# BPMiner: Mining Developers' Behavior Patterns from Screen-Captured Task Videos

Jing Li[†], Lingfeng Bao[‡], Zhenchang Xing[†], Xinyu Wang[‡] and Bo Zhou[‡]
[†]School of Computer Engineering, Nanyang Technological University, Singapore
[‡]College of Computer Science, Zhejiang University, Hangzhou, China
[†]{jli030, zcxing}@ntu.edu.sg; [‡]{lingfengbao, wangxinyu, bzhou}@zju.edu.cn

## ABSTRACT

Many user studies of software development use screen-capture software to record developers' behavior for post-mortem analysis. However, extracting behavioral patterns from screen-captured videos requires manual transcription and coding of videos, which is often tedious and error-prone. Automatically extracting Human-Computer Interaction (HCI) data from screen-captured videos and systematically analyzing behavioral data will help researchers analyze developers' behavior in software development more effectively and efficiently. In this paper, we present BPMiner, a novel behavior analysis approach to mine developers' behavior patterns from screen-captured videos using computer vision techniques and exploratory sequential pattern analysis. We have implemented a proof-of-concept prototype of BPMiner, and applied the BPMiner prototype to study the developers' online search behavior during software development. Our study suggests that the BPMiner approach can open up new ways to study developers' behavior in software development.

## CCS Concepts

**•Software and its engineering → Development frameworks and environments; Software development process management;**

## Keywords

Software development; HCI data; screen-captured video; developers' behavior; online search~

## 1. INTRODUCTION

Researchers have investigated many perspectives of how developers seek and use information in software development tasks [3, 14]. To study the developers' information needs and behavior in software development, researchers have used human observer [23], think aloud [17], screen-captured videos [4], software instrumentation [11] to collect the observational data about the developers' information needs and behavior in software development tasks. Among these data collection methods, screen-captured videos provide a generic and easy-to-deploy method to record the developers' interaction with several software tools and application content during the task, for example, the IDE and the edited code, and the web browsers, search queries, and web pages visited.

Screen-captured videos can be analyzed to identify types of information the developers explored [13], information foraging actions [17], and behavioral patterns [16]. Analyzing screen-captured videos often requires significant efforts to manually transcribe and code the videos. For example, Ko and Myers [15] reported "analysis of video data by repeated rewinding and fast-forwarding" in their study of the cause of software errors in programming systems. Wang et al. [25] reported that one hour task video often requires 6-8 hours analysis time depending on the details of the information to be transcribed. The manual analysis of screen-captured videos often limits the use of video data in fine-grained study of developers' behavior in software development.

In this paper, we present BPMiner, a novel behavior analysis approach to mine developers' behavior patterns from screen-captured videos. BPMiner uses our home-made computer-vision-based video scraping tool (*scvRipper* [1]) to extract time-series HCI data from screen-captured task videos. The extracted time-series HCI data is a sequence of time-ordered items. Each item captures the software tool(s) and application content shown on the screen at a specific time in the task video. BPMiner then mines behavior patterns in the time-series HCI data using sequence abstraction, clustering, statistical analysis and data visualization.

We have implemented a proof-of-concept prototype of BPMiner and applied the BPMiner prototype to study the developers' online search behavior during software development. We conducted a case study to evaluate the BPMiner prototype with the 29 hours of screen-captured task videos collected in the study [18] on the developers' online search behavior in the two software development tasks. Compared with the high-level view of the developers' online search process observed from the manual analysis of the task videos in [18], this study using BPMiner reveals micro-level online search strategies and patterns during software development, which



*SAC 2016, April 04-08, 2016, Pisa, Italy*

DOI: http://dx.doi.org/10.1145/2851613.2851771

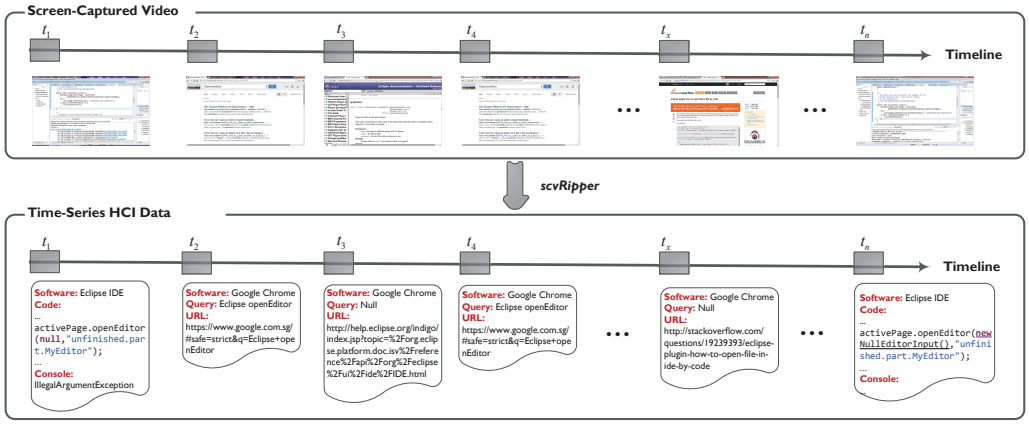


**Figure 1: Extracting Time-Series HCI Data from Screen-Captured Videos by *scvRipper***

offer insights for enhanced software tools that can deepen the integration of coding and web search in software development [19]. This case study provides initial evidence that the BPMiner approach can open up new ways to study developers' behavior in software development.

The remainder of the paper is structured as follows. Section 2 details the BPMiner approach and how it can be applied to analyze online search behavior. Section 3 reports the evaluation of the BPMiner prototype. Section 4 discusses the generalizability and characteristics of BPMiner. Section 5 concludes the work and discusses our future plan.

## 2. APPROACH

The BPMiner approach consists of five modules. First, BPMiner extracts time-series HCI data from the screen-captured task videos (Module 1). Then, BPMiner segments the time-series HCI data into sessions based on prosodic or semantic markers in the data (Module 2). Next, BPMiner extracts a set of features that can represent the key characteristics of the sessions (Module 3). After that, BPMiner clusters the sessions into groups to facilitate the discovery of behavior patterns (Module 4). Finally, BPMiner mines behavior patterns by statistical analysis and data visualization (Module 5).

### 2.1 Time-Series HCI Data Extraction

A screen-captured video is a sequence of time-ordered screenshots that a screencasting software (e.g., Snagit[1]) takes at a given time interval (often 1/30-1/5 second). To study the developers' behavior in software development using screen-captured task videos, the first step of BPMiner is to extract time-series HCI data from the screenshots of the task videos. To that end, BPMiner uses our home-made video scraping tool *scvRipper* [1]. The time-series HCI data BPMiner extracts from the video is a sequence of time-ordered items.

DEFINITION 1 (ITEM). *An item is an application with distinct content in the time-series HCI data.*

An item identifies which software the developer uses at what time and with which content. It has two attributes: application type and a set of application content (can be empty).

[1]http://www.techsmith.com/snagit.html

We denote an item as $Type\langle Content\rangle$. As such, an item is associated with a list of time stamps that record the time of all the occurrence of the item in the time-series HCI data.

Fig. 1 illustrates the input and output of the *scvRipper* tool in this setting. The screencasting software will record the developer's working process as a sequence of screenshots as shown in the upper part of Fig. 1. Given this task video, the *scvRipper* tool can automatically extract the time-series HCI data from the video as shown in the lower part of Fig. 1.

In the time-series HCI data shown in Fig. 1, two types of items can be defined: Eclipse IDE (IDEItem or II) and web browser (BrowserItem or BI). An IDEItem's content can contain a code fragment and a console output. A BrowserItem's content can contain an URL and a search query. We categorized the web sites that the developers frequently visit into seven web categories: search engines (SE), document sharing sites (DS), technical tutorials (TT), topic forums (TF), code hosting sites (CH), Q&A sites (QA), and API specifications (API) [18]. Thus, a BrowserItem can be further categorized into one of these seven web categories based on the web site of its URL. For example, the BrowserItems at time $t_2$, $t_3$ and $t_x$ will be categorized as search engines (SE), API specifications (API), and Q&A sites (QA), respectively.

### 2.2 Session Segmentation

The next step of BPMiner is to segment the time-series HCI data into meaningful sessions for further analysis. Depending on the purpose of the study, time-series HCI data can be segmented based on prosodic or semantic markers relevant to the study.

To study the developers' online search behavior during software development, BPMiner can segment the time-series HCI data into search sessions based on the occurrence of distinct search queries (i.e., semantic markers) over time, as defined below. The search sessions identify the time periods in which developers search for different information.

DEFINITION 2 (SEARCH SESSION). *A search session is a sequence of items between a search engine (SE) BrowserItem with query $Q1$ and the subsequent search engine BrowserItem with a different query $Q2$.*

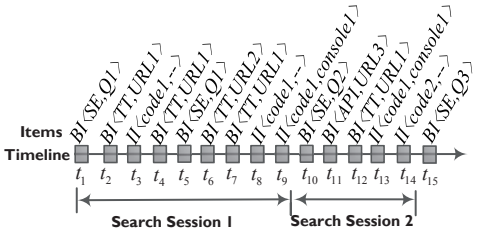


**Figure 2: An Illustrative Example of Search Sessions**

**Table 1: 12 Features from Each Search Session**

| Features | Content Usage | Application Usage |
|---|---|---|
| Overall features | | |
| NumItems | √ | |
| NumIDEBrowserSwitches | | √ |
| Duration | | √ |
| Web browser features | | |
| NumBrowserItems | √ | |
| NumWebCategories | √ | |
| NumKeywords | √ | |
| NumNewURLs | √ | |
| NumCategorySwitches | | √ |
| NumWebPageSwitches | | √ |
| BrowserDuration | | √ |
| IDE features | | |
| NumIDEItems | √ | |
| IDEDuration | | √ |

Given the sequence of all the items ordered by their appearance time stamps, BPMiner can segment the sequence into a sequence of search sessions by subsequent search engine BrowserItems with different queries. Fig. 2 shows an example of two search sessions segmented by the three search engine BrowserItems with different queries, i.e., $BI\langle SE, Q1\rangle$ at $t_1$, $BI\langle SE, Q2\rangle$ at $t_{10}$, and $BI\langle SE, Q3\rangle$ at $t_{15}$.

## 2.3 Session Representation

To characterize search sessions, BPMiner can use a feature space based on a set of descriptive statistics of application usage and content usage in search sessions. In the proof-of-concept prototype, BPMiner can automatically extract 12 features from search sessions, which were summarized in Table 1.

Among these 12 features, 6 features represent content usage in a search session: NumItems, NumBrowserItems, NumWebCategories, NumKeywords, NumNewURLs, NumIDEItems. The other 6 features represent application usage and exploration behavior in a search session: NumIDEBrowserSwitches, Duration, NumCategorySwitches, NumWebPageSwitches, BrowserDuration, IDEDuration.

The NumItems, NumBrowserItems and NumIDEItems counts the number of items, BrowserItems and IDEItems in a search session, respectively. The Duration, BrowserDuration and IDEDuration are the time duration (minutes) of a search session and the time spent on BrowserItems and IDEItems in a session. The NumKeywords counts the number of keywords in the query of a search session. The NumNewURLs counts the number of non-search-engine URLs in a search session that are not visited before. The rest Num...Switches features count the number of item switchings as defined below.

Definition 3 (Item Switching). *An item switching is the switching between two consecutive items with different attributes.*

Given the sequence of all the items ordered by their appearance time stamps, BPMiner can identify the switchings between IDE and web browser (i.e., an IDEItem followed by a BrowserItem or a BrowserItem followed by an IDEItem). It can also identify the switchings between different web categories (i.e., a BrowserItem with one web category followed by a BrowserItem with another web category), and the switchings between different web pages (i.e., a BrowserItem with one URL followed by a BrowserItem with another URL).

## 2.4 Session Clustering

In many situations, individual sessions carry only limited information about behavior patterns. We need to cluster similar sessions such that behavior pattern can emerge from clusters of sessions. What type of clustering techniques to use depends on the nature of the data and the cost and benefits of data labelling.

Assume that BPMiner extracts code fragments and error messages from the IDE, and web page content from the web browser. It can use topic modeling techniques such as Latent Dirichlet Allocation (LDA) [2] to cluster search sessions based on the topics of code fragments, error messages, and web page content that the developer works on in the sessions. If think aloud [24] protocol is adopted, recorded speech information may be analyzed automatically using speech recognition techniques [21], and then used as labels of the IDE and browser content. BPMiner can then use supervised clustering techniques such as Labelled LDA [22] or Explicit Semantic Analysis [5] to cluster search sessions.

## 2.5 Pattern Discovery

Once sessions are clustered, we can use data mining techniques (e.g., sequential pattern mining [7]), statistical analysis (e.g., Markov model, statistical testing), and data visualization (e.g., heat map, timeline plot, parallel coordinates, star plot) to discover patterns in session clusters.

For example, to study the developers' online search behavior during software development, BPMiner can integrate the following statistical analysis methods and data visualizations that are widely supported in data analysis libraries such as Matlab [9] and R [10]. In the prototype, we use heat map and timeline plot to compare when the developers perform what type of search sessions and how their online search unfolded in different development tasks. We can consider search sessions as subjects, different types of development tasks and clusters of search sessions as independent variables, and the 12 features of search session as dependent variables. BPMiner can then perform Multivariate ANalysis of VAriance (MANOVA) analysis [6] to test the between-subjects effect of tasks, search session clusters, and the combination of different levels of tasks and search session clusters (i.e., Task×SessionType interaction) on the 12 features of search sessions.

# 3. EMPIRICAL EVALUATION

To evaluate if the proposed BPMiner approach can help us mine developers' behavior patterns from screen-captured videos, we have implemented a proof-of-concept prototype of BPMiner and applied the prototype to study the developers' online search behavior during software development. In this evaluation, we aim to use the BPMiner prototype to answer the following two research questions regarding the developers' online search behavior:

**RQ1** *Are there latent types of search sessions? What are commonalities and differences of different types of search sessions?*

**RQ2** *How do different types of tasks and different types of search sessions affect the developers' online search behavior?*

## 3.1 Dataset

This evaluation used the task videos collected in a study of developers' online search behavior during software development [18], which involved two types of software development tasks. The first task (Task1) is to develop a new P2P chat software. The Task1 requires the knowledge about Java multi-threading, socket APIs, and GUI framework (e.g., Java Swing). The second task (Task2) is to maintain an existing Eclipse editor plugin. The Task2 includes two subtasks. The first subtask is to fix two bugs in the existing implementation. To fix these two bugs, developers need knowledge about Eclipse editor API and plugin configuration. The second subtask asks developers to extend existing editor plugin with file open/save/close features and file content statistics (e.g., word count). This subtask requires developers to program to Eclipse editor and view extension points (e.g., *EditorPart*).

11 graduate students were recruited in the first task, and 13 different graduate students were recurited in the second task. The developers were asked to work on the development task in a 2-hours session. They were allowed to end the task anytime during the session. All the developers used Windows 7 or newer operating systems. They used Eclipse3.6 (or newer) or MyEclipse8.0 (or newer). They used Google Chrome, Mozilla Firefox, or Internet Explorer to browse the Internet.

The developers were instructed to use a screencasting software to record their task process once they started working on the assigned task. These screen-captured task videos were the primary input for this study. The task videos of 3 developers in the first task and the task video of 1 developer in the second task were corrupted. As such, this study analyzed the task videos of 8 developers in the first task and the task videos of 12 developers in the second task.

## 3.2 RQ1: Search Session Commonalities and Differences

The BPMiner prototype identifies 168 search sessions in the extracted time-series HCI data. It computes the 12 features of these search sessions as defined in Table 1. It clusters these 168 search sessions using EM algorithm [20] implemented in Weka [8]. We first report our modeling and analysis of search session commonalities and differences.

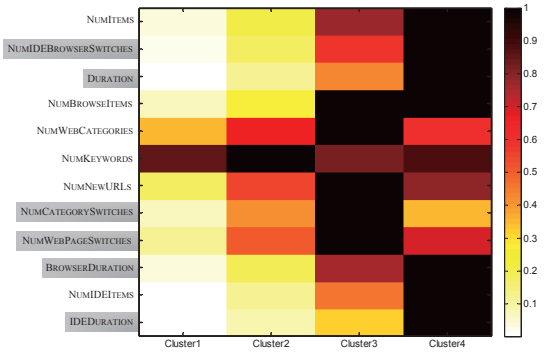


**Figure 3: Heat map of the Feature Values of the 4 Search-Session Clusters**

Fig. 3 shows the heat map of the normalized average feature values of the 12 features across the four clusters. Larger values were represented by darker colors, while smaller values were represented by brighter colors. Features in gray background are application usage features, while those in white background are content usage features. The heat map of feature values reveals three distinct meta-clusters: *short* (cluster1), *medium* (cluster2), and *long* (cluster3 and cluster4).

The duration of *short* sessions is very short (less than 1 minute). In these short sessions, the developers mainly used web browsers (6.02±5.86 BrowserItems, compared with 0.30±0.54 IDEItems). They spent on average 93% of total session time in the web browser, but very little time in the IDE. There were a very small number of switchings (0.56±0.99) between IDE and web browser. The developers opened a small number of new URLs (1.62±0.12), and switched a small number of times between different web pages (2.18±0.47) and between different web categories (0.66±0.94).

The duration of *long* sessions ranges from 4.67 minutes to 104.16 minutes. In these long sessions, the participants frequently used both the web browser and the IDE. There were a large number of switchings (17.63±12.70 in Cluster3 and 30.21±35.46 in Cluster4) between IDE and web browser. Compared with short sessions, the developers opened a large number of new URLs (9.12±3.26 in Cluster3 and 7.26±5.92 in Cluster4), switched a large number of times between different web pages (18.13±4.26 in Cluster3 and 12.53±8.44 in Cluster4) and between different web categories (9.44±2.56 in Cluster3 and 3.26±2.38 in Cluster4).

The statistics of feature values of the *medium* sessions fall in between those of *short* sessions and *long* sessions.

An interesting observation is that search sessions of different clusters differ mainly in application usage features. The differences of content usage features (especially NumWebCategories and NumKeywords) are smaller across search sessions of different clusters.

Our results identify 4 types of search sessions: *refine* (cluster1), *medium select* (cluster2), *long select* (cluster3), and *integrate* (cluster4). *Refine* sessions are characterized by the least diverse transitions between different types of items, and the high probabilities to transit from the IDE or different categories of web sites to search engine. The model shows

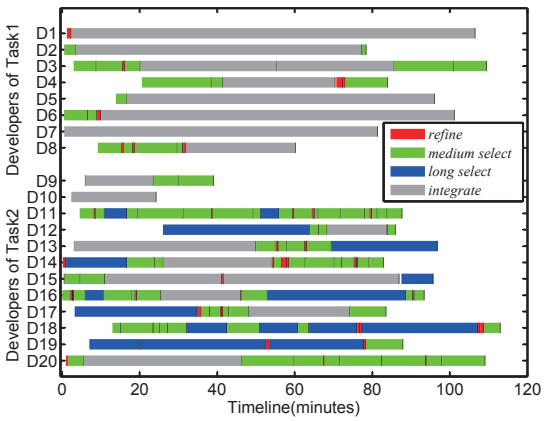


**Figure 4: Timeline Plot of Search-Session Types**

that the developers were highly likely to visit search engine after visiting topic forms (TF) and technical tutorials (TT). *Select* sessions are characterized by the most diverse transitions between different types of items with similar transition probabilities. That is, the developers explored many types of online resources in *Select* sessions. *Integrate* sessions are characterized by the transitions mainly between the IDE and different categories of web content. That is, the developers were likely to go back to the IDE after visiting certain online resources in *Integrate* sessions.

Different types of search sessions reflect distinct online search behaviors during software development. *Refine* sessions are *short*. The developers refine search based on a quick exploration of the search results of previous search, for example when they find some hints in the search results or feel that the previous search was unsuccessful. *Select* sessions are *medium* or *long*. The developers explore, compare and select online resources to determine useful information in these search sessions. This exploration and selection process are very diverse and can be time-consuming. *Integrate* sessions are *long*. The developers find useful online resources in these sessions and integrate online resources in the IDE. The integration can take long time.

## 3.3 RQ2: Task Differences and Task× Session-Type Interaction

This section reports our modeling and analysis of the effects of tasks and search session types on the developers' online search behavior.

### 3.3.1 The Effect of Task Differences

Fig. 4 shows the timeline plot of the four types of search sessions in the working process of the 20 developers. Different colors represent different types of search sessions. Black lines represent the ending of the search sessions.

Fig. 4 shows that the developers in the first task (D1-D8) usually had less search sessions than the developers in the second task (D9-D20). The first-task developers had only *refine*, *medium select* and *integrate* sessions, but no *long select* sessions. The second-task developers had the search sessions of all the four types. The two second-task developers (D9 and D10) had Eclipse plugin development experience. They were able to quickly find and integrate relevant online resources to complete the task in short time. Thus, their task processes were different from the rest 10 second-task developers.

**Table 2: Test Results of Between-subjects Effects**

| | SessionTypes | | Tasks | | Task×SessionType | |
|---|---|---|---|---|---|---|
| | $p$ | $\eta_p^2$ | $p$ | $\eta_p^2$ | $p$ | $\eta_p^2$ |
| NumItems | **.00** | .51 | **.01** | .04 | **.02** | .05 |
| NumIDEBrowserSwitches | **.00** | .33 | **.00** | .11 | **.00** | .14 |
| Duration | **.00** | .72 | **.00** | .17 | **.00** | .24 |
| NumBrowserItems | **.00** | .34 | **.01** | .04 | **.02** | .05 |
| NumWebCategories | **.00** | .33 | .56 | .01 | .48 | .00 |
| NumKeywords | .33 | .02 | .12 | .02 | .92 | .00 |
| NumNewURLs | **.00** | .31 | .11 | .02 | **.03** | .04 |
| NumCategorySwitches | **.00** | .41 | .55 | .01 | .41 | .01 |
| NumWebPageSwitches | **.00** | .41 | **.04** | .03 | .10 | .03 |
| BrowserDuration | **.00** | .41 | **.00** | .05 | **.00** | .06 |
| NumIDEItems | **.00** | .55 | .21 | .01 | .50 | .00 |
| IDEDuration | **.00** | .72 | **.00** | .19 | **.00** | .25 |

The first-task developers usually started the task with several *refine* and *medium select* sessions, followed by a long *integrate* session till the end of the task. Only 3 out of the 8 first-task developers searched again (one or two *refine* or *medium select* search sessions) after the *integration* session started. In contrast, the second-task developers had much more dynamic search sessions over time. They had much more *refine* and *select* search sessions. About 17% of their *select* sessions were *long select* sessions. These *refine* and *select* sessions were scattered throughout the task process. The second-task developers usually had shorter *integrate* sessions than the first-task developers. Three second-task developers (D11, D18, D19) did not have *integrate* sessions.

We attributed these differences in the developers' online search behavior to the differences of the two development tasks. The first task is to develop a new P2P chat software using Java socket. The participants can easily find many online code examples on Java socket programming or even code examples implementing similar features as required by the first task. Most participants can successfully modify online code examples without much need for further search. In contrast, the second task is to fix bugs in an Eclipse editor plugin and extend the plugin with new features. The participants in general lacked the knowledge of the Eclipse APIs involved in the task. Unforunately, no single online resource can cover all the involved APIs. Furthermore, due to the unfamiliarity with these APIs, the developers often encountered unexpected issues while integrating online resources. Thus, they had to keep searching and learning throughout the task.

### 3.3.2 Task×SessionType Interaction

Table 2 summarizes the test results of between-subjects effects of the 2 (tasks) × 4 (search-session types) MANOVA analysis on the 12 search-session features. *p-values* in bold font indicates that the main effect of session types, the main effect of tasks, and the interaction effect of tasks and search session types are significant on the difference of the corresponding features. $\eta_p^2$ values in red, blue and black font indicate the large, small and trivial effect respectively.

The MANOVA test results show that the main effect of search session types is significant for all the search session

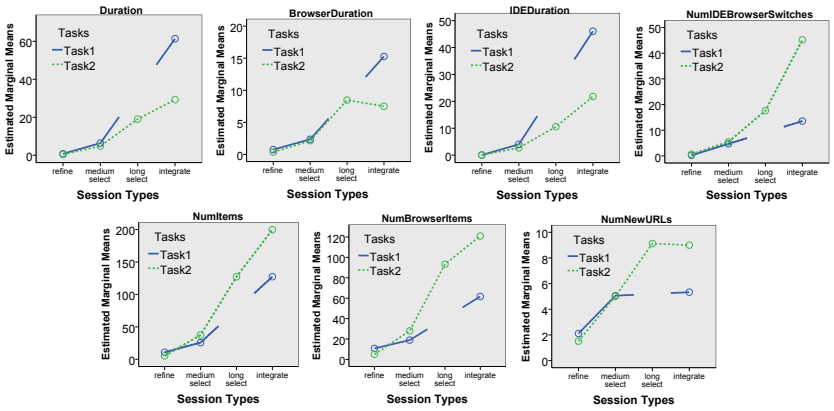


**Figure 5: Task×SessionType Interactions on Features**

features except NumKeywords. This result is consistent with the heat map visualization in Fig. 3. $\eta_p^2$ values indicate that search session types have large effect on all the search session features except NumKeywords. The main effect of tasks is significant for 2 content usage features and 5 application usage features. Among these seven features, tasks have large effect on three application usage features (i.e., NumIDEBrowserSwitches, Duration, IDEDuration), and have small effect on the other four features. The interaction effect of tasks and search session types is significant for 3 content usage features and 4 application usage features. Among these seven features, Task×SessionType interactions have large effect on three application usage features (NumIDEBrowserSwitches, Duration, IDEDuration), and have small effect on the other four features.

Fig. 5 visualizes the interaction effect of tasks and search-session types on the 7 features on which the Task×SessionType interaction effect was significant. As Task1 did not have *long select* session, there was no data point for Task1 at *long select*. We can see that the difference of the 7 features between Task1 and Task2 was greatest in *integrate* sessions, while the difference of the 7 features between Task1 and Task2 was very small in *refine* and *medium select* sessions. In *integrate* sessions, Task1 had larger values in three duration features (Duration, BrowserDuration, and IDEDuration), while Task2 had larger values in one application usage feature (NumBrowserSwitches) and three content usage features (NumItems, NumBrowserItems, NumURLs).

Our results show that search-session types have large and significant effect on both application usage and content usage in search sessions, except for NumKeywords. Different types of tasks affect the developers' overall search process, and affect mainly application usage at session level. Different types of tasks has little impact on the search behavior in *refine* and *medium select* sessions, but has big impact on the search behavior in *integrate* sessions.

## 4. DISCUSSION

Our BPMiner approach uses video scraping tool to extract HCI data from screen-captured videos. Alternatively, software instrumentation can be used to log the developers' interaction with the software tools they use and the application content. Instrumenting many of today's software system is considerably complex. Compared with software instrumentation, our BPMiner approach provides a generic, easy-to-deploy solution to collect time-series HCI data for studying the developers' behavior patterns and problem solving strategies using screen-captured videos. The generality of the underlying data model of BPMiner allows it to analyze time-series HCI data from a variety of software development tasks. BPMiner can model the time-series HCI data using meta-model techniques such as Eclipse ECore framework. Then, based on the task-specific requirements, BPMiner could support the on-demand design of an analysis process by interactively "mashing-up" available data analysis and visualization methods.

Screen-captured videos have been widely used to collect observational data in studying human aspects of software engineering, especially for modeling the developersą́ behavior in software development tasks [16, 18] and eliciting design requirements for innovative software development tools [12, 23]. However, studying micro-level behavior patterns in software development tasks is time consuming, because it often requires iterative open coding of screen-captured videos [15]. Compared with these qualitative data collection and analysis methods, our BPMiner approach supports micro-level, quantitative analysis of developers' behaviors in software development tasks. Our study shows that BPMiner can help researchers discover latent behavior patterns in the time-series HCI data. As such, BPMiner could provide new insights into the outstanding difficulties in software development tasks and the limitations of existing tool supports.

Understanding developers' information behavior and needs is crucial for improving existing software development practices. Our findings unveil that developers have to perform many context switchings and explore information from various sources. However, in current practice, developers work in the IDE but search online resources in the web browser. This insight inspired the development of an in-IDE ambient search agent [19]. The in-IDE search agent can unobtrusively monitor the developers' programming activity in the IDE and visualize their working focus over time. The developer can use the working context to augment his search query or refine the search results. The search agent can use the working context to tweak the ranking of the search results. It also uses the context to annotate the search results and web pages to help the developer assess and browse the

search results. This in-IDE search agent can enable deeper integration of developers' working and search context, and thus smoother interleaving of coding and web search in software development.

# 5. CONCLUSION AND FUTURE WORK

We have presented a novel approach BPMiner for extracting, modeling, and analyzing developers' behavior patterns using screen-captured videos. Unlike instrumentation approach, BPMiner extracts time-series HCI data from screen-captured videos using computer vision technique. BPMiner supports exploratory sequential pattern analysis for the discovery of developers' behavior patterns. We implemented a prototype of BPMiner, and conducted an evaluation of the BPMiner prototype. In this evaluation, we used BPMiner to analyze developers' online search behavior during software development. The empirical evaluation demonstrates that BPMiner is a promising approach to mine developers' behavior patterns from screen-captured task videos. As an initial evaluation, our study was limited by its small dataset. The proposed BPMiner approach needs to be further evaluated for different software development tasks. We will also improve the generality of our BPMiner approach with meta-modeling techniques and "mash-up" support for on-demand pattern discovery and analysis.

## Acknowledgments

The authors thank the reviewers for their helpful comments. This work is partially supported by MOE AcRF Tier 1 grant M4011165.020 and MOE Scholarships.

# 6. REFERENCES

[1] L. Bao, J. Li, Z. Xing, X. Wang, and B. Zhou. scvripper: video scraping tool for modeling developers' behavior using interaction data. In *Proc. ICSE*, volume 2, pages 673–676, 2015.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[3] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In *Proc. CHI*, pages 1589–1598. ACM, 2009.

[4] E. Duala-Ekoko and M. P. Robillard. Asking and answering questions about unfamiliar apis: An exploratory study. In *Proc. ICSE*, pages 266–276, 2012.

[5] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611, 2007.

[6] J. F. Hair. Multivariate data analysis. 2009.

[7] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proc. SIGKDD*, pages 355–359, 2000.

[8] http://www.cs.waikato.ac.nz/ml/weka/.

[9] http://www.mathworks.com/products/matlab.

[10] http://www.r project.org/.

[11] J. H. Kim, D. V. Gunn, E. Schuh, B. Phillips, R. J. Pagulayan, and D. Wixon. Tracking real-time user experience (true): a comprehensive instrumentation solution for complex systems. In *Proc. CHI*, pages 443–452, 2008.

[12] A. J. Ko, H. H. Aung, and B. A. Myers. Design requirements for more flexible structured editors from a study of programmers' text editing. In *CHI*, pages 1557–1560, 2005.

[13] A. J. Ko, H. H. Aung, and B. A. Myers. Eliciting design requirements for maintenance-oriented ides: a detailed study of corrective and perfective maintenance tasks. In *Proc. ICSE*, pages 126–135, 2005.

[14] A. J. Ko, R. DeLine, and G. Venolia. Information needs in collocated software development teams. In *Proc. ICSE*, pages 344–353, 2007.

[15] A. J. Ko and B. A. Myers. A framework and methodology for studying the causes of software errors in programming systems. *Journal of Visual Languages & Computing*, 16(1):41–84, 2005.

[16] A. J. Ko, B. A. Myers, M. J. Coblenz, and H. H. Aung. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Trans. Softw. Eng*, 32(12):971–987, 2006.

[17] J. Lawrance, C. Bogart, M. Burnett, R. Bellamy, K. Rector, and S. D. Fleming. How programmers debug, revisited: An information foraging theory perspective. *IEEE Trans. Softw. Eng*, 39(2):197–215, 2013.

[18] H. Li, Z. Xing, X. Peng, and W. Zhao. What help do developers seek, when and how? In *Proc. WCRE*, pages 142–151, 2013.

[19] H. Li, X. Zhao, Z. Xing, X. Peng, D. Gao, L. Bao, and W. Zhao. amassist: In-ide ambient search of online programming resources. In *Proc. SANER*, 2015.

[20] T. K. Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[21] M. Nishimura. Speech recognition method, Sept. 17 1991. US Patent 5,050,215.

[22] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proc. of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 248–256, 2009.

[23] M. P. Robillard, W. Coelho, and G. C. Murphy. How effective developers investigate source code: An exploratory study. *IEEE Trans. Softw. Eng*, 30(12):889–903, 2004.

[24] M. W. Van Someren, Y. F. Barnard, J. A. Sandberg, et al. *The think aloud method: A practical guide to modelling cognitive processes*, volume 2. Academic Press London, 1994.

[25] J. Wang, X. Peng, Z. Xing, and W. Zhao. An exploratory study of feature location process: Distinct phases, recurring patterns, and elementary actions. In *Proc. ICSM*, pages 213–222, 2011.