Hierarchical Task Instance Mining in Interaction Histories

Benedikt Schmidt, Johannes Kastl, Todor Stoitsev
SAP Research Darmstadt
62483 Darmstadt, Germany
firstname.lastname@sap.com

Max Mühläuser
Telecooperation Group, Darmstadt University of Technology
64289 Darmstadt, Germany
max@informatik.tu-darmstadt.de

ABSTRACT

Knowledge work at computer workplaces involves execution of multiple concurrent tasks with frequent task interruptions. The complexity of the resulting work processes makes task externalization a desired goal towards facilitating analysis and support of knowledge work, e.g. by extracting and disseminating best practices.

In this paper, we present a task mining method that identifies tasks based on interaction histories. The method generates instances of a semantic hierarchical task model which captures an abstraction of the work processes. A specific characteristic of the method is that it mines tasks based on a combination of semantic and temporal features, extracted from enriched interaction histories. The use of semantic similarity results in a high robustness of the system with respect to task interruption and concurrent task execution. An evaluation of our task mining method based on a study with users executing frequently interrupted tasks is presented. One element of the evaluation is the assessment of different algorithms for semantic similarity computing, namely Term Matching (TM), Vector Space Model (VSM) and Latent Dirichlet Allocation (LDA). For an approach using VSM a precision of 0.83, a recall of 0.76 and a F1-measure of 0.79 is reached.

Categories and Subject Descriptors

H.4.1 [**Information Systems Applications**]: Office Automation

General Terms

Human Factors

Keywords

human-computer interaction, task execution support, context, knowledge work support

SIGDOC'11, October 3–5, 2011, Pisa, Italy.

1. INTRODUCTION

The externalization of knowledge work execution is an efficient foundation for various knowledge work support scenarios such as the mitigation of prospective memory failures [9, 18], the automation of recurring work, improved information access/organization [13], and knowledge dissemination [21]. User interaction histories are frequently used for such externalization [13, 18, 17]. These histories are temporally ordered lists of classified events that stand for user-system interactions, and generally comprise complex mixtures of information about resources a user operated on, using different applications. The complexity of these histories results from the fact that knowledge work in organizations includes frequent disruptions and task-switches [9] that veil the structure of work execution, i.e. it remains unclear which acts serve similar goals and are constrained by situational requirements or individual competencies.

Task mining is the identification of tasks as logical units of work in interaction histories resulting from externalization of work execution. This paper contributes to the research on unsupervised task mining based on interaction histories (c.f. [18, 17, 5]). Our mining method emphasizes robustness of the system with respect to task switches and a process representation of the mined task instances. Task switch robustness is the result of a clustering of interaction history elements based on semantic similarity, not mainly on temporal relations. The process representation follows a Semantic Hierarchical Task Model (SHTM) that presents work execution processes as action and operation hierarchy, following Activity Theory (AT). SHTM enables the mining of abstract process representations of tasks.

The remainder of this paper is organized as follows. First, background on task execution is given. Second, a SHTM to capture task execution processes is presented. Then, a task mining process to generate task model instances is introduced and the task mining is evaluated by a user study. Finally, related work is discussed and the conclusion is given.

2. BACKGROUND

A decent understanding of knowledge work execution at computer workplaces, their formation and externalization opportunities is essential for task mining. Task execution at computer workplaces generally addresses different concurrent goals, resulting in parallel or rapid succession of task executions. Task interruptions occur frequently and stimulate task switches. Studies report 50 task switches over a single week [9]. Thereby, execution processes for similar tasks are not uniform, but emerge in an individual, intentional prob-

lem solving process to reach a goal in reaction to situational dependencies [16, 25, 23, 6]. One can distinguish an external and a subjective perspective on the execution process. From an external perspective, a work process manifests as a series of objective observable acts. For a subject, the work process is a dynamic sequence of hierarchically related goal episodes which involve subgoals and the realization by acts [16, 25].

The computer workplace is an environment for work execution, that shapes work execution through its intrinsic design, i.e. supported interactions and available tools. Nevertheless, the mediation by the computer is given implicitly as the computer as tool has no prominent influence on the work. Individuals plan and perceive their interaction with the computer in terms of goals with accompanying tasks rather than in terms of system commands [2]. The technology disappears although it shapes the individual interaction. From this perspective, two characteristics of computer workplaces are striking:

- *Relevance of text:* Computer work boils down to the creation, consumption and transformation of encoded signs that transfer information. The importance of the relationship between text and activity with respect to Vygotsky, Leonitev, Bakhtin and Burke's idea of "Language as Symbolic Action" has been highlighted in [7].
- *Highly structured environment:* Regular computer workplaces include a set of standard software that follows common interaction paradigms. The standardized interaction is familiar to individuals and performed without cognitive efforts i.e. in the form of operations.

The characteristics of the computer workplace shape and structure the work process. Actions and operations of knowledge workers at computer workplaces are not directly observable, as an externalization by an interaction history only includes acts as event representation. Nevertheless, it is beneficial to abstract from observable acts and describe knowledge work processes by means of actions and operations. Such abstraction unfolds the solution processes which structure the work processes of individuals.

3. SEMANTIC HIERARCHICAL TASK MODEL (SHTM)

The SHTM is a task model for work execution at computer workplaces that follows the terminology of activity theory that decomposes activities into actions and operations [14]. This understanding is explained in detail in [20]. SHTM stipulates the transition from objectively observable acts at computer workplaces to the hierarchical structure of operations and actions. The model supports the process of task extraction from interaction histories. SHTM is a semantic model, as it includes information about user-system interaction that enriches information included in interaction histories and enables abstraction from single interaction events to abstract representations of work episodes. For example, for an authoring process with different applications the model does not suggest representing a set of application switches but extraction of the interactions of the different used application and the purpose of the interactions.

SHTM decomposes a task t into knowledge-intensive actions, termed knowledge actions in the following and represented by the attribute $knowledgeActions : KnowledgeActionList$ of t. Knowledge actions themselves are decomposed into operations at the computer desktop, termed desktop operations in the following and represented by the attribute $desktopOperations : DesktopOperationList$ for a knowledge action k. By organizing tasks based on knowledge actions, an abstraction from the actual execution process is realized that describes tasks in terms of aggregated and classified execution process fragments. The connection between the different concepts is visible in figure 1 and is discussed in the following.

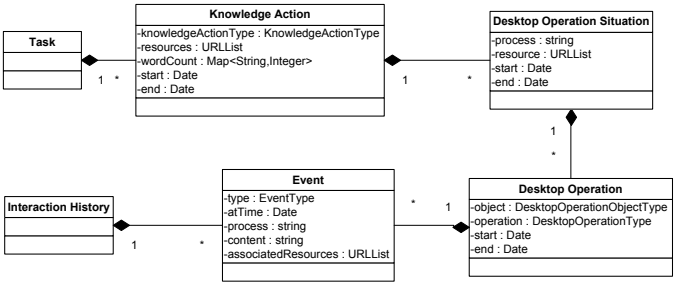

Figure 1: Main elements of SHTM

3.1 Desktop Operations

A desktop operation is a user-system interaction that follows standard interaction-patterns. These paradigms are well trained by individuals and can be applied without much cognitive effort. In terms of AT, a user-system interactions at computer workplaces is situated on the operation level.

A desktop operation d is specified by a tuple of operation and object, represented by the attributes $operation : DesktopOperationType$ and $object : DesktopObjectType$. A desktop operation references events that are the objective indicators of the desktop operation (e.g. a set of events in an interaction history) by the attribute $event : EventList$. Additional information is the $start : Date$ and $end : Date$ of a desktop operation. The following objects are considered for desktop operations:

- Application: An application is a piece of software that can be run in the computer environment and is used to transform information. An operating system process is a running instance of an application.
- File: A file is a container structure for information. Files have an address and permission structures to organizes access and modification.
- Folder: A folder is a container structure to organize files. Generally, folders have an address, encapsulate an arbitrary number of files and are described by a label.
- Information Object: Information objects are pieces of information presented to the user. This includes, for example, textual information represented by a string of characters. Information objects often have a structure to make them accessible by functionalities of software system. One method to persist information objects are files.

Opr \ Obj	App	File	Folder	Information Object	Window
Open	x	x	x		
Close	x	x	x		
Save		x			
Rename		x	x		
Delete		x	x	x	
Cut		x	x	x	
Paste		x	x	x	
Print		x			
Create		x	x	x	
Execute	x				
Focus			x		x

Table 1: Desktop operations: pairs of operation (OPR) and object (OBJ)

Possible associations of desktop operation types and desktop object types are given in table 1.

Desktop operations that are executed in sequence for the same resource using the same application are combined to desktop operation situation. I.e. a desktop operation situation is considered as a continuous work episode with a single application on a specific resource. A desktop operation situation *dos* has the attributes $process : String$, $resource : URL$, $start : Date$, $end : Date$ and $desktopOperations : DesktopOperationList$.

3.2 Knowledge Actions

Knowledge actions are techniques applied to execute working tasks. These techniques are tangible subgoals that support an individual in structuring a task execution process. An example is the goal of writing a document. An individual structures the respective working process by subgoals that can be directly translated to an interaction with a tool, e.g. using the knowledge action *authoring*. Still, the *authoring* is not straight forward but needs adaptation for the specific task and the specific working situation. Thus, knowledge actions accomplish the transition from planning to actual work execution combined with a process of situative adaptation. At the computer workplace knowledge actions are executed by means of desktop operations. As the adaptation of the technique to an explicit goal requires cognitive efforts, the described techniques are situated on the action level of AT (see also [11]).

For the computer workplace a set of knowledge actions has been identified based on a literature review [11, 24] and discussions. The SHTM includes the following knowledge actions:

- Consuming: A knowledge worker focuses a resource on the computer desktop and focuses his attention to the visual representation of an information.
- Authoring: A knowledge worker creates a representation of information in an existing or new information object through a knowledge transformation process.
- Communicating: A knowledge worker shares information with others.
- Organizing: A knowledge worker organizes existing information resources.
- Browsing: A knowledge worker reviews different information objects in rapid succession.

A knowledge action k is modeled as a classified carrier for desktop operations. k is classified by the attribute $action : KnowledgeActionType$. A knowledge action has the attribute $situations : DesktopOperationSituationList$. To specify knowledge actions, they additionally contain the attributes $resources : URLList$ for all resources required for the execution and $wordCount : Map < String, Integer >$ for an abstract representation of the textual content of the knowledge action (what it is about). The duration is captured by $start : Date$ and $end : Date$.

4. TASK MINING USING A SEMANTIC TASK MODEL

The following section describes task mining as extraction of SHTM instances based on ex post analysis of interaction histories. Basically, the mining process presents itself as connection of the user-interaction events contained in interaction histories with the SHTM in a bottom-up process of data aggregation as depicted in figure 2. Interaction history events are aggregated to desktop operations which themselves are combined to knowledge actions which finally enable the identification of tasks. The task mining method reflects concurrent task executions and task switches, as it ignores temporal information, but focuses completely on semantic similarity of enriched textual content included in the interaction histories.

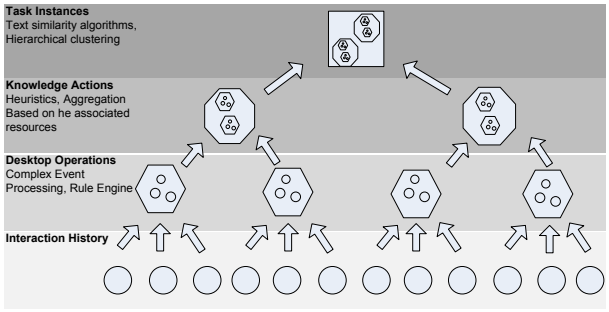

Figure 2: Hierarchy of Behavior Situatedness in Activity Theory

4.1 Interaction History Creation

Interaction histories can be basically considered as logs of user-system interactions. The literature describes different applications to generate this type of data [18, 13]. Interaction history data gives an insight into work as temporally ordered information about user-system interaction. The approach presented in this paper requires interaction histories that extract text displayed to the user and include information about the time frame during which the text was visible. An event e has the attributes $name : String$ to classify an event, $atTime : Date$ to specify the occurrence, $content : String$ for the textual content, $process : String$ for the application generating the event and optionally $associatedResource : URLList$ for resources that are locked by the process

To create interaction histories, a sensor application has been developed for the Windows operating system (see figure 3 for an excerpt from an interaction history). The sensor application has been developed in C# and generates a sensor stream that can be subscribed by other applications via a CORBA interface. The monitor includes a keyboard hub, mouse hub and listeners to different applications (includes: Microsoft Business Office Suite, web browser, Adobe Reader, Process Monitor, Mouse and Keyboard Hub). Additionally, the UI Automation Framework is used to generate rich information about each focused user interface element. Filter methods like black lists and comparison of temporally related events are applied to optimize the quality of the interaction history, as the system events often send events twice or internal methods may fail. The sensor application realizes events as aforementioned. For technical reasons attributes not mentioned here are included additionally in the actual implementation. The sensor application generates an interaction history as a stream of temporal ordered events $e_1...e_n$.

4.2 Desktop Operation Extraction

The SHTM describes desktop operations d as classified by the attributes $d.operation$ and $d.object$. The events delivered by the sensor application are basic events and need to be processed to identify desktop operations. Mapping events to desktop operations is a process of abstraction, as different interaction types are traced back to the basic terminology of desktop operations. This desktop operation identification is identified through complex event processing: rules are modeled to create desktop operations based on sensor events. Extraction of desktop operations requires basic rules RL of the form $antecedent \Rightarrow consequent$. The $antecedent$ relates to the attributes of the events e. The $consequent$ is a new desktop operation d. The aforementioned stream of temporal ordered events $e_1...e_n$ is processed by the rules $RL_1...RL_n$ which results in the creation of new desktop operations $d_0...d_n$.

A realization of the concept has been implemented using Drools Fusion (JBoss Drools, http://jboss.org/drools) which is capable of event processing and temporal reasoning. In total 98 different rules were developed to extract desktop operations. The amount shows the complexity of the abstraction process, as a large set of interaction types needs to be traced back to the respective desktop operations (e.g. rules for 15 different types of file closing were modeled).

4.3 Knowledge Action Extraction

Knowledge actions as work techniques present themselves as disjoint units of continuous work on a single resource, i.e. desktop operations that stand for the same knowledge action are scattered among the interaction history due to disruptions and a mixture of subgoals executed in parallel. Consequently, knowledge action identification turns out as collection and classification of scattered desktop operations belonging to the same knowledge action. The respective method steps are described in the following:

Create situations: Desktop operation situations dos are used to organize desktop operations. Two desktop operations d_i and d_j get associated to a situation dos, if $d_i.process = d_j.process$ and no d_k exists with $d_k.process \neq d_{i/j}.process$ and $d_i.atTime < d_k.atTime < d_k.atTime$. A list of desktop operation situations $dos_1...dos_n$ is created.

Create unclassified knowledge actions: Sets of desktop operation situations are created. dos_i and dos_j are included in the same set, if $dos_i.process = dos_j.process$ and $dos_i.resource = dos_j.resource$ is valid. This connection to situations as sets of desktop operations based on process and resource is visible in figure 4. Each set of situations represents an unclassified knowledge action k without specification of $k.action$. Knowledge actions, which have a duration of less than six seconds are filtered as irrelevant which is based on detailed review of interaction histories and comparable to [18] where durations below four seconds are filtered.

Classify knowledge actions: For the detected k the type $k.action$ is identified based on heuristics that make use of an application taxonomy. Each unclassified k contains the desktop operations ($k.situations.desktopOperations$) executed at $k.process$. The heuristics decide for a knowledge action based on the types of desktop operations (e.g. consume, focus) and the classification of $k.process$ in the application taxonomy. For example, the usage of Microsoft Word is generalized to the usage of a word processor. Using this information, one can classify the knowledge actions based on the desktop operations they contain. The desktop operation "Creating"-"Information Object", for example, indicates a knowledge action of the type "Authoring". "Renaming"-"Folder" indicates the knowledge action "Organizing".

Content extraction For each classified knowledge action, the map $k.wordCount$ is extracted. Therefore, the $e.content$ attribute of each event contained in $k.situations.desktop\ Operations.events$ is subject to tokenization, stop-word-detection, part-of-speech tagging, filtering, stemming and term counting to create a bag-of-words. The bag-of-words contains all terms considered relevant (nouns, verbs) and the number of times they occur. The content extraction has been implemented, using a UIMA [10] pipeline of processing resources.

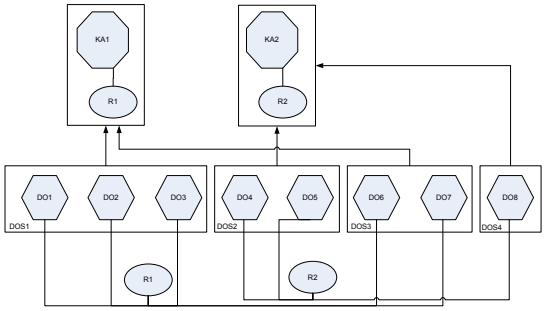

Figure 4: Aggregation of desktop operations to knowledge actions (R=Resource, DO=Desktop Operation, DOS= Desktop Operation Situation, KA=Knowledge Action)

4.4 Task Instance Mining

Task instance mining uses knowledge actions $k_0...k_n$ as generated by the aforementioned method. Each knowledge action stands for a work technique applied at a similar resource with a similar application. Task instances are collections of knowledge actions that serve a similar goal, with other words related knowledge actions.

Here, the term related precisely addresses semantic relatedness. Based on $k.wordCount$ text similarity algorithms

<event eventName="FOREGROUND_WINDOW_CHANGED" atTime="01.02.2011 16:19:26.484" eventCategory="Process"><eventattributes>
<eventattribute name="processname" type="String" value="POWERPNT" />
<eventattribute name="windowtitle" type="String" value="Microsoft PowerPoint - [Planning-Roundup_v3.pptx]" />
<eventattribute name="associatedFile" type="String" value="\BaseNamedObjects\MSCTF.Shared.SFM.AFDB" />
<eventattribute name="associatedFile" type="String" value="C:\Documents and Settings\evaluationuser3\Desktop\Work\Planning-
Roundup_v3.pptx" /></eventattributes></event>
<event eventName="FILESYSTEM_OBJECT_DELETED" atTime="01.02.2011 16:19:26.828" eventCategory="Filesystem"><eventattributes>
<eventattribute name="name" type="String" value="C:\Documents and Settings\evaluationuser3\Desktop\Work\Planning-
Roundup_v3.pptx" />
<eventattribute name="notifyfilter" type="String" value="FileName" /></eventattributes></event>
<event eventName="FILESYSTEM_OBJECT_CREATED" atTime="01.02.2011 16:19:26.828" eventCategory="Filesystem"><eventattributes>
<eventattribute name="name" type="String" value="C:\Documents and Settings\evaluationuser3\Desktop\Work\Planning-
Roundup_v3.pptx" />
<eventattribute name="notifyfilter" type="String" value="FileName" /></eventattributes></event>
<event eventName="MSPOWERPOINT_EVENT_THROWN" atTime="01.02.2011 16:19:26.968" eventCategory="Application"><eventattributes>
<eventattribute name="title" type="String" value="Planning-Roundup_v3.pptx" />
<eventattribute name="content" type="String" value=" Planning Demand and Master Planning Roundup Introduction Author: Andreas
Goeb
Date: 2011/02/01 Demand Planning Purpose
improve decisions affecting demand accuracy
calculation of buffer or
safety stocks
Results:
Demand Plan
Benefit:
Increased performance of each supply chain entity Master Planning
Purpose
synchronize the flow of materials along the supply chain
Range: Mid-term At least one seasonal
cycle
Contents:
Production
Transport
supply capacities
Seasonal stock
balancing of supply and

Figure 3: Excerpt from interaction history

generate input values for a clustering algorithm (used algorithms are given below).

Different methods have been evaluated to calculate semantic relatedness by using a bag-of-words approach. Based on calculations applied to the bag-of-words, the similarity of two bags-of-words is calculated, returning a similarity value between zero and one. An important modification to the bag-of-words was applied here to use the time as relevance factor, i.e. text presented for a longer period of time to a user was considered more important than text presented for a short period of time. Therefore the duration of a knowledge action k was used to weight $k.wordCount$. Three methods to identify semantic relatedness have been applied and compared:

- Term Matching (TM): The number of words that occur in both texts is calculated and scaled by the lengths of both texts (total number of words) [3]. Here, TM is considered as a baseline method.
- Vector Space Model (VSM): The Vector Space Model is an algebraic model to represent text documents [19]. Every text is represented as a term-TF*IDF vector in the N-dimensional space (N representing the number of different terms in both documents). Text similarity is measured by the distance of the vectors within the model.
- Latent Dirichlet Allocation: Latent Dirichlet Allocation is a generative model which regards each text as a mixture of topics and traces each word's creation to one of the text's topics [4]. The model can be applied to realize topic detection and map each text to a probability distribution over the detected topics. The "distance" of two probability distributions can be obtained by utilizing a suitable divergence measure.

The resulting semantic relatedness is weighted based on the temporal closeness of two knowledge actions. The time acts as a gravity for semantic. The technique follows the idea that proximity (spatial or temporal) influences semantic similarity: e.g. homonyms are understood based on a context organized based on spatial and temporal proximity (asking someone for a bank during a finance conversation will most probably not result in a hint for a place to sit down). Therefore the semantic relatedness is weighted by a factor between zero and one that can be calculated by a sigmoid function on all temporal distances between knowledge actions. The weighted semantic relatedness is input for a hierarchical clustering algorithm [12]. Hierarchical clustering is an unsupervised algorithm which builds a hierarchy of clusters, given a set of input data, an appropriate distance metric, and a linkage criterion. The linkage criterion determines, how the distance between clusters is computed. An average linkage clustering was used, which means that the distance between two clusters is computed by the average distance of all elements within one cluster to all elements within the other cluster. Within the task mining module, an agglomerative variant of hierarchical clustering was implemented, i.e. a "bottom up" approach is used. At the beginning, each knowledge action belongs to its own cluster. Then the algorithm finds the pair of clusters which has the highest similarity. Those clusters are merged into a new cluster and a new level of the hierarchy is created. The algorithm repeats this step until only one cluster remains and the hierarchy is complete. In order to obtain clusters of knowledge actions which are similar to each other and represent a task, a threshold which terminates the algorithm is required.

The last step identifies clusters that belong to one task, even if they are not semantically connected. If a user tended to switch very often between two clusters, i.e. used the applications and resources of both clusters simultaneously or in rapid succession, then the clusters can be merged. For this purpose a matrix that counts all cluster switches is calculated and the clusters are merged, if one of the clusters does not contain more elements than specified by a treshold. As long as clusters are merged, process is repeated.

5. EVALUATION

The following section reports on a study that addresses important aspects of the task mining approach described above. The first aspect is the overall applicability of the approach to task mining in interaction histories. The second aspect is the effect of different methods of semantic similarity calculation (Comparison of TM, VSM and LDA).

The study included eight participants that work for an international software company in the field of IT research. The participants executed a set of predefined, knowledge-intensive tasks (see table 2). Five participants had post-doctoral positions and three participants were PhD students. The tasks were executed in random order and were disrupted during execution. Disruption means that tasks were interrupted randomly to generate task switches as shown in figure

Task 1	Provide information on related work on individual topic
Task 2	Set up meeting to discuss conference paper review
Task 3	Decide on applicant invitation and communicate your decision
Task 4	Plan a trip and inform your colleague with all involved information
Task 5	Present a paper from a foreign language to your colleagues
Task 6	Find Application partners and experts for research project
Task 7	Search for Information on software functionality and save for later use

Table 2: Tasks used for the user study

5. During the execution of the tasks, an interaction history was captured, using the sensor application. The created interaction histories were used as input to the task mining method discussed in the previous section.

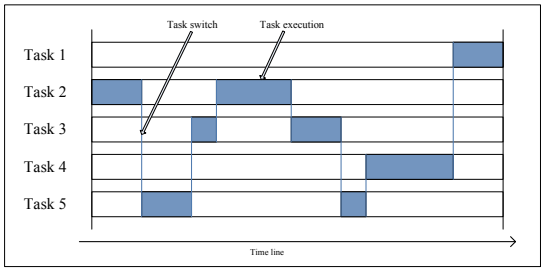

Figure 5: Example for task execution process with task switches

5.1 Evaluation Methodology

The interaction history of each user including the execution process of five frequently switched tasks was input for the task mining method. The mined task instances were assessed against a manually created ground truth. Task instance and ground truth can be considered as clusters of knowledge actions. Input for the creation of a ground truth and task mining were knowledge actions that were extracted from interaction histories based on the aforementioned processes of desktop operation extraction and knowledge action extraction. For these knowledge actions the following process was performed:

- **Ground truth generation:** The study supervisors used interaction logs and notes taken during study execution to validate the quality of knowledge actions and create clusters of knowledge actions that were labeled with the respective task numbers. Thus, the ground truth assigns a task number to each knowledge action extracted from an interaction history.

- **Task mining:** Semantic similarity is the most important aspect of the task mining method. To evaluate the performance of different textual similarity measures, three different cluster distributions (TM, VSM, LDA) were produced and used as input for the hierarchical clustering algorithm. The algorithm requires a threshold as termination criterion. Tresholds for TM, VSM and LDA were identified in a different study. For the other study, clustering results for input data of interaction histories generated by 90 minutes work executed by 20 users was analyzed to identify optimal treshold values (VSM=0.15, LDA=0.9, TM=0.05).

- **Labeling:** The mined clusters should be similar to the manually labeled clusters of the ground truth, i.e. it should be the same number of clusters, containing the same knowledge actions. In order to compare the knowledge action clusters of the ground truth with the corresponding clusters identified by the system, the following labeling method was applied for each identified cluster: 1) Select a cluster from the task mining as *tolabel* 2) Select the ground truth cluster with the largest percentage of knowledge actions matching the selected cluster as *compareCluster* 3) Label the *tolabel* cluster with the label of *compareCluster* (cf. [5, 17]).

- **Assessment:** A high quality is reached, if a mined task shares many knowledge actions with the corresponding cluster of the ground truth. Three quantitative measures were extracted: 1) Precision: The fraction of knowledge actions in a mined cluster compared to the corresponding manually labeled cluster of the ground truth. The corresponding cluster is the one with the largest matching percentage of items. 2) Recall: The fraction of all knowledge actions in a manually labeled cluster corresponding to a calculated cluster. 3) F1-measure: The weighted harmonic mean of precision and recall. The higher the value of the F1-measure generally the better the result of the algorithm [15].

5.2 Evaluation Results

In total 120 runs of the hierarchical clustering algorithm were analyzed. Table 3 indicates the obtained results. The results for VSM clearly surpass those of both LDA and TM. The low values for TM are expected as it serves as a base line method. The values for the clustering based on LDA are higher than the base line. However, the achieved F1-measure of 0.59 is clearly outperformed by the clustering based on VSM with a F1-measure of 0.79. This result can be partly explained by the amount of input which is used to perform LDA. Only the data, that was collected from an independent run of the task detection system, was used for the similarity calculation step. This is the setup for each similarity algorithm. LDA should produce better results if the complete data of all participants is used to build one single topic model. With an increasing amount of data available, the quality of the inferred topic model will increase [4]. This should be the focus of further investigation regarding the applicability of topic models for task similarity. In the current setup the clustering based on the VSM delivers the best results.

On the total data set average precision of 0.83, a recall of 0.76, and an F1-measure of 0.79 for the Vector Space Model measure was achieved. Figure 6 shows the obtained results for the approach using VSM for the different users. Some differences between the results for the different users are evident. For instance, the system was able to mine tasks with an F1-measure of 0.91 from the task execution of user 2.

	precision	recall	F1-measure
VSM	0.83	0.76	0.79
LDA	0.66	0.57	0.61
TM	0.78	0.41	0.59

Table 3: Average results of the different task similarity measures

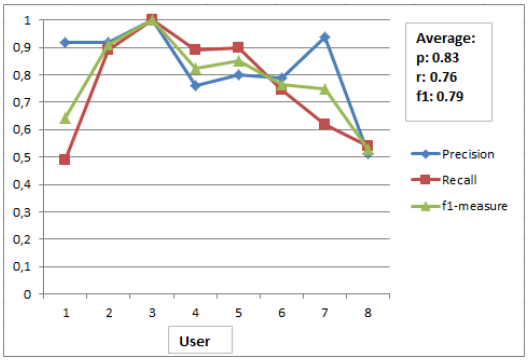

Figure 6: Precision, recall and f1-measure for VSM

For user 8 the system achieved an F1-measure of 0.53. The interaction histories for both task executions were analyzed and show as reason for this difference that user 2 executed the tasks in a fairly straightforward manner. The number of used resources was significantly smaller than what was observed for user 8. This indicates that the more complex the task execution process gets, the harder it is for the task detection system to mine tasks correctly. The combination of semantic and temporal aspects was intended to mitigate this problem, but did not completely resolve it. The task mining method utilizes the full text contents of all associated resources to obtain a notion of task similarity. The assumption underlying the task mining method, that semantic relatedness is key to task clustering when mining interaction histories is central for the discussed approach and brings many advantages but also some disadvantages as discussed in the following:

- **Rejected resources that are semantically not related:** In the study user 8 accessed several web resources that were not relevant for the task. These were for example resources which were accessed and rejected during an information search, because they did not contain the desired information. Such rejected resources were monitored but not assigned to the respective task. Omitting rejected resources brings advantages, as these resources are not relevant for the task.

- **Relevant resources that are semantically not related:** Some resources contain information that can only be assigned to a task with a deeper understanding of semantic relatedness. For example, a web site for currency conversion does contain clearly different textual contents than a web site for flight bookings. A human can link these resources easily if they need to calculate flight ticket prices in a different currency, but as both web sites have different purposes it might be difficult to relate them semantically in algorithmic fashion. The same applies for resources that use a different vocabulary while describing the same topic. More sophisticated text similarity measures, like LDA, could target this problem, as they do not rely on the exact terms, but built a generative model on how texts are created [4]. Semantic similarities as a measure of task similarity has clear limitations in this respect. The final merge of clusters based on the cluster switches help to mitigate the problem of resources that are not semantically related but belong together.

Overall, the results indicate that the task mining method presented in this paper can sufficiently detect knowledge-intensive tasks. An F1-measure of 0.79 was achieved using the Vector Space Model measure. A precision of 0.83 shows that the identified clusters already sufficiently represent task executions. Overall, the results are promising and indicate that an automatic task detection and task modeling of knowledge-intensive tasks in the computer desktop environment is possible.

6. RELATED WORK

Initial work on interaction histories addressed the analysis of work execution [2]. Interaction histories have proven useful to improve user system interaction, e.g. the UMEA system uses interaction histories to add resources to so called context pools [13]. Many publications involving interaction histories focus on the identification of known task instances generally used to proactively propose resources relevant for the identified task to the user [22, 18]. To provide proactive user support a task repository is required which can be generated manually [1, 8], in a supervised manner through labeled task instances [22] or in an unsupervised manner [5, 17, 18]. Here we focus on unsupervised mining [5, 17, 18] that creates task representations based on interaction histories. Mined tasks are composed of elements of different quality in the different approaches and are evaluated differently. For example Rattenbury et al. [18] evaluate their system by a usefulness study and do not report any values for precision, recall or F1-measure. Oliver et al. [17] and Brdiczka et al. [5] report performance values which are based on different data sets. Thus comparison of the performance is not possible. Nevertheless, in the following different assumptions are given about the evaluations of [17], [5] and the task mining method presented in this paper.

Brdiczka et al. [5] perform task mining based on document usage patterns identified by clustering of events up to a treshold, i.e. the treshold value modifies the number of identified tasks. An F1-measure of 0.32 with a precision of 0.20 is reported for a data set of ten users and 50 tasks, collected over up to three work days. The results are explained by the amount of noise in the input data. By limiting the data set to the six most frequent tasks a f-measure of 0.74 is obtained. Oliver et al. [17] report an F1-measure of 0.58 for a data set of one user and five tasks, collected over about four hours. The results are improved to a recall of 0.76% by using 1 hour chunks of data. Probabilistic latent semantic indexing is used by a mining method that requires a-priori knowledge about the number of tasks in the data set. This previous knowledge about the number of tasks is not required for the method presented in this paper and for [5]. Oliver et al. [17] is related to the approach presented here, as

semantic similarity is used. The difference is the amount of text used for the semantic similarity ([17] limits the text to the window titles), knowledge about the task number ([17] requires a-priori knowledge) and the task model ([17] reports about the process but does not provide a task model).

The main difference to existing task mining approach is the obtained task model. While the reviewed systems reduce activities to associated window titles [17], documents [5] or context structures (task relevant information and people) [18], the aforementioned task mining method populates a SHTM to provide an abstraction of the execution process, including details about the execution process. In contrast to existing approaches, mined SHTM instances can be used not only to enable support in the form of resource recommendation, but best practices and next step recommendations. To this point, this quality of support was only possible, if an expert modeled all tasks manually [1, 8].

7. CONCLUSION

Task mining is a relevant research domain to provide a data basis to describe, analyze and support knowledge workers in their daily activities. Mined tasks are foundations for user support systems and contribute to informal learning. This paper contributes a method that mines task instances based on a SHTM. A specific benefit of the SHTM is the hierarchical decomposition of interaction histories into knowledge actions and desktop operations that preserves abstract work execution information. Albeit the expressiveness of the model an automatic identification of model instances in a mining process is possible. We plan to use the mined task instances to propose work execution processes and deliver externalized best practices to knowledge workers.

The method of clustering by semantic similarity has been evaluated, using different approaches, TM, VSM and LDA. The best results were obtained for VSM. The weak results of LDA can be explained as the text corpus of the LDA algorithm was limited to the data of one study participant, each time it was applied. In the future we plan to do a longer study of knowledge workers performing their daily work and believe that LDA will perform better in such a setting with more data. Nevertheless, the results of the task mining method using VSM for semantic relatedness with an F1-measure of 0.79 (precision of 0.83, recall 0.76) are good.

8. REFERENCES

[1] B. Bailey, P. Adamczyk, T. Chang, and N. Chilson. A framework for specifying and monitoring user tasks. *Computers in Human Behavior*, 22(4):709–732, 2006.

[2] L. Bannon, A. Cypher, S. Greenspan, and M. Monty. Evaluation and analysis of users' activity organization, 1983.

[3] J. Bao, C. Lyon, P. Lane, W. Ji, and J. Malcolm. Comparing Different Text Similarity Methods. *UH Computer Science Technical Report 461*, 2007.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, May 2003.

[5] O. Brdiczka and J. B. Begole. Temporal Task Footprinting : Identifying Routine Tasks by Their Temporal Patterns. *Word Journal Of The International Linguistic Association*, 2010.

[6] K. Byström and P. Hansen. Conceptual framework for tasks in information studies. *Journal of the American Society for Information Science and Technology*, 56(10):1050–1061, 2005.

[7] J. Canny. GaP: a factor model for discrete data. *Proc. of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004.

[8] B. Cheikes, M. Geier, R. Hyland, F. Linton, and L. Rodi. Embedded training for complex information systems. *Tutoring Systems*, pages 36–45, 1998.

[9] M. Czerwinski, E. Horvitz, and S. Wilhite. A diary study of task switching and interruptions. *Proc. of the SIGCHI*, 2004.

[10] D. Ferrucci and A. Lally. Uima: An architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10:327–348, 2004.

[11] T. Hädrich. Situation-oriented Provision of Knowledge Services. *Dissertation, Martin Luther Universität Halle-Wittenberg*, 2008.

[12] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2nd edition, 2009.

[13] V. Kaptelinin. UMEA: translating interaction histories into project contexts, 2003.

[14] A. N. Leontiev. *Activity and consciousness*. Progress Publishers, 1977.

[15] C. Manning, P. Raghavan, and H. Schütze. *An Introduction to Information Retrieval*. Number c. Cambridge University Press, 2009.

[16] A. Newell and H. Simon. *Human problem solving*. Prentice-Hall, Englewood Cliffs, NJ, 1972.

[17] N. Oliver, G. Smith, C. Thakkar, and A. Surendran. SWISH: semantic analysis of window titles and switching history, 2006.

[18] T. L. Rattenbury and J. F. Canny. *An Activity Based Approach to Context-Aware Computing*. University of California at Berkeley Berkeley, 2008.

[19] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, Nov. 1975.

[20] B. Schmidt, H. Paulheim, T. Stoitsev, and M. Mühlhäuser. Towards a Formalization of Individual Work Execution at Computer Workplaces. In *Forthcoming: Lecture Notes in Artificial Intelligence, ICCS*, pages 270–283, 2011.

[21] B. Schmidt, T. Stoitsev, and M. Mühlhäuser. Activity-Centric Support for Weakly-Structured Business Processes, 2010.

[22] J. Shen. Detecting and correcting user activity switches: Algorithms and interfaces, 2009.

[23] T. Trabasso, P. Broek, and S. Suh. Logical necessity and transitivity of causal relations in stories. *Discourse Processes*, 12(1):1–25, 1989.

[24] M. Völkel. *Personal Knowledge Models with Semantic Technologies*. Ph.d. thesis, KIT, 2010.

[25] J. M. Zacks and B. Tversky. Event structure in perception and conception. *Psychological bulletin*, 127(1):3–21, Jan. 2001.