version of the Eclipse Mylyn open source tooling tailored to knowledge workers similar to the tool shown in Figure 2. The main difference between the tool used by these individuals and the tool used in the earlier knowledge worker study was more support for tracking detailed parts of information artifacts, such as parts of web pages. We analyzed data collected as these individuals worked over periods ranging from just over 300 to almost 600 days. We found that the maximum number of task activations per day ranged between 16 and 24. The average number of task reactivations per day was between 3 and 4 and the maximum number of task reactivations across the period was between 85 and 148.

From an analysis of the task contexts associated with these individuals' work, we found that the average number of files per context exceeded the average interesting files per context, suggesting decay was effective in trimming the size of an individual's task context. We saw a similar trend for the web pages referenced as part of a task context.

From this study, we can see that tasks are revisited substantially over time and that the contexts formed for a task are used by knowledge workers. We also learned there is a need for models of task context to automatically trim the contents of the task context and reflect a knowledge worker's current needs.

Beyond Task Contexts

Task contexts provide a full history of the interaction with a task. When considered alongside the rich history of documents that are often maintained through version repositories, shared file systems, and the like, there arises the possibility of rewinding both task context and artifact state concurrently, enabling a knowledge worker to easily return to a previous state in their work history. In essence, such a facility could be used to rewind to a previous activity state of a knowledge worker.

To date, we have provided and experimented with situations in which one task can have multiple task contexts at different points in time contributed by different individuals, but tasks do not share task contexts. At times, users of our tools have requested the ability to allow task contexts to refer to other tasks and their associated contexts. For example, if task A has subtasks B and C, the context of task A could be the composite of task B and C. Support for cascading task contexts along such relationships would substantially change our treatment in the tools of task contexts. Instead of a task context being a snippet of one user's interaction history, a task context could be identified by a set of spans of an interaction history and relations to other such spans. Activating a task could then involve swapping in the corresponding segments of interaction. Such a facility would further support rewinding interaction (and hence activity) across one or more tasks.

More recently, in a commercial context, we have been gaining experience with the representation of tasks, communication activity, and artifacts across many different repositories serving many different kinds of knowledge workers. This exposure has helped us refine our model for representing task and task context information. We see artifacts as playing more than one role in any larger schema describing tasks and their context. In particular, artifacts and their versions need to be more strongly modeled as parts of tasks at different points in time with task contexts referring to these different versions.

Our work to date has focused on the representation of task context once a task is indicated by the user. Future directions to investigate include the automatic determination of task boundaries and the integration of user goals within a task to task context information. Work in activity recognition that can consider the structure of user goals may be helpful in improving the representation and population of task contexts (Natarajan et al. 2008) and in enhancing task-focused user interfaces with additional inference features.

Summary

There is no lack of information available to knowledge workers today. Although current tools make it easy for knowledge workers to browse and query the information space with which they must work, these tools do not support a knowledge worker in managing the information needed for the tasks the worker performs. Proper and easy management of information relevant to tasks is needed given the high velocity with which knowledge workers switch tasks. Without appropriate support, knowledge workers are burdened with repeatedly creating and recreating the context they need to get work done.

We have found that the task-focused interface that leverages knowledge workers' episodic memory to form and recall tasks along with a model for task context based on the knowledge workers' interaction and activity can provide a means to reduce the friction and improve the flow of knowledge work.

Acknowledgements

This work was supported in part by IBM CAS, NSERC and Tasktop Technologies. We thank the participants in the studies and the Eclipse Mylyn community for their ongoing input.

Notes

1. *Merriam-Webster's Collegiate Dictionary,* 2003.
2. eclipse.org, verified 11/22/13
3. eclipse.org, verified 11/22/13
4. This interface was put into open source as the first version of the Eclipse Mylyn project.

References

Belotti, V.; Ducheneaut, N.; Howard, M.; and Smith, I. 2003. Taking Email to Task: The Design and Evaluation of a Task

Management Centered Email Tool. In *Proceedings of the 2003 ACM SIGCHI Conference on Human Factors in Computing Systems,* 345–352. New York: Association for Computing Machinery.

Card, S., and Henderson, D. 1987. A Multiple, Virtual-Workspace Interface to Support User Task Switching. In *Proceedings of the 1987 ACM SIGCHI Conference on Human Factors in Computing Systems,* 53–59. New York: Association for Computing Machinery.

Dourish, P. 2004. What We Talk About When We Talk Context. *Personal and Ubiquitous Computing* 8(1): 19–30. dx.doi.org/10.1007/s00779-003-0253-8

Dourish, P.; Edwards, K.; LaMarca, A.; and Salisbury, M. 1999. Using Properties for Uniform Interaction in the Presto Document System. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, 55–64. New York: Association for Computing Machinery. dx.doi.org/10.1145/320719.322583

Dragunov, A.; Dietterich, T.; Johnsrude, K.; McLaughlin, M.; Li, L.; and Herlocker, J. 2005. Tasktracer: A Desktop Environment to Support Multi-Tasking Knowledge Workers. In *Proceedings of the 10th International Conference on Intelligent User Interfaces,* 75–82. New York: Association for Computing Machinery.

Gonzales, V., and Mark, G. 2004. Constant, Constant, Multi-Tasking Craziness: Managing Multiple Working Spheres. In *Proceedings of the 2004 ACM SIGCHI Conference on Human Factors in Computing Systems,* 113–120. New York: Association for Computing Machinery. dx.doi.org/10.1145/985692.985707

Kaptelinin, V. 2003. UMEA: Translating Interaction Histories into Project Contexts. In *Proceedings of the 2003 ACM SIGCHI Conference on Human Factors in Computing Systems,* 353–360. New York: Association for Computing Machinery.

Kersten, M., and Murphy, G. C. 2006. Using Task Context to Improve Programmer Productivity. In *Proceedings of the 14th ACM SIGSOFT International Symposium on the Foundations of Software Engineering, 1–11. New York: Association for Computing Machinery.*

Kersten, M., and Murphy, G. 2012. Task Context for Knowledge Workers. In Activity Context Representation: Techniques and Languages: Papers from the 2012 AAAI Workshop. Technical Report WS-12-05. Palo Alto, CA: AAAI Press..

Lettkeman, A.; Stumpf, S.; Irvine, J.; and Herlocker, J. 2006. Predicting Task-Specific Webpages for Revisiting. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence,* 1369-1374. Menlo Park, CA: AAAI Press.

Natarajan, S.; Bui, H. H.; Tadepalli, P.; Kersting, K.; and Wong, W.-K. 2008. Logical Hierarchical Hidden Markov Models for Modeling User Activities. In *Inductive Logic Programming,* Lecture Notes in Computer Science volume 5194, 192-209. Berlin: Springer. dx.doi.org/10.1007/978-3-540-85928-4_17

Plotnik, R. 2004. I*ntroduction to Psychology.* San Francisco: Wadsworth Publishing Company.

Rattenbury, T., and Canny, J. 2007. CAAD: An Automatic Task Support System. In *Proceedings of the 2007 ACM SIGCHI Conference on Human Factors in Computing Systems,* 687-696. New York: Association for Computing Machinery. dx.doi.org/10.1145/1240624.1240731

Snowden, J. 1996. Semantic-Episodic Memory Interactions in Semantic Dementia: Implications for Retrograde Memory Function. *Cognitive Neuropsychology* 13(8): 1101–1139. dx.doi.org/10.1080/026432996381674

Mik Kersten is chief executive officer and cofounder of Tasktop Technologies, creator and lead of the Eclipse Mylyn open source project, and the inventor of the task-focused interface. His goal is to create the collaborative infrastructure needed to connect knowledge workers in the new world of software delivery. He earned his Ph.D. in computer science from the University of British Columbia.

Gail C. Murphy is a professor in the Department of Computer Science and associate dean (research and graduate studies) at the University of British Columbia. She is a cofounder and currently chief scientific officer of Tasktop Technologies. Her research interests are in improving the productivity of software developers and knowledge workers by giving them tools to identify, manage, and coordinate the information that really matters for their work. She earned her Ph.D. in computer science and engineering from the University of Washington.