EXPLOITING THE USER INTERACTION CONTEXT FOR AUTOMATIC TASK DETECTION

Didier Devaurs, Andreas S. Rath & Stefanie N. Lindstaedt

Taylor & Francis
Taylor & Francis Group

EXPLOITING THE USER INTERACTION CONTEXT FOR AUTOMATIC TASK DETECTION

Didier Devaurs[1,2], Andreas S. Rath[3], and Stefanie N. Lindstaedt[3,4]

[1]*CNRS; LAAS, Toulouse, France*
[2]*Université de Toulouse; UPS, INSA, INP, ISAE; LAAS, Toulouse, France*
[3]*Know-Center GmbH, Graz, Austria*
[4]*Knowledge Management Institute, Graz University of Technology, Graz, Austria*

☐ *Detecting the task a user is performing on his/her computer desktop is important in order to provide him/her with contextualized and personalized support. Some recent approaches propose to perform automatic user task detection by means of classifiers using captured user context data. In this paper we improve on that by using an ontology-based user interaction context model that can be automatically populated by (1) capturing simple user interaction events on the computer desktop and (2) applying rule-based and information extraction mechanisms. We present evaluation results from a large user study we have carried out in a knowledge-intensive business environment, showing that our ontology-based approach provides new contextual features yielding good task-detection performance. We also argue that good results can be achieved by training task classifiers "offline" on user context data gathered in laboratory settings. Finally, we isolate a combination of contextual features that present a significantly better discriminative power than classical ones.[1]*

INTRODUCTION

Today, knowledge workers have to handle incessantly increasing amounts of digital information, in terms of text documents, emails, multimedia files, etc., located on their own computer desktops, on company networks, and on the World Wide Web. Personal information management and search and retrieval systems can help in coping with this ever-growing challenge. They can do so even more efficiently if they provide contextualized

and personalized support. Various research areas have already emphasized the use of contextual information as one of the key elements for enhancing current applications. Examples can be cited in personal information management Sauermann, Bernardi, and Dengel 2005; Catarci et al. 2007; Chernov et al. 2008; Jones et al. 2008), user modeling (Van Kleek and Shrobe 2007), information retrieval (Callan et al. 2007; Tang et al. 2007; Mylonas et al. 2008), technology-enhanced learning (Schmidt 2005; Wolpers et al. 2007), and others.

An important issue in the context detection area is automatic user task detection on the computer desktop (Dey, Abowd, and Salber 2001; Coutaz et al. 2005). If the user's current task is automatically detected, the user can be better supported with relevant information, such as learning and work resources or task guidance. A classical approach has been to model task detection as a machine-learning problem. However, the focus so far has been on using only text-based features and switching sequences (Oliver et al. 2006; Shen, Li, and Dietterich 2007; Chernov et al. 2008; Granitzer et al. 2008), which do not rely on ontology models, for detecting the user's task. Furthermore, controlled user studies and standard datasets for the evaluation of task-detection approaches are still missing. This implies that the mechanisms underpinning the achievement of good task-detection performance are yet to be unveiled.

In this paper we focus on (1) proposing new contextual features yielding improvements over the current results achieved by task-detection techniques and (2) studying some aspects of the task-detection problem in order to better understand in which settings it can be successfully applied. The first part of our contribution consists of proposing a generic ontology-based user context model for increasing the performance of user task detection. Our approach is based on using context sensors to capture simple interaction events (keyboard strokes and mouse clicks) from the user's computer desktop. Then, we utilize rule-based and information extraction techniques to automatically populate our user interaction context model by discovering instances of concepts and deriving interconcepts relationships. Using an ontology-based user context model offers several advantages, such as an easy integration of new contextual attention metadata (Wolpers et al. 2007), a simple mapping of the raw sensor data into a unified model, and an easy extendability ofthe user context model with concepts and properties about new resources and user actions. We present an evaluation of our approach based on a large[2] controlled user study (containing five task classes and 220 task instances recorded from 14 participants) that we have carried out in a knowledge-intensive business environment. It shows that using an ontology-based representation of the user context allows derivation of new ontology-specific, performance-increasing features for machine learning algorithms.

The second part of our contribution consists of investigating the following questions: (1) How good can the performance of a task classifier be when used in a real work environment, if it has been trained with contextual data gathered in laboratory settings? (2) Which are the automatically observable contextual features that allow for good task-detection performance? Both questions are concerned with work efficiency. The goal of the first is to determine whether a task classifier can be trained "offline". This would spare the user the burden of performing a manual training during work processes, which might slow down the computer and have a negative influence on user work efficiency and experience. The second aims at finding the most discriminative features among the automatically captured contextual features in order to achieve a good balance between task-detection accuracy and classification workload. This would also influence which context sensors have to be developed to perform user task detection. To get a first impression on the answers to these questions, we have analyzed the classification results provided by the user study previously mentioned. In this study, users performed their tasks both on a single laboratory computer and on their personal workstations. Our first results indicate that: (1) reliable detection of real tasks via offline training is possible, (2) the good discriminative power of the classical *window title* feature (Oliver et al. 2006; Shen, Li, and Dietterich 2007; Granitzer et al. 2008) is confirmed, and (3) classification accuracy is significantly increased by using a combination of six features specific to our approach.

The rest of this article is organized as follows. First we describe how we define, conceptualize, and model the user interaction context. We mainly focus on the presentation of our ontology-based context model. Then we elaborate on the sensors recording the interaction events and the techniques used to automatically populate the proposed ontology. Next we present the approach we follow to perform user task detection, and how we evaluate it. Our experimental results are discussed in the following section, including a comparison of several contextual feature sets in terms of task detection accuracy, as well as an analysis of several aspects of the task detection problem. Finally, we provide concluding remarks and an outlook on future work.

MODELING THE USER INTERACTION CONTEXT

Our view of the "*user context*" goes along with Dey's definition that context is "*any information that can be used to characterize the situation of entities that are considered relevant to the interaction between a user and an application, including the user and the application themselves*" (Dey, Abowd, and Salber 2001). We specifically focus here on the *user interaction context* that we define

as the interactions of the user with applications and resources on the computer desktop. With this definition in mind, we will sometimes simply use the term *user context* subsequently. Our perspective puts the individual user and his/her actions at the center of attention. It aims at learning as much as possible about this individual and his/her actions. Our goal is to study the relations between the event objects generated by the user's interactions with the computer system, and their meaning and relevance to the user's task.

Conceptual Model–The Event Aggregation Pyramid

The conceptual representation we propose for the user interaction context is the *event aggregation pyramid* (see Figure 1). Its layers represent the different aggregation levels of the user's actions. At the bottom of the pyramid are event objects (or simply events) that result from the user's interactions with the computer desktop. Above are aggregate events (or simply event-blocks), which are sequences of events that belong logically together, each event-block encompassing the user's actions associated with a specific resource acted on. At the top are tasks, defined as groupings of event-blocks representing well-defined steps of a process. This model also integrates the idea of delivering resources that are relevant to the user's actions based on his/her information needs.

UICO–User Interaction Context Ontology

A context model is needed to store the user context data in a machine-processable form. Various context model approaches have been
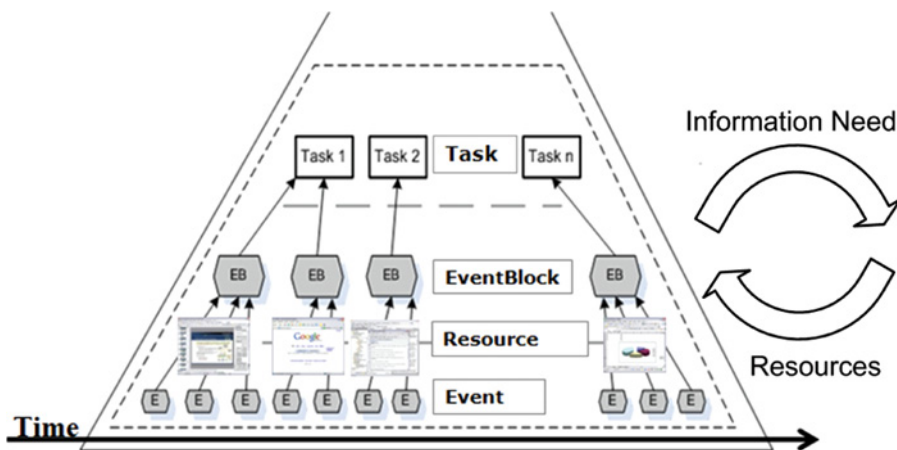

FIGURE 1 The event aggregation pyramid represents our conceptual view of the user interaction context. (Figure is provided in color online.)

proposed, such as key-value models, markup-scheme models, graphical models, object-oriented models, logic-based models, or ontology-based models (Strang and Linnhoff-Popien 2004). The ontology-based approach has been advocated as being the most promising mainly because of its dynamicity, expressiveness, and extensibility (Strang and Linnhoff-Popien 2004; Baldauf, Dustdar, and Rosenberg 2007). In our specific case, we argue that an ontology-based context model offers the following advantages: (1) It allows easy integration of new context data sensed by context observers and mapping of the sensor data into a unified context model. (2) It can be easily extended with concepts and properties about new resources and user actions. (3) The relationships between resources on various granularity levels can be represented. (4) The evolution of datatype properties (i.e., data and metadata) into objecttype properties (i.e., relations between instances of ontology concepts) can be easily accomplished. (5) Being a formal model, it also allows other applications and services to build on it and to access the encapsulated context information in a uniform way. Most importantly, we subsequently show that the performance of user task detection can be enhanced by using an ontology-based context model.

Our *User Interaction Context Ontology* (UICO) can be seen as the realization of the event aggregation pyramid with the support of semantic technologies. We follow a bottom-up approach and build the UICO by incrementally adding relations when new sensor data or algorithms are added. The UICO holds the contextual information representing the user interaction context. This includes the data provided by the context sensors observing the user's actions on the computer desktop and the information automatically derived from it. Based on the application domain in which the UICO is used, we can decide which relations and concepts are useful and which are not. In the case of ontology-based task detection, we study concepts and relations that are significant for a specific task, i.e., highly discriminative among tasks. At the moment, the UICO contains 88 concepts and 272 properties, and is modeled in OWL-DL,[3] using the Protégé ontology modeling tool[4] (see Figure 2). From these 272 properties, 215 are datatype properties and 57 are objecttype properties. From a top-level perspective, we define five different dimensions in the UICO: the *action dimension*, the *resource dimension*, the *information need dimension*, the *user dimension* and the *application dimension*.

Action Dimension

The action dimension consists of concepts representing user actions, task states, and connection points to top-down modeling approaches. User actions are distinguished based on their granularity, corresponding to the levels of the event aggregation pyramid: Event at the lowest level, then

FIGURE 2 The concept hierarchy of the User Interaction Context Ontology (UICO), visualized with the Protégé tool. The left area contains the *action dimension*, the right area contains the *resource dimension*, and the bottom area contains the *user dimension*, and the *information need dimension*. (Figure is provided in color online.)

EventBlock and then Task. The ActionType concepts specify which types of actions are defined on each level. Currently we distinguish action types only on the event level (EventType concept): there are 25 of them (Open, Save, Print, Copy, Reply, ClipboardChange, DesktopSearch, etc.). As an example, if the user clicks on the search button of a search engine's web page, this user interaction will generate an Event of type WebSearch. The TaskState concept and its subconcepts model the way the user manages and executes tasks. We borrow the types of task states from the *Nepomuk Task Management Model* (Groza et al. 2007). These types allow us to model a user creating, executing, interrupting, finishing, aborting, approving, and archiving a task. Besides, each change in task state is tracked via the TaskStateChange concept. The Model concept has been introduced to have connection points to top-down modeling approaches. Currently only one connection point is available: the TaskModel concept. This concept is similar to what is defined in the areas of workflow management systems and task-process analysis. At the moment, the TaskModel concept is used to categorize a task. An example of instances of the TaskModel and Task concepts is "Planning a journey" and "Planning the journey to CHI 2011," respectively.

Resource Dimension

The resource dimension contains concepts for representing resources on the computer desktop. We specifically focus on modeling resources commonly used by the knowledge workers we have interviewed. We have identified 16 key resource concepts (email, person, image, file, folder, etc.) but more can be easily added if required. A resource is constructed from the data and metadata captured by the context sensors. Relations can be defined between concepts of the resource dimension and the action dimension, for modeling based on which resources the kinds of user actions have been executed. For example, if the user enters a text in a Microsoft Word document, all keyboard entries will be instances of the Event concept, connected via the isActionOn objecttype property to the same instance of a TextDocument (and a xtttFileResource) representing that document.

Information Need Dimension

The information-need dimension represents the context-aware proactive information delivery aspect of the UICO. An information need is detected by a set of fixed rules, based on the available contextual data. An InformationNeed concept has properties defining the accuracy of the detection and the importance to fulfill this need within a certain time frame. An information need is associated with the user's action(s) that trigger(s) it, thus creating a connection between the information-need

dimension and the action dimension. The information-need dimension is also connected to the resource dimension, because each resource that helps to fulfill the user's information need is related via the objecttype property suggestsResource to the InformationNeed.

User Dimension

The user dimension contains only two concepts: User and Session[5]. Its main interest is to maintain a link between a user and the interaction events he/she performs. The User concept defines basic user information such as user name, password, first name, and last name. The Session concept is used for tracking the time of a user login and the duration of a user session in our application. The user dimension is connected to the action dimension in that each Action is associated with a User via the objecttype relation hasUser. It is also indirectly related to the resource and information-need dimensions via the action dimension.

Application Dimension

The application dimension is a "hidden" dimension, because it is not modeled as concepts in the UICO. It is, however, present in that each user interaction happens within the focus of a certain application, such as Microsoft Word or Windows Explorer. The Event concept holds the information about the user interaction with the application through the datatype properties hasApplicationName and hasProcessId. Standard applications that run on the Microsoft Windows desktop normally consist of graphical user interface (GUI) elements. Console applications also contain GUI elements such as the window itself, scroll bars, and buttons for minimizing, maximizing, and closing the application. Most of GUI elements possess an associated accessibility object[6] that can be accessed by context sensors. Datatype properties of the Event concept hold the data about the interactions with GUI elements. We show later on that these accessibility objects play an important role in task detection. A resource is normally accessed by the user within an application, hence there exists a relation between the resource dimension and the application dimension. This relation is indirectly captured by the relation between the resource dimension and the action dimension, i.e., by the datatype property hasApplicationName of an Event.

Related Ontologies

The UICO is similar to the *Personal Information Model Ontology* (PIMO) (Sauermann, van Elst, and Dengel 2007) developed in the NEPOMUK research project,[7] in terms of representation of desktop resources.

However, they differ in terms of granularity of concepts and relations. The UICO is a fine-grained ontology, driven by the goal of automatically representing low-level captured contextual information, whereas the PIMO enables the user to manually extend the ontology with new concepts and relations, to define his/her environment for personal information management. The *native operations* (NOP) ontology,[8] used in the Mymory project (Biedert, Schwarz, and Roth-Berghofer 2008), models native operations (e.g., AddBookmark or CopyFile) on *generic information objects* (e.g., email, bookmark, or file) recorded by system and application sensors. Native operations are similar to the UICO's ActionType concepts, and more specifically to the EventType concepts. The DataObject concepts describe several desktop resources in a more coarse-granular way than we do for the UICO's Resource concepts. In Xiao and Cruz (2005) a layered and semantic ontology-based framework for personal information management following the principles of *semantic data organization, flexible data manipulation,* and *rich visualization* is proposed. The framework consists of an *application layer,* a *domain layer,* a *resource layer* and a *personal information space.* The resource dimension of the UICO can be seen as a combination of these domain and resource layers, because resource instances are mapped to concepts of the domain layer.

AUTOMATIC POPULATION OF THE CONTEXT ONTOLOGY

It is not realistic to have a user manually providing the data about his/her context on such a fine-granular level, as is defined in our UICO. Hence, semi-automatic and automatic mechanisms are required to ease the process of "populating" the ontology. We use rule-based, information extraction and machine-learning techniques to automatically populate the ontology and automatically derive relations among the model's entities. We now describe how we build instances of concepts and augment relations among the concept instances. We also show which kind of sensors we use to observe user interaction events on the computer desktop, how we discover resources the user has utilized, unveil connections among resources, and aggregate single user interaction events into event-blocks and tasks.

Context Observation

Context observation mechanisms are used to capture the user's behavior while he/she is working on his/her computer desktop, i.e., performing tasks. Simple operating system and application events initiated by the user while interacting with the desktop are recorded by context observers, acting as event object sources. Our use of context observers is

similar to the approach followed by contextual attention metadata (Wolpers et al. 2007) and other context observation approaches (Dragunov et al. 2005; Van Kleek and Shrobe 2007). Context observers, also referred to as context sensors, are programs, macros, or plug-ins that can be distinguished based on the origin of the data they deliver. *System sensors* are deeply hooked into the operating system. *Application sensors* are developed for standard office applications. We focus on supporting applications that knowledge workers use in their daily work, which is generally in a Microsoft Windows environment. Table 1 presents a list of the available system and application sensors, and a description of what kind of contextual information they are able to sense. The produced events are sent as an XML stream via an event channel to the context capturing framework (i.e., our event-processing agent) for storage,[9] processing, and analysis. We also refer to this contextual attention metadata stream as the *event stream.* Our event-processing network is rather a static one, evolving only when new sensors are added.

TABLE 1 List of Our Application and System Sensors, and Data Recorded by These Sensors

Sensor	Observed data and metadata
Application Sensors	
Microsoft Word	document title, document url, language, text encoding, content of visible area, file name, folder, user name
Microsoft PowerPoint	document title, document url, document template name, current slide number, language, content, file name
Microsoft Excel	spreadsheet title, spreadsheet url, worksheet name, content of the currently viewed cell, authors, language, file name, file uri, folder, user name
Microsoft Internet Explorer	currently viewed url, urls of embedded frames, content as html, content as plain text
Microsoft Explorer	currently viewed folder/drive name, url of folder/drive path
Mozilla Firefox 2.x and 3.x	currently viewed url, urls of embedded frames, content as html
Mozilla Thunderbird	(html/plain text) content of currently viewed or sent email, subject, unique path (uri of email/news message) on server, user's mail action (compose, read, send, forward, reply), received/sent time, email addresses and full names of the email entries
Microsoft Outlook 2003/2007	(create, delete, modify, open and distribute) tasks, notes, calendar entries, contacts, data about email handling
Novell GroupWise email client	(create, delete, modify, and distribute) tasks, notes, calendar entries, todos, data about email handling
System Sensors	
File System Sensor	copying from/to, deleting, renaming from/to, moving from/to, modification of files and folders (file/folder url)
Clipboard Sensor	clipboard changes (i.e., text copied to clipboard)
Network Stream Sensor	header and payload content from network layer packets (http, ftp, nttp, smtp, messenger, ICQ, Skype . . .)
Generic Windows XP System Sensor	mouse movement, mouse click, keyboard input, window title, date and time of occurrence, window id/handle, process id, application name
Accessibility Object Sensor	name, value, description, help text, help text description, tooltip of the accessibility object

Resource Discovery

The resources that populate the ontology model are, for example, links, web pages, persons, organizations, locations, emails, files, folders, text documents, presentations, or spreadsheets. Resource discovery is about the identification of resources and the extraction of related metadata in the event stream. It is also about unveiling the resources the user has interacted with, and the resources that are included or referenced in a used resource. We say that a resource is *included* in another one if its content is part of the content of another resource, e.g., when copying some text from an email to a text document. A resource is *referenced* by another one if its location appears in the content of another resource, e.g., when a link to a web page appears in the content of an email. The resources identified by the resource discovery mechanisms are related to instances of the Event concept by the isActionOn objecttype property.

We apply three different techniques to discover resources: regular expressions, information extraction, and direct resource identification. (1) The *regular expression* approach identifies resources in the event stream based on certain character sequences predefined as regular expressions. This is used to identify files, folders, web links or email addresses. (2) The *information extraction* approach extracts person, location, and organization entities in text-based elements in the event stream. This extraction is rule based and utilizes natural language specifics. The extracted entities are mapped to concepts of the UICO, based on the available contextual information. As an example, when the name of a person is identified in a text document, it is mapped to an instance of a Person concept and a relation specifying that this person is mentioned in that document is built. (3) The *direct resource identification* approach finds data about the resource to build directly into the sensor data, and directly maps certain fields of the event stream data to the resource. An example is the ClipboardSnippetResource, which is built from the content of the clipboard application sensed by the clipboard observer. Another example is the sensor data about an email opened by the user. In this case, the sensor sends the information that a specific email identified by its server message id has been opened for reading. Additional metadata about the email is attached by the sensor and added to the discovered resource.

Event to Event-Block Aggregation

Context sensors observe low-level contextual attention metadata that result in simple events. For *logically* aggregating events that belong together (i.e., grouping user's actions) into blocks of events, so-called event-blocks, static rules are used. "Logically" here implies grouping the events that capture the interactions of a user with a single resource. Resources can

be of various types and opened in different applications. Therefore, for different types of applications, different rules are applied in the grouping process. An application can handle multiple resource types, as is the case, e.g., for Microsoft Outlook or Novell GroupWise, in which emails, tasks, notes, appointments, and contact details are managed. The complexity and accuracy of the static rules depend on the application mechanisms for identifying a single resource and on the possibility to capture this resource ID with a sensor. If it is not possible for a sensor to capture a unique resource ID in an application, heuristics are used to uniquely identify the resource.

Two types of rules can be distinguished for the event to event-block aggregating process. The first type is a set of rules designed for specific applications and is referred to as *application-specific rules*. An example of such a rule is: aggregate all events that happened on the same slide in the Microsoft PowerPoint application. The second type of rules, referred to as *default application rules*, is applied if no application-specific rule is applicable. The second type also serves as backup rules when there is not enough information in the event stream to apply an application-specific rule. The goal of these rules is to heuristically aggregate events into event-blocks, based on event attributes that can be observed operating system-wide by the context sensors. These attributes are the *window title* of the application, the process number, and the window handle ID.[10] The window title and the process number perform best for a generic event to event-block aggregation in which no application-specific attribute is present. The discriminative power of the window title has been observed in other work as well (Oliver et al. 2006; Shen, Li, and Dietterich 2007; Granitzer et al. 2008).

Event to Task Aggregation

Aggregating interaction events into tasks cannot be done with the previous rule-based approach, because it would require manually designing rules for all possible tasks. This might be a reasonable approach for well-structured tasks, such as administrative or routine tasks, but is obviously not appropriate for tasks involving a certain freedom and creativity in their execution, i.e., for knowledge-intensive tasks such as "Planning a Journey" or "Writing a Research Paper." A solution is to automatically extract tasks from the information available in the user interaction context model by means of machine-learning techniques. Once detected, these tasks will enrich the ontology model.

USER TASK DETECTION

Here, by task detection we mean *task-class detection,* also referred to as task classification, as opposed to *task-switch detection.* Task classification deals

with the challenge of classifying usage data from user task executions into task classes or task types. Task-switch detection involves predicting when the user switches from one task to another (Oliver et al. 2006; Shen, Li, and Dietterich 2007).

Task Classification Approach

Task detection is classically modeled as a machine-learning problem and more precisely a classification problem. This method is used to recognize web-based tasks (Kellar and Watters 2006; Gutschmidt, Cap, and Nerdinger 2008), tasks within emails (Kushmerick and Lau 2005; Dredze, Lau, and Kushmerick 2006; Shen, Li, and Dietterich 2007) or from the user's complete desktop (Oliver et al. 2006; Shen, Li, and Dietterich 2007; Granitzer et al. 2008). All these approaches are based on the following steps: (1) The contextual data is captured by system and application sensors. (2) Features, i.e., parts of this data, are chosen to build classification training/test instances at the task level: each task represents a training/test instance for a specific class (i.e., a task model) to be learned. (3) To obtain valid inputs for machine-learning algorithms, these features are first subjected to some feature engineering (Witten and Frank 2005), which may include data preprocessing operations, such as removing stopwords (Granitzer et al. 2008) and application-specific terms (Oliver et al. 2006), or constructing word vectors. (4) Feature-value selection (Witten and Frank 2005; Shen, Li, and Dietterich 2007; Granitzer et al. 2008) is (optionally) performed to select the best discriminative feature values. (5) Finally, the classification/learning algorithms are trained/tested on the training/test instances built from the feature values. Having multiple task models results in a *multiclass* classification problem. In this work, we also adopt this classical approach for task detection. We use the machine-learning toolkit Weka (Witten and Frank 2005) for parts of the feature engineering (steps (2) to (4)), and classification (step (5)).

Feature Engineering

Based on the UICO, we have defined 50 features that can be grouped in six categories: ontology structure, content, application, resource, action, and switching sequences. The *ontology structure category* contains features representing the number of instances of concepts and the number of datatype and objecttype relations used per task. The *content category* consists of the content of task-related resources, the content in focus, and the text input of the user. The *application category* contains the classical *window title* feature (Oliver et al. 2006; Shen, Li, and Dietterich 2007; Granitzer et al.

2008), the application name feature (Granitzer et al. 2008) and our newly introduced GUI elements (accessibility objects) features. The *resource category* includes the complete contents and URIs (URLs) (Shen, Li, and Dietterich 2007) of the used, referenced, and included resources, as well as a feature that combines all the metadata about the used resources in a "bag of words". The *action category* represents the user interactions and contains features about the interactions with applications (Granitzer et al. 2008), resource types, resources, key input types (navigational keys, letters, numbers), the number of eventsand event-blocks, the duration of the event-blocks, and the time intervals between event-blocks. The *switching sequences category* comprises features about switches between applications, resources, and event or resource types.

The process of transforming the event attributes associated with our 50 features into feature values that are usable by machine-learning algorithms (step (3) of the task classification approach) is referred to as *feature engineering* (Witten and Frank 2005). The following steps are performed to pre-process the content of text-based features (in this sequence): (1) remove end-of-line characters, (2) remove markups such as \&lg or ! [CDATA, (3) remove all characters but letters, (4) remove German and English stop-words, (5) remove words shorter than three characters. For each text-based feature, we generate values as a vector of words with the StringToWordVec-tor function of Weka. For example, for the *window title* feature, possible generated values are "{*review, Notepad*}," "{*Adobe, Reader, ontology,*}" "{*firefox, google*}," "{*Microsoft, Word, Document*}," etc. For numeric features, we apply the Weka `PKIDiscretize` filter to generate values as intervals rather than numbers.

Evaluation Methodology

Evaluating a task-detection approach is complex and requires assessing the impact of the various factors involved. In this work, we evaluate the influence of the following parameters on the task-detection performance: (1) the set of used features, (2) the number of used values generated from the features and (3) the learning algorithm (classifier). The set of used features is varied by including (1) each feature individually, (2) each feature category individually, (3) all feature categories, or (4) the *top k* best performing single features, with $k \in \{2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20\}$.

To build a task instance, we select the *g* feature values having the highest *Information Gain* (IG) (Witten and Frank 2005), where *g* is varied among 50 different integers distributed between 3 and 10000. Half of these integers are equally distributed between 3 and the number of available feature values, with an upper bound of 5000. The other half is defined by $G = \{3, 5,$

10, 25, 50, 75, 100, 125, 150, 175, 200, 250, 300, 500, 750, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 5000, 7500, 10000}, the upper bound being 10000 or the maximum number of available feature values, whichever is less. We have chosen to use IG to select feature values because it is one of the fastest and most popular methods in the text-classification field. Comparing it with other feature-selection methods is beyond the scope of this article. We use the IG computation functionality of Weka. For a given feature value v and a given set of classes C, we have

$$IG(v) = \sum_{c \in C} P(v, c) \log \frac{P(v, c)}{P(v) \, P(c)}. \tag{1}$$

The classifiers we evaluate are Naïve Bayes (NB), J48 decision tree (J48), k-Nearest Neighbor (KNN-k) with the number of neighbors $k \in \{1, 5, 10, 35\}$, and Linear Support Vector Machine (SVM-c) with cost parameter $c \in \{2^{-5}, 2^{-3}, 2^{-1}, 2^0, 2^1, 2^3, 2^5, 2^8, 2^{10}\}$.[11] The Weka machine learning library (Witten and Frank 2005) and the Weka integration of the libSVM[12] provide the necessary toolkit to evaluate these algorithms. For J48 decision tree, we use the default version of Weka, which performs some pruning (using the subtree raising approach) but no error pruning. To evaluate the learning algorithms, we perform a stratified 10-fold cross-validation (Witten and Frank 2005). Additionally, the training set and test set instances are strictly parted (i.e., constructed and preprocessed independently) to avoid any bias. Finally, we compute the mean values across all folds of the achieved accuracy, micro-precision, and micro-recall.

EXPERIMENTAL RESULTS

Experiment Design

Our experiment was carried out in the knowledge-intensive domain of the Know-Center. It was preceded by an analysis phase, during which several requirements were defined, by interviewing knowledge workers. Users required to know what kind of data was recorded, to be able to access and modify it, and that the evaluation results were anonymized. They could practice with the recording tool for a week before the experiment in order to reduce the unfamiliarity bias. This study was exploratory, the comparison was *within subjects* and the manipulations were achieved by the working environment (laboratory vs. personal workstation) and the executed task (five different tasks).

The first manipulation was achieved by varying the working environments (i.e., the computer desktop environments) of the participants. Each

participant performed the same set of tasks both on a *laboratory computer* on which a set of standard software used in the company had been installed beforehand, and on their company *personal workstations* with their personal computer desktop settings and access to their personal files, folders, bookmarks, emails, and so on. Half of the participants worked first on the laboratory computer and then on their personal workstations, and vice versa for the other half. The assignment of the participants to each group was randomized.

The second manipulation resulted from varying the tasks themselves. During a preliminary meeting, the participants of the experiment agreed on a selection of five tasks typical of the Know-Center domain: (1) "Filling in the Official Journey Form," (2) "Filling in the Cost Recompense Form for the Official Journey," (3) "Creating and Handing in an Application for Leave," (4) "Planning an Official Journey," and (5) "Organizing a Project Meeting." It is worth noting that these tasks present different characteristics in terms of complexity, estimated execution time, number of involved resources, granularity, and so on. A short questionnaire was issued before starting the experiment to make sure that the probands understood the tasks they had to perform, and also to have them think about the tasks before they actually executed them.

The dataset gathered during the experiment contains 220 task instances recorded from 14 users. Each user was supposed to perform all five tasks at least twice in both working environments, which should have produced at least 56 instances for each task class, and a total of at least 280 task instances. However, some instances were lost because of technical problems, and some were simply not performed by the users. The representatives of each task class are almost equally distributed (see Table 2) except for Task 5. Because most users expressed that this task was too difficult to perform on the laboratory computer (due to the lack of personal calendar, files, emails, etc.), we discard it in some parts of the subsequent analysis.

Comparison of Contextual Feature Sets

Because comparing different approaches cannot be done directly because of the difference in the granularity of training instances, we have decided to focus our comparison on the feature engineering part. We compare the performance of various combinations of contextual features in terms of task-detection accuracy. We use features that are specific to our ontology-based approach and more classical features, as used in the Task-Predictor (Shen, Li, and Dietterich 2007), SWISH (Oliver et al. 2006) and Dyonipos (Granitzer et al. 2008) systems. When using features from other approaches, we preprocess them based on the information available

TABLE 2   Distribution of the Task Instances with Respect to the Task Class and the Environment

	Task 1	Task 2	Task 3	Task 4	Task 5
Laboratory computer	30	26	26	24	3
Personal workstations	25	19	25	28	14
Sum	55	45	51	52	17

in the papers mentioned above and evaluate the performance according to our experimental setup.

   We consider all task instances recorded for Tasks 1 to 5, without taking into account the difference in working environment, which corresponds to the number of instances presented in the last row of Table 2. To evaluate the classification results, we apply a stratified 10-fold cross-validation, which varies the partition of the task instances between the training set and the

TABLE 3   Overview of the Best Accuracies *a* Achieved by Various Feature Sets *f*

	f	*l*	*g*	*a*	*p*	*r*
UICO	All Categories	J48	750	86.71	0.96	0.85
	Application Category	J48	50	86.71	0.96	0.84
	Resource Category	NB	3000	66.49	0.89	0.72
	Content Category	NB	4000	65.15	0.87	0.65
	Ontology Structure Category	J48	359	63.38	0.86	0.64
	accessibility object name	J48	50	86.21	0.96	0.86
	window title	J48	75	81.26	0.94	0.81
	accessibility object value	J48	25	71.15	0.90	0.69
	content in focus	J48	1500	64.72	0.87	0.64
	content of event-block	KNN-1	75	63.74	0.87	0.66
	Top 4 features	**J48**	**5000**	**88.55**	**0.97**	**0.87**
	Top 5 features	J48	1000	88.53	0.97	0.87
	Top 2 features	KNN-10	25	88.12	0.96	0.89
	Top 20 features	J48	500	88.07	0.96	0.87
	Top 3 features	J48	3500	87.66	0.96	0.88
	Top 7 features	J48	300	87.19	0.96	0.85
	Top 6 features	J48	250	87.16	0.96	0.86
	Top 8 features	J48	3000	87.14	0.96	0.84
	Top 10 features	J48	300	86.73	0.96	0.85
	Top 15 features	J48	500	86.28	0.96	0.85
	Top 9 features	J48	300	85.76	0.96	0.83
Dyonipos	*ApplicationName (A)*	J48	24	48.20	0.78	0.52
	Content (C)	KNN-1	125	63.38	0.86	0.63
	WindowTitle (W)	NB	100	78.87	0.93	0.80
	A C	J48	150	68.27	0.89	0.69
	A W	J48	100	80.28	0.94	0.80
	C W	NB	750	82.58	0.95	0.85
	A C W	NB	300	83.51	0.95	0.85
SWISH	*W*	J48	150	79.35	0.93	0.78
TaskPredictor	*W & filepath*	J48	1387	79.42	0.94	0.81

The learning algorithm *l*, the number of used features values *g*, the micro-precision *p* and the micro-recall *r* are also given.

test set. Table 3 presents a comparison of the best algorithm runs in terms of task detection accuracy for various features and feature categories. It shows that the best performing feature set is the *top 4 features*, as defined in our UICO approach, namely the *accessibility object name*, the *window title*, the *accessibility object value,* and the *content in focus.* The *window title* is a very classical feature used in all other approaches, but the other features are new and specific to our approach. The paired t-tests we performed show that the *top 4 features* performed statistically significantly better than all other features and feature combinations on a $p < 0.05$ significance level. The results in Table 3 also highlight that the J48 decision tree and the Naïve Bayes algorithms perform much better than the others. From the paired t-tests computed based on the classifiers' achieved accuracy, we can derive the following partial order of these classifiers: $\{J48, NB\} \gg \{KNN\text{-}1,\ KNN\text{-}5,\ KNN\text{-}10,\ KNN\text{-}35\} \gg \{SVM\}$, with $\gg$ indicating a statistical significance on a $\alpha = 0.005$ level.

Investigating Task Detection

We now focus on investigating the following questions: (1) How good can the performance of a task classifier be when used in a real work environment, if it has been being trained with contextual data gathered in laboratory settings? (2) Which are the automatically observable contextual features that allow for good task-detection performance? To carry out our analysis, we consider only the 203 task instances associated with Tasks 1 to 4 (see Table 2). We use the tasks recorded from the laboratory computer as the training set (106 task instances), and those recorded from the personal workstations as the test set (97 task instances). An overview of the results about the performance of detecting real workstation tasks by training on contextual data from laboratory settings is given in Table 4.

The best *feature category* is the application category, which correctly identifies 91.75% of the real tasks (l=NB, g=500, p=0.97, r=0.92). Approximately five points behind in terms of accuracy, is the content category (l=NB, a=86.60%, g=500, p=0.95, r=0.87). Using all 50 features results in a 82.47% accuracy, which is about 9% worse than the best performing feature category. This illustrates that using all available features together does not necessarily provide the best results. This is interesting because it suggests that smaller feature combinations, which are less computationally expensive to deal with, can achieve a better accuracy. This has directed our decision to study single features in greater detail.

After evaluating the performance of each *single feature* separately, we confirm that the *window title* (Oliver et al., 2006; Shen, Li, and Dietterich 2007; Granitzer et al. 2008) is the best discriminative feature, with an accuracy of

TABLE 4 Overview of the Best Accuracies *a* (Ranked Within Each Section)

	f	*l*	*g*	*a*	*p*	*r*	R_G
Feature categories	**Application Category**	**NB**	**500**	**91.75**	**0.97**	**0.92**	**3**
	Content Category	NB	500	86.60	0.95	0.87	5
	All Categories	NB	750	82.47	0.93	0.83	8
	Resource Category	NB	1000	68.04	0.86	0.67	14
	Ontology Structure Category	J48	359	65.98	0.86	0.67	15
	Action Category	SVM-2[5]	11	59.79	0.81	0.58	17
	Switching Sequence Category	NB	1832	40.21	0.66	0.39	20
Single features	window title	J48	100	85.57	0.95	0.87	6
	content in focus	NB	150	84.54	0.94	0.84	7
	accessibility object name	J48	100	80.41	0.92	0.81	9
	event-block content	NB	200	73.20	0.89	0.76	10
	accessibility object value	J48	150	71.13	0.89	0.72	11
	UICO datatype relations	J48	221	70.10	0.88	0.71	12
	used resources metadata	J48	1000	68.04	0.86	0.68	13
	used resources content	NB	175	62.89	0.84	0.65	16
	content of resources	NB	250	58.76	0.81	0.61	18
	accessibility object role	J48	31	54.64	0.79	0.55	19
Top *k*	**k = 6**	**NB**	**250**	**94.85**	**0.98**	**0.95**	**1**
best single	**k = 5**	**NB**	**150**	**92.78**	**0.97**	**0.94**	**2**
features	**k = 4**	NB	500	89.69	0.96	0.91	4

Results achieved while detecting real workstation tasks by training on contextual data from laboratory settings, for each feature category, all feature categories combined, each single feature as well as the *k* top performing single features *f*. The learning algorithm *l*, the number of used feature values *g*, the micro precision *p*, the micro recall *r*, and the global ranking R_G across sections are also given.

85.57% (*l*=J48, *g* = 100, *p* = 0.95, *r* = 0.87). Of great interest are the good performances of our newly introduced accessibility object features: the *accessibility object name* with an 80.41% accuracy (*l*=J48, *g* = 100, *p* = 0.92, *r* = 0.81) and the *accessibility object value* with a 71.13% accuracy (*l*=J48, *g* = 150, *p* = 0.89, *r* = 0.72). Simply counting the number of UICO datatype relations is also quite efficient (*a* = 70.10%). Seeing the good results achieved by the single features individually, we were wondering whether we could do even better by following the simple approach of combining the *k* single features performing best with respect to classification accuracy.

We have studied the performance of the *top k features*, with different values for *k*. With the *top 6* features, including the NB classifier and 250 used feature values, we obtain the highest accuracy (*a* = 94.85%), precision (*p* = 0.98) and recall (*r* = 0.95), among all studied features, feature categories and top *k* feature combinations. This is an accuracy increase of 9.28, a precision increase of 0.03, and a recall increase of 0.08, compared with the performance of the window title feature alone. These top 6 features are: *window title, content in focus, accessibility object name, event-block content, accessibility object value*, and *UICO datatype relations*. The number of used feature values (*g* = 250) of this best-performing approach also supports

prior work showing that a good choice for it is between 200 and 300 feature values (Shen, Li, and Dietterich 2007).

Discussion

Our results give only a first impression of the fact that the working environment in which users perform their tasks has no significant influence on the task-detection performance. Because only four tasks were involved in this analysis, the generalizability of our results is of course limited, and further experiments (with other tasks, other users, and in other domains) are needed. However, it is well recognized that the *window title* feature has a good cross-domain discriminative power, and we think it should be true for other contextual features.

Context detection frameworks, which observe user contextual information, differ in terms of utilized sensors and of granularity of the captured contextual data. Our approach is very fine-granular. We observe not only the content currently viewed by the user or the window title of the application in focus, but also the user's interactions with all desktop elements and application controls (accessibility objects). In our approach, every single interaction of the user with an application and a resource is deemed important, and hence is captured, stored, and analyzed. Using a different context-detection framework could result in leaving out contextual features having a good discriminative power, and could hence have a negative impact on the task-detection performance.

On the other hand, the strong positive influence of specific context features on task-detection performance is an indication that it may not be necessary to track all the user's interactions with the computer desktop, but only the most relevant elements. This obviously has an impact on what kind of sensors have to be developed, i.e., which context features have to be sensed, in order to achieve a reasonable task-detection performance. It would also impact the user's system performance because capturing less data should lead to fewer CPU requirements. Furthermore, if we know which features are performing well for supervised machine-learning algorithms in laboratory settings, it could provide a first indication with regard to which features could be used in an unsupervised learning approach and in real-world settings. However, this requires further experiments in laboratory and real-world settings.

CONCLUSION

We have studied the question of automatically detecting the task a user is performing on a computer desktop. We have introduced an

ontology-based user interaction context model (UICO) that extends the spectrum of features that can be used for task detection. Based on these novel features, we were able to provide a combination of features that outperforms other feature sets, especially those including only classical features. This result has been obtained on the dataset collected from a large user study carried out in a knowledge-intensive business environment. Our experiment has also shown that it is possible to obtain good task-detection results for real user tasks with a classifier trained offline on laboratory contextual data. We have also studied the discriminative power of individual and combined contextual features. The good performance of the classical *window title* feature has been confirmed and even significantly outperformed by a specific combination of six features. Within this combination are contextual features that are specific to our UICO approach.

The analysis presented in this paper was limited to very classical machine-learning algorithms. This was a reasonable first step, as similar analysis reported in the literature use the same methods. To extend our study, we now plan to use more state-of-the-art algorithms for classification and feature selection. For example, it could be interesting to use multiclass polynomial SVMs, because they allow us to model feature conjunctions and dependencies. Also, instead of using the IG method for feature selection, we could use more sophisticated, multivariate methods, such as the Recursive Feature Elimination method.

We plan to study the discriminative power of our ontology-specific contextual features more thoroughly by performing further experiments with tasks having different characteristics, because the results we have obtained here are perhaps domain-specific. We have already started to investigate this point. We have performed two similar experiments in another domain, and the results obtained confirm what is reported here (Rath et al. 2010). In general, we would like to understand why some features perform better than others for task detection. Our objective is to exhibit a small combination of contextual features with a strong discriminative power, independently of the domain, in order to enhance automatic task detection performance. More generally, since the number of controlled user studies in the task detection area is low, we plan to perform further ones to get a deeper insight on which kind of tasks can be automatically detected and in which settings. As an example, we have recently shown that knowledge-intensive tasks can be detected as well as routine tasks (Rath, Devaurs, and Lindstaedt 2010).

This work is integrated in our Knowledge Services framework, *KnowSe*, which strives to provide highly contextualized and personalized knowledge services to the user. In addition to providing a user task-detection module, KnowSe can also perform *information need discovery*. Other services focus on

context-aware information retrieval and *proactive context-aware information delivery*, which involve spreading activation on the graph-based representation of the user context model, and a ranking of search results based on resource usage and interconnectivity. Among the services are also tools developed for *visualizing* the individual and organizational context of the user (graph views, time lines, and self-organizing map views).

NOTES

1. This paper is a synthesis of our work on the topic and is based on already published results (Rath et al. 2009a,b,c,d).
2. Involving 14 users, this study is much larger than previous studies reported in the field which involve only a few users (Oliver et al. 2006; Shen, Li, and Dietterich 2007; Granitzer et al. 2008).
3. http://www.w3.org/2004/OWL
4. http://protege.stanford.edu
5. The user dimension should not be mistaken for a user profile containing preferences, etc. Maintaining such a profile is out of the scope of this work.
6. Microsoft Active Accessibility: http://msdn.microsoft.com/en-us/accessibility.
7. http://nepomuk.semanticdesktop.org
8. http://usercontext.opendfki.de/wiki/NopOntology
9. We store the recorded contextual data in a triple store, and more precisely a quad store, featuring named graphs and SPARQL query possibilities.
10. The window handle is a unique window identifier, constructed by the Microsoft Windows operating system.
11. These values are borrowed from the libSVM practical guide: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
12. http://www.cs.iastate.edu/~yasser/wlsvm/

REFERENCES

Baldauf, M., S. Dustdar, and F. Rosenberg. 2007. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing* 2 (4): 263–277.

Biedert, R., S. Schwarz, and T. R. Roth-Berghofer. 2008. Designing a context-sensitive dashboard for an adaptive knowledge worker assistant. In *Proceeding of workshop on modeling and reasoning in context, Human Centered Processes (HCP) '08.*

Callan, J., J. Allan, C. L. A. Clarke, S. Dumais, D. A. Evans, M. Sanderson, and C. Zhai. 2007. Meeting of the MINDS: An information retrieval research agenda. *ACM SIGIR Forum* 41 (2): 25–34.

Catarci, T., A. Dix, A. Katifori, G. Lepouras, and A. Poggi. 2007. Task-centered information management. In *Proceedings of DELOS Conference on Digital Libraries '07*, 253–263.

Chernov, S., G. Demartini, E. Herder, M. Kopycki, and W. Nejdl. 2008. Evaluating personal information management using an activity logs enriched desktop dataset. In *Proceedings of the workshop on personal information management, CHI '08.*

Coutaz, J., J. L. Crowley, S. Dobson, and D. Garlan. 2005. Context is key. *Communications of the ACM* 48 (3): 49–53.

Dey, A. K., G. D. Abowd, and D. Salber. 2001. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human Computer Interaction* 16 (2): 97–166.

Dragunov, A. N., T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, and J. L. Herlocker. 2005. Task-Tracer: A desktop environment to support multi-tasking knowledge workers. In *Proceedings of the international conference on intelligent user interfaces '05*, 75–82.

Dredze, M., T. Lau, and N. Kushmerick. 2006. Automatically classifying emails into activities. In *Proceedings of the international conference on intelligent user interfaces '06*, 70–77.

Granitzer, M., M. Kröll, C. Seifert, A. S. Rath, N. Weber, O. Dietzel, and S. N. Lindstaedt. 2008. Analysis of machine learning techniques for context extraction. In *International conference on digital information management '08*, 233–240.

Groza, T., S. Handschuh, K. Möller, G. Grimnes, L. Sauermann, E. Minack, C. Mesnage, M. Jazayeri, G. Reif, and R. Gudjónsdóttir. 2007. The NEPOMUK project–on the way to the social semantic desktop. In *Proceedings of I-SEMANTICS '07*, 201–211.

Gutschmidt, A., C. H. Cap, and F. W. Nerdinger. 2008. Paving the path to automatic user task identification. In *Proceedings of the workshop on common sense knowledge and goal-oriented interfaces, intelligent user interfaces '08*.

Jones, W., P. Klasnja, A. Civan, and M. L. Adcock. 2008. The personal project planner: Planning to organize personal information. In *Proceedings of CHI '08*, 681–684.

Kellar, M., and C. Watters. 2006. Using web browser interactions to predict task. In *WWW '06 – Poster*.

Kushmerick, N., and T. Lau. 2005. Automated email activity management: An unsupervised learning approach. In *Proceedings of the international conference on intelligent user interfaces '05*, 67–74.

Mylonas, P., D. Vallet, P. Castells, M. Fernandez, and Y. Avrithis. 2008. Personalized information retrieval based on context and ontological knowledge. *Knowledge Engineering Review* 23 (1): 73–100.

Oliver, N., G. Smith, C. Thakkar, and A. C. Surendran. 2006. SWISH: Semantic analysis of window titles and switching history. In *Proceedings of the international conference on intelligent user interfaces '06*, 194–201.

Rath, A. S., D. Devaurs, and S. N. Lindstaedt. 2009a. Contextualized knowledge services for personalized learner support. In *Proceedings of the Demonstrations European conference on technology enhanced learning '09*.

Rath, A. S., D. Devaurs, and S. N. Lindstaedt. 2009b. Detecting real user tasks by training on laboratory contextual attention metadata. In *Proceedings of the workshop on exploitation of usage and attention metadata, informatik '09*.

Rath, A. S., D. Devaurs, and S. N. Lindstaedt. 2009c. KnowSe: Fostering user interaction context awareness. In *Proceedings of the Demonstrations of European conference on computer supported cooperative work '09*.

Rath, A. S., D. Devaurs, and S. N. Lindstaedt. 2009d. UICO: An ontology-based user interaction context model for automatic task detection on the computer desktop. In *Proceedings of the workshop on context information and ontology, European semantic web conference '09*.

Rath, A. S., D. Devaurs, and S. N. Lindstaedt. 2010. Studying the factors influencing automatic user task detection on the computer desktop. In *Proceedings of the European conference on technology enhanced learning '10*, 292–307.

Sauermann, L., A. Bernardi, and A. Dengel. 2005. Overview and outlook on the semantic desktop. In *Proceedings of the workshop on the semantic desktop, international semantic web conference '05*.

Sauermann, L., L. van Elst, and A. Dengel. 2007. PIMO – a framework for representing personal information models. In *Proceedings of I-SEMANTICS '07*, 270–277.

Schmidt, A. 2005. Bridging the gap between knowledge management and e-learning with context-aware corporate learning solutions. In *Proceedings of WM Conference on Professional Knowledge Management '05*, 203–213.

Shen, J., L. Li, and T. G. Dietterich. 2007. Real-time detection of task switches of desktop users. In *Proceedings of the international joint conference on artificial intelligence '07*, 2868–2873.

Strang, T., and C. Linnhoff-Popien. 2004. A context modeling survey. In *Proceedings of the workshop on advanced context modelling, reasoning and management, UbiComp '04*.

Tang, J. C., J. Lin, J. Pierce, S. Whittaker, and C. Drews. 2007. Recent shortcuts: Using recent interactions to support shared activities. In *Proceedings of CHI '07*, 1263–1272.

Van Kleek, M., and H. E. Shrobe. 2007. A practical activity capture framework for personal, lifetime user modeling. In *Proceedings of the UM User Modeling Conference '07 – Poster*, 298–302.

Witten, I. H., and E. Frank. 2005. *Data mining: practical machine learning tools and techniques*. San Francisco, CA, USA: Morgan-Kaufmann.

Wolpers, M., J. Najjar, K. Verbert, and E. Duval. 2007. Tracking actual usage: The attention metadata approach. *Educational Technology & Society* 10 (3): 106–121.

Xiao, H., and I. F. Cruz. 2005. A multi-ontology approach for personal information management. In *Proceedings of the workshop on the semantic desktop, international semantic web conference '05*.