Proactive Task Execution Using Data Sharing and Event Transition among Personal Devices

Hocheol Jeon, Taehwan Kim and Joongmin Choi
Department of Computer Science and Engineering, Hanyang University
Ansan, Gyeonggi-Do 426-791 - Korea
[e-mail: hochuls@chollian.net]
[e-mail: kimth@islab.hanyang.ac.kr]
[e-mail: jmchoi@hanyang.ac.kr]
*Corresponding author: Joongmin Choi

Received August 13, 2010; revised September 10, 2010; accepted October 11, 2010; published December 23, 2010

Abstract

This paper proposes an intelligent technique for data sharing and event transition among personal devices including smart phones, laptops, and desktops. We implemented the PES (Personal Event Service) system that proactively executes appropriate tasks across multiple devices without explicit user requests by sharing the data used by the user and recognizing user intention based on the observed actions of the user for specific devices. The client module of PES installed on each device monitors the user actions and recognizes the intention of the user. The server provides data sharing and maintenance for clients. The connection between client and server is established by Java RMI (Remote Method Invocation). A series of experiments were performed to evaluate user satisfaction and system accuracy, and the results showed that the PES system can proactively provide appropriate, personalized services with a high degree of satisfaction to the user in an effective and efficient manner.

Keywords: Personal event service, user intention recognition, proactive task execution, data and event sharing

This work was supported by the Industrial Strategic Technology Development Program (10035348, Development of a Cognitive Planning and Learning Model for Mobile Platforms) funded by the Ministry of Knowledge Economy (MKE), the Korean government.

DOI: 10.3837/tiis.2010.12.015

1. Introduction

Due to the popularization of personal computers and the rapid distribution of network-based devices such as smart phones and IPTVs, users are increasingly using multiple personal devices including desktops, laptops, smart phones, IPTVs, and so on [1][2]. Many users are working with these devices in different places, such as home, office, school, or even in the subway train [3][4].

More often, there are situations when users who use their office PCs during daytime want to continue their work using home PCs at night, or on their way home in the subway train using smart phones. Some preparatory work is necessary to accomplish this task. In order to continue the job at home, users have to save the work data either by copying them onto a USB memory or by sending them to a P2P server and later restoring them into local PCs. Even for remote access service, they have to leave the office PCs powered up. These procedural behaviors are performed manually, and if more devices are involved, the procedures would be complicated and inefficient.

The motivation of this paper is to automate this process by using data sharing and event transition among several personal devices, and, consequently, to mitigate cumbersome preparatory work. The main idea of our proposed system is to proactively execute tasks by monitoring user behavior and recognizing user intention. The data obtained as a result of user intention recognition is shared among personal devices so that the user task being done in one device can be continued to be done in other devices seamlessly.

Proactive execution of a task is accomplished by actively recognizing context information and user intention when the goal is not explicitly given, and proactively executing suitable tasks without user intervention [5]. As stated in [6], proactive systems are assumed to work on behalf of the user and to act on their own initiatives.

We implemented the PES (Personal Event Service) system that proactively executes appropriate tasks across multiple devices without explicit user requests by sharing the data used by the user and recognizing user intention based on the observed actions of the user for specific devices. The client module of PES installed on each device monitors the user actions and recognizes user intention. The server manages the shared data and events and provides maintenance for the clients. The connection between a client and the server is established by Java RMI (Remote Method Invocation). A series of experiments were performed to evaluate user satisfaction and system accuracy, and the results of these experiments showed that PES can provide appropriate, personalized services with high satisfaction to the user in an effective and efficient manner.

2. Related Work

In this section we describe the background for our work by first presenting some studies on user intention recognition based on users' implicit feedback, and then discussing proactive task execution.

2.1 User Intention Recognition

User intention recognition means that the system recognizes user intention for the next action based on the observation of the user's previous behavior without explicit requests [7]. Two processes are important for successful intention recognition; continuous monitoring of user

behavior and prediction of intentions that are related to the monitored actions.

As a study for intention recognition in personal computers, the Lumiere research project [8] explored probabilistic techniques to provide improved help guide to users while they interact with the Microsoft Office applications. A special event monitoring system, *Eve*, was developed to capture a wide range of user actions. Bayesian models were developed to predict user goals by considering their backgrounds and their interactions with the applications, as well as their explicit queries. Several context-aware applications capable of deriving user's information needs also have been developed.

Watson project [9] and IntelliZap project [10] use the user information context in the form of open Web pages and opened documents as a guide to Web and database searching. Many researches in information retrieval also attempt to recognize correct user intention based on implicit feedback or search behaviors to provide more accurate search results. In particular, [11] observes user behavior in PCs and predicts user intention based on the observed actions. A user's browsing activity is modeled by a tuple with three attributes including the name, the duration, and the parameters of the activity. It infers user intentions probabilistically by using a hidden Markov method. WebWatcher [12] acts as a Web tour guide that suggests appropriate hyperlinks based on user interests that are learned from previous actions of other users with similar interests. In [13], a Web browsing assistant traces user behavior in a conventional Web browser. It analyzes user behavior such as follow-up of the hyperlinks in an HTML document, and then estimates user interests by parsing the document and recommending HTML documents.

In [14], agents were used to learn a user's past browsing behaviors, and by using them to suggest a better location to visit from the current position. To serve users with some appropriate assistance in computer-assisted teleoperation, [15] applied a Hidden Markov Model approach to recognize human intentions from time series haptic data that were gathered during interaction. Also, some recent studies tried to recognize user intentions to provide better information or services by observing user behaviors in various fields by applying diverse approaches [16][17][18][19].

2.2 Proactive Execution of Tasks

Proactive execution actively recognizes context information and user intention when the goal is not explicitly given, and eventually plans proactively and executes the plan without user intervention [5].

The most common type of proactive execution of tasks is to proactively provide information relevant to the current job [20][21]. Similar work is described in [22][23] that can proactively suggest appropriate pages to users who are searching for Web information that are related to the current task. For example, upon editing a document, relevant keywords would be extracted and utilized to search on the Web for additional useful resources on the same topic. Similarly, activities such as accessing a Web page or an email message can be interpreted as implicit expression of information need, which is used to retrieve potentially relevant information proactively without explicit user requests [24][25] proactively cares customers' health by applying a multi-agent system where each agent interacts with other agents and proactively makes a plan to provide appropriate services.

However, previous researches focused only on a single device, without handling the situations when the observed user behavior in one device can affect the same behavior in other devices. We try to deal with this issue by proposing a method of proactive task execution across multiple devices that are used by the same person.

3. The PES (Personal Event Service) System

As shown in **Fig. 1**, the PES system employs client-server system architecture. The server side of PES consists of Server Task Manager, Database Manager, and Server Data Manager. The client side of PES employs Client Data Manager, Client Intention Recognizer, Client Monitoring Manager, and Client System Manager.

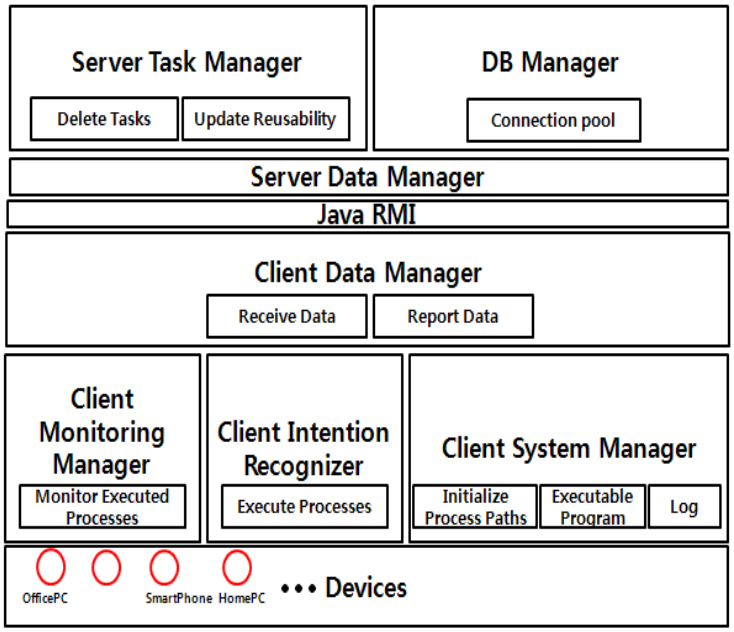


Fig. 1. The system architecture of PES

The *Server Task Manager* manages the task information that is periodically received from the clients. This includes storing the task information to the database, deleting expired task information from the database, and updating the reusability of devices. The *Database Manager* manages the user information, the device information, and the task information of each device. The *Server Data Manager* is the main component in the server, executing the *Server Task Manager*, managing the connection with a client, storing monitoring results to the database, and saving the used files to the server file system.

Monitoring information includes the user ID, the device sequential number, the application ID number, the event type, and the absolute paths of the files that were received from the clients. The network connection between the server and each client is implemented using Java RMI. Through this connection, a client periodically sends the data of the applications that are executed by the user to the *Server Data Manager* that stores this data to the database.

The *Client Data Manager* requests and manages the connection with the server. It receives the information about the tasks performed by the user in other devices from the server, and sends the monitoring results that are periodically gathered to the server. The *Client System Manager* finds the applications that can be monitored in the device and writes their absolute paths of execution files in an XML format. The *Client Monitoring Manager* periodically monitors the processes that are executed by the user in the current device, and gathers the information about the processes and their data files with the CPU time values greater than the threshold α. The *Client Intention Recognizer* monitors the current processes to recognize the intention of the user, and executes proper tasks based on the recognized intentions.

The PES system interacts with users via the Web to obtain basic information about themselves and their personal devices. Users should register each of their personal devices to PES through a Web browser with its IP address, device type, manufacturer, and model name. After registration, the client module of PES must be downloaded and installed on each registered device.

Fig. 2 illustrates the interaction of the user who has several personal devices with the PES system, along with the data flow between PES and the server.

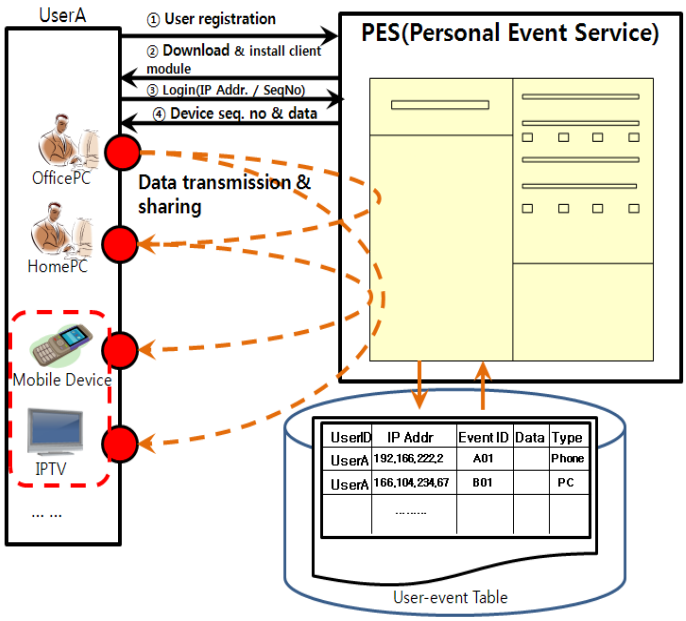


Fig. 2. PES interactions with the user and the server

4. Intention Recognition and Proactive Execution

In this section, we describe the client module in detail, and explain the employed methods of intention recognition and monitoring. The client module is installed on each user device, sharing task data with other client modules via server, monitoring performed processes, and sending monitoring results to the server.

4.1 Event

An event model E is defined as $E = <S, U, St>$, where S is the system in which the event occurred, U is the user who generates the event, and St is the status of the event. U contains the user information such as the user ID and the status of user devices including an IP address, the availability status, and a unique sequence number. St contains the information of the event such as the event type, the application type, the used program ID, a local file path, a server file path, and a file name. The same event occurred in a device by the same user are continuously updated to keep consistency of file contents.

4.2 Executable Applications

We selected some common applications for monitoring user intention and executing tasks proactively, and classified these applications into various types, because it is important to

relate the close connection between an application program type and its *CPU time* usage in order to recognize user intention. The CPU time means the amount of time the CPU is actually executing instructions for an active application program. For example, a process for an editor application would consume CPU time to handle the events of keyboard inputting, scrolling, and clicking. In this paper, we classify application programs into 3 types including Editor, Reader, and Viewer as shown in **Table 1**.

Table 1. Classification for Applications

Type	Example Applications	Description
Editor	Hwp(Korean Word processor), MS-Word, MS-PowerPoint, MS-Excel, Notepad, UltraEdit, ERWin, StarUML	Readable and writeable applications
Reader	Acrobat Reader, Ghost View	Only readable applications
Viewer	MSPaint, AlSee	Only viewable applications

In general, the Editor consumes more CPU time than the Reader and the Viewer. Hence, we set different CPU time thresholds for different application types, assigning a larger value for the Editor type events and smaller values for the Reader type and the Viewer type events. However, for simplicity, we set the same threshold value for the same application type in different devices.

4.3 User Intention Recognition

The purpose of user intention recognition in PES is to decide whether the files that were used in other devices would be *reused* by the same user in the current device. We assume that users double-click the shortcut icon of the application to create a new file or open an existing file. To achieve this purpose, the client module employs a monitoring technique for target processes.

Fig. 3 describes our user intention recognition algorithm. User intention recognition begins with the user information including the user ID and the device information including the device ID and an IP address. Then, it stores the event information for all executing processes in the current device to the *Event* array. For each event *e*, if its CPU time exceeds the threshold α, then the algorithm finds the absolute path of its data file, converts it to a serialized form, and sends it to the server along with *e*.

```
IntentionRecognition(user_info, device_info)
    Event ← events for all processes that are running in
            the current device
    for each event e in Event do
      p ← Event.Process
      if p.CPU time ≥ α then
        file ← the full path for the data file
        usedFile ← convert file to serialized format
        sendRealFileToServer(usedFile)
        sendEventsToServer(e, user_info, device_info)
      end if
    end for
```

Fig. 3. User intention recognition algorithm

4.4 Proactive Execution of Tasks

Fig. 4 describes the proactive execution algorithm in PES. The algorithm begins with the events and their data files received from the server that was used previously in other devices. Then, it loads the information of all installed application programs and some compatible programs including their absolute paths for executables, program names, and executable file names. The algorithm then finds all processes that are just executed to reuse the data files. For each event *e*, our system proactively executes the same applications that were used previously in other devices. In cases where the same applications are not installed in the current device, the system finds compatible programs and executes them. If even compatible applications do not exist, it notifies the user with warning messages.

5. Experimental Results and Analysis

In this section, we describe a simple scenario and our experiment environment, and evaluate PES with user satisfaction and system accuracy measures.

5.1 Scenario

Fig. 5 shows a scenario for data and event sharing among personal devices, including a home PC, an office PC, and a smart phone, using intention recognition and proactive execution. In this scenario, assume that John is a company researcher involved in an important project. He is preparing a report by editing a file with MS-Word, and at the same time he is listening to some music through a MP3 player on his office PC. Since he cannot finish his job at the office, he wants to continue doing it at home. On his way home, when he turns on the music player in his smart phone, the music that was playing on his office PC is proactively played again. When he arrives home and launches MS-Word in the home PC, the report files that he was working on previously are proactively opened, and he can continue his intended work instantly. In a similar fashion, when he goes to his office the next day, the applications and the data files that were used in his home PC can be executed proactively in his office PC.

```
ProactiveExecution( )
    Event ← prcocess events previously occurred in other devices
    Process ← processes running in the current device
    Programs ← absolute paths, process names, and names of
        executable files for all installed programs in the current device
    CompatiblePrograms ← absolute paths, process names, and names
        of executable files for all installed compatible programs in the
        current device
    for each process p in Process do
        find an event e in Event where e.Process.Name = p.Name
        find a program pg in Programs
            where p.Name = pg.processName
        if found then
            executeProcess(pg, e);
        else
            find a program cpg in CompatiblePrograms
                where p.Name = cpg.processName
            if found then
                executeProcess(cpg, e);
            else
                displayWarning(e);
            end if
        end if
    end for
```

Fig. 4. Proactive execution algorithm

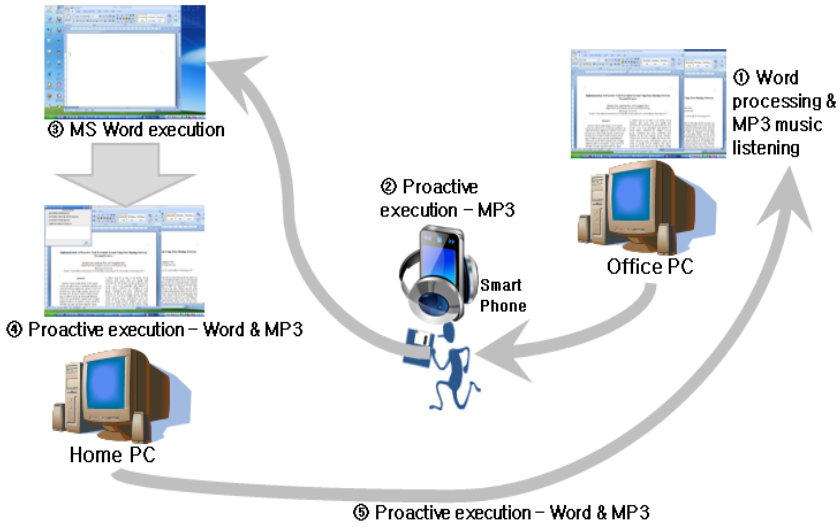


Fig. 5. A simple scenario

5.2 Experiment Environment

For the experiment, we collected 1389 events from 6 users during a period of approximately 3 months. The users registered all their devices used in the experiment to the server, and downloaded and installed the client module of PES to all devices. Also, each user manually recorded information about the data files that were used, tagging each file with *true* or *false* indicating whether it will be reused later.

Each client module reports locally-used applications to the server based on their CPU times, recent working directories, and window titles of the processes. The threshold values of the CPU time are changed every 10 days, assigning different values to different application types. For example, for the Reader type, we assigned the threshold values of the CPU time t = 0, 5, 10, 30, 60, 90, 180 (sec), and for the Editor type, we assigned t = 0, 10, 20, 30, 60, 120, 300 (sec). The information about the file name and the full path of a process is extracted from the active window's title and the shortcut file in the "Recent" directory.

The client module gathers satisfaction and accuracy information to check the user's reuse intention for every proactively executed file via a popup window as shown in **Fig. 6**.

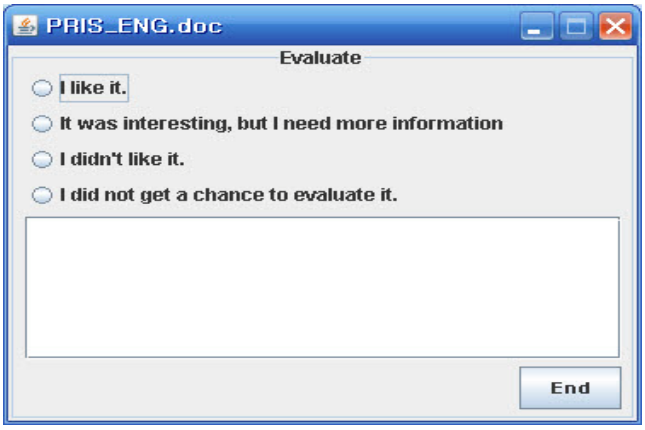
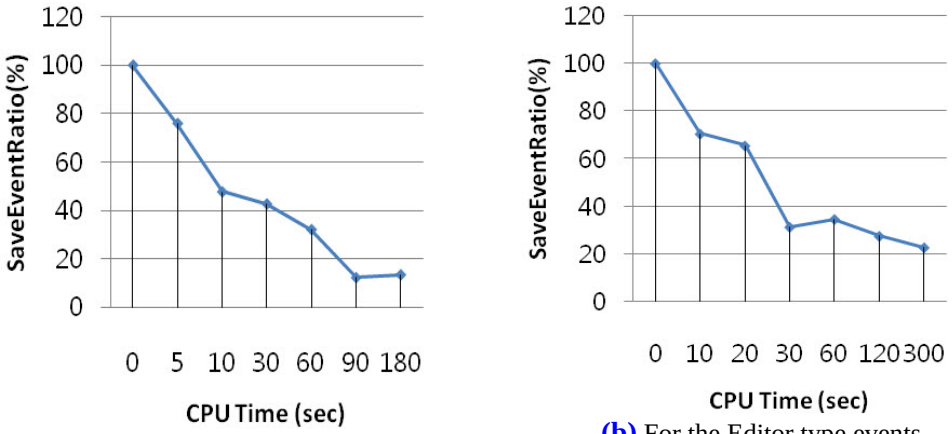


Fig. 6. A popup window for checking user satisfaction

5.3 Evaluation

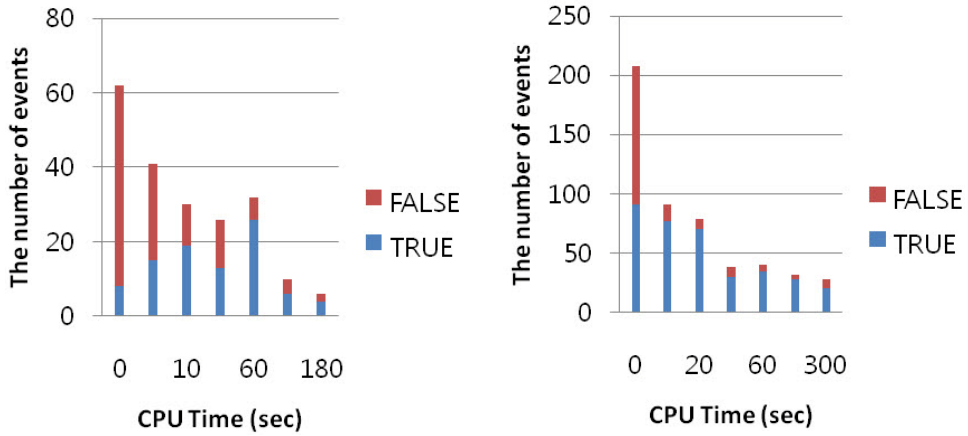
Fig. 7-(a) and **7-(b)** show the ratio of saved events according to the CPU time threshold for the Reader and Editor types, respectively. As shown in these figures, the number of saved events in the server is decreasing as the CPU time threshold is increasing.

Fig. 8-(a) and **8-(b)** show the ratio of satisfying events denoted by *TRUE* and unsatisfying events denoted by *FALSE* for the Reader and Editor types, respectively. Note that the red bar(upper part) in the graphs represents the number of unsatisfying events, and the blue bar(lower part) indicates the number of satisfying events. As CPU time threshold increases, the ratio of satisfying events over all saved events also increases. Naturally, the ratio of unsatisfying events is decreasing as the threshold is increasing.



(a) For the Reader type events **(b)** For the Editor type events

Fig. 7. The ratio of saved events for the Reader and Editor types



(a) For the Reader type events **(b)** For the Editor type events

Fig. 8. The ratio of satisfying and unsatisfying events for the Reader and Editor types

These evaluation results can be explained as follows: When the CPU time threshold is very low, all applications easily exceed the threshold value, and consequently, many events are stored into the server regardless of user intentions. Consequently, the ratio of unsatisfying events would be high. On the other hand, a higher threshold causes only those events spending more CPU time to be stored, resulting in higher ratio of satisfying events.

To evaluate the performance of proactive execution in our PES system, we adopted a satisfaction degree and accuracy measure that has been used in machine learning and classification society. In this paper, we measured the satisfaction and accuracy for proactive execution of our system using the following formulas.

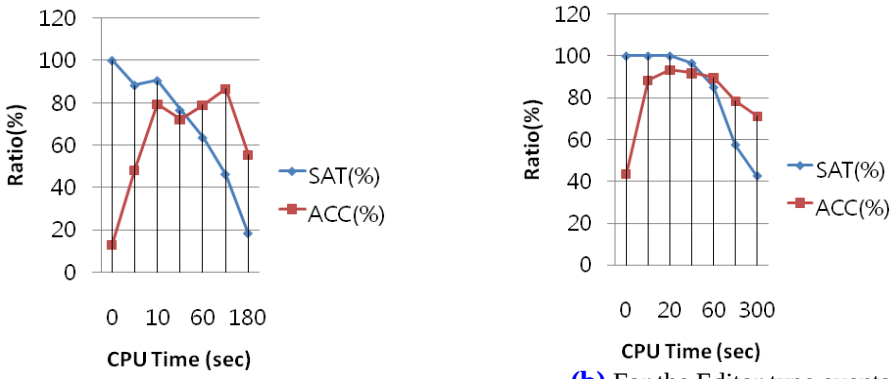
$$SAT = \frac{\#SavedTrueEvent}{\#TotalTrueEvent} \tag{1}$$

Equation (1) measures the satisfaction degree by calculating the ratio of stored satisfying(*TRUE*) events over the total satisfying(*TRUE*) events.

$$ACC = \frac{\#SavedTrueEvent+\#UnsavedFalseEvent}{\#TotalEvent} \tag{2}$$

Equation (2) measures the accuracy of proactive execution by the ratio of the sum of saved satisfying(*TRUE*) events and unsaved unsatisfying(*FALSE*) events over the total events.

Fig. 9-(a) and **9-(b)** show the satisfaction and accuracy measures for all events according to the CPU time threshold for the Reader and Editor types, respectively.

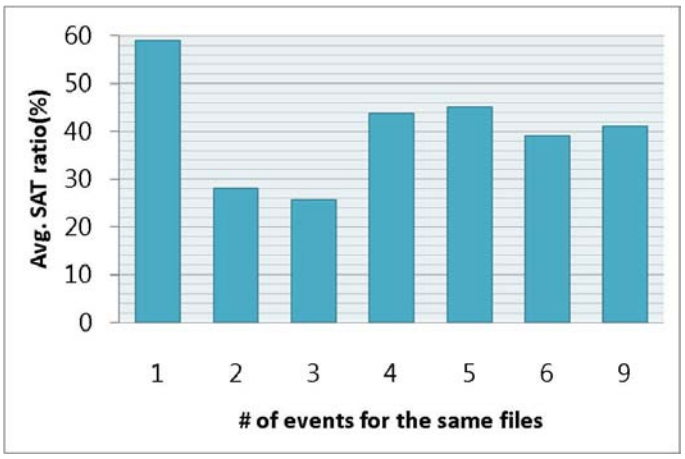


(a) For the Reader type events **(b)** For the Editor type events

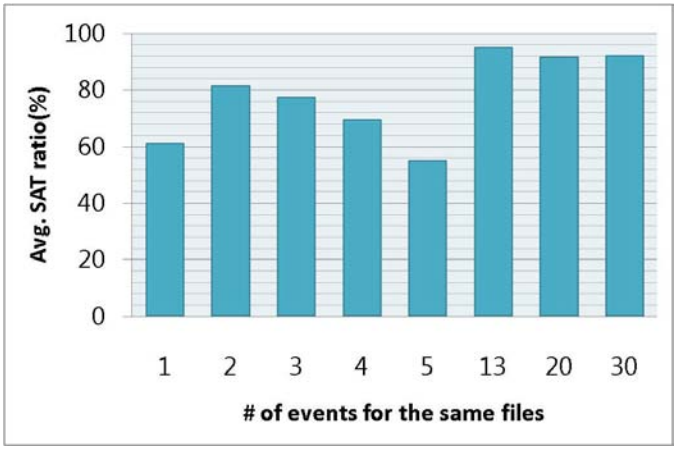
Fig. 9. Satisfaction and accuracy measures for the Reader and Editor type events

Note that the satisfaction degree is decreasing as the CPU time threshold is increasing because a low threshold caused more events to be saved into the server that resulted in lower possibility of finding files that will be reused by the user. Moreover, we could see that the accuracy increases for low CPU time thresholds, but starts to decrease at some point (90 sec. for the Reader type and 20 sec. for the Editor type) when the rate of the saved *TRUE* events is decreasing.

Fig. 10-(a) and **10-(b)** show the number of repeating events for the same files for the Reader and the Editor types, respectively, with their satisfaction ratios. As shown in **Fig. 10-(a)**, the satisfaction ratio is relatively low (i.e., the highest ratio is below 60%), and the average is 40.1%. This data indicates that most of the Reader type documents were used once rather than repeatedly and continuously used. On the other hand, the average satisfaction ratio of the Editor type is 77.83%, as shown in **Fig. 10-(b)**, especially the average ratio of the events which occurred more than 10 times is very high with 92.9%. This data implies that the documents of the Editor type were repeatedly and continuously used. In fact, the Editor type documents were very frequently modified by the users who own them.



(a) For the Reader type events



(b) For the Editor type events

Fig. 10. Average satisfaction ratio for the Reader and Editor type events

Table 2 and **Fig. 11** show the number of events occurred in a day. According to these data, the participants used 7.68 Reader type documents and 18.595 Editor type documents in a day, and hence each participant used 1.28 Reader type documents and 3.1 Editor type documents, in average. The number of participants is rather small, but this result is enough to tell us that the number of events occurring in a day is not large. Therefore, we claim that the efficiency of the PES system depends on the size of the used files rather than the number of events, which is demonstrated by the subsequent figures and tables.

Table 2. The average number of events in a day

Week / Type	1st	2nd	3rd	4th	average
Reader	7.14	7.29	8.29	8	7.68
Editor	18.58	18.7	18	19.1	18.595

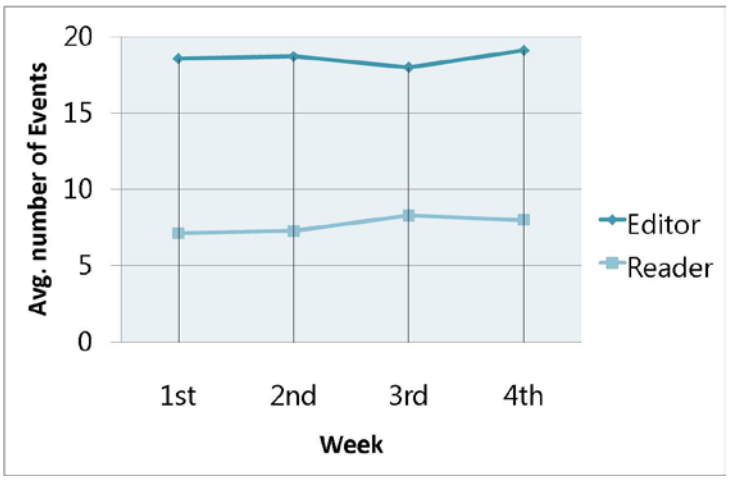


Fig. 11. The average number of events in a day

Table 3 shows the size of used files by all participants in a day. The data represents the required space of the server in order to save all the used files in a day. As shown in this table, 5.76MB and 13.31MB are used for the Reader type and the Editor type events, respectively, in average. These spaces of the server are only used to save the used files rather than events.

Table 3. The size of used files in a day (Kbyte)

Week / Type	1st	2nd	3rd	4th	average
Reader	3676.49	5430.03	7551.514	6406.89	5766.23
Editor	10900.93	19883.35	7897.699	14557.3	13309.82

Table 4 shows the amount of spaces of the server in a day needed for the Reader type and the Editor type events, including the average number of events, the average amount of spaces for all events, and the average amount of spaces for all the used files.

In this paper, the size of an event is 1070 byte, and by multiplying the event size to the average number of events, we can get the average amount of spaces for all events which are 8.025 Kbyte and 19.43 Kbyte for the Reader type and the Editor type events, respectively. This space size excludes the space for the used files, and therefore, if we add the spaces that are required for the used files then the space sizes become 5.64 MB and 13.017 MB, respectively. Of course, these sizes should be changed according to the sizes of the used files, but through this experiment, we claim that our PES server was efficiently operating by using very small amount of spaces in a day (about 19MB).

Table 4. The amount of spaces of the server in a day

Type	Avg. num. of events	Avg. amount of spaces for all events (Kbyte)	Avg. amount of spaces for all used files (Kbyte)	Total (Mbyte)
Reader	7.68	8.025	5766.23	5.64
Editor	18.595	19.43	13309.82	13.017

To explain the efficiency of the PES system, **Table 5** and **Table 6** show the status of resource consumption during intention recognition for the Reader type and the Editor type, respectively. *File Size* is size of the used data files that we classified into 6 categories: below 10KB, below 100KB, below 500KB, below 1MB(1000KB), below 2MB, and below 10MB. Because the amount of transmittable data at a time is limited (about 15MB) using Java RMI, very large files are not dealt with in this paper.

The analysis of the data in **Table 5** and **Table 6** indidates that intention recognition is executed in a short period of time (i.e., 0.73 sec. for the Reader type and 1.09 sec. for the Editor type, in average), and the time is increasing proportional to the file size because larger files need more time for file transmission. Moreover, the amounts of CPU and memory usages during intention recognition are very small regardless of the event types. As the file size is increased, more memory spaces are required, because memory spaces are used to transmit the used files to the server.

Table 5. The status of resource consumption during Intention Recognition(*IR*) for the *Reader* type

File Size Type (Kbyte)	Avg. File Size (Kbyte)	Avg. IR Time (sec)	Avg. CPU Usage (%)	Avg. Memory Usage	
				Avg. Heap Mem Usage (Kbyte)	Avg. Non-Heap Mem Usage (Kbyte)
<10	4.204	0.514	0.565	21.091	0.31
<100	51.189	0.553	0.589	64.69	0.22
<500	310.094	0.610	0.574	591.63	0.38
<1000	690.348	0.675	0.607	1112.39	0.37
<2000	1431.94	0.694	1.149	1889.82	0.31
<10000	4002.33	1.392	0.59	3875.86	0.59

Table 6. The status of resource consumption during Intention Recognition(*IR*) for the *Editor* type

File Size Type (Kbyte)	Avg. File Size (Kbyte)	Avg. IR Time (sec)	Avg. CPU Usage (%)	Avg. Memory Usage	
				Avg. Heap Mem Usage (Kbyte)	Avg. Non-Heap Mem Usage (Kbyte)
<10	2.39	0.891	0.16	22.41	0.139
<100	41.48	0.924	0.26	59.19	0.282
<500	208.17	0.999	0.8	400.24	0.287
<1000	782.17	1.056	0.51	1259.59	0.529
<2000	1405.24	1.278	0.08	1706.11	0.312
<10000	3467.83	1.429	0.88	1182.12	1.792

Table 7 and **Table 8** show the status of resource consumption during proactive execution for the Reader type and the Editor type, respectively. Proactive execution proceeds in two phases; the monitoring phase and the file execution phase. In the monitoring phase, PES checks whether the files that are used before in other devices by the user are reused in the current device. As mentioned in Section 4, the purpose of this phase is to detect the time when the user executes the application. In the file execution phase, PES executes the files by launching compatible applications.

The analysis of the data in **Table 7** and **Table 8** indicates that proactive execution is also accomplished in a short period of time (i.e., 0.329 sec. for the Reader type and 0.914 sec. for the Editor type, in average). For the Reader type events, proactive execution consumes very small amount of CPU and memory resources, whereas the Editor type events need relatively

large amount of resources for proactive execution due to the characteristics of the Editor type applications programs.

Table 7. The status of resource consumption during Proactive Execution(*PE*) for the *Reade*r type

File Size Type (Kbyte)	Avg. File Size (Kbyte)	Avg. PE Time (ms)		Avg. CPU Usage (%)		Avg. Mem Usage			
		Monitoring	File Execution	Monitoring	File Execution	Avg. Heap Mem Usage (Kbyte)		Avg. Non-Heap Mem Usage (Kbyte)	
						Monitoring	File Execution	Monitoring	File Execution
<10	5.64	293.8	13.7	0.313	1.33	39.81	2.96	8.17	0.93
<100	58.11	316.4	11.8	0.169	1.604	40.09	1.51	12.61	1.52
<500	303.04	323.6	11.5	0.246	1.051	472.62	9.86	12.59	1.54
<1000	764.14	322.1	11.4	0.31	1.149	424.78	3.85	13.43	1.65
<2000	1356.47	317.2	11.1	0.18	1.464	376.83	8.39	11	1.39
<10000	4163.76	326.2	12.6	0.18	1.12	451.29	5.89	6.84	0.79

Table 8. The status of resource consumption during Proactive Execution(*PE*) for the *Editor* type

File Size Type (Kbyte)	Avg. File Size (Kbyte)	Avg. PE Time (ms)		Avg. CPU Usage (%)		Avg. Mem Usage			
		Monitoring	File Execution	Monitoring	File Execution	Avg. Heap Mem Usage (Kbyte)		Avg. Non-Heap Mem Usage (Kbyte)	
						Monitoring	File Execution	Monitoring	File Execution
<10	4.41	322.2	10.64	0.026	0.56	669.27	9.13	7.052	0.724
<100	41.38	805.98	39.38	0.93	3.46	263.53	7.31	20.25	1.4
<500	255.29	1278.2	64.37	2.19	4.23	269.96	8.64	15.41	0.03
<1000	813.23	875.92	61.38	2.02	3.84	265.84	11.24	18.59	1.3
<2000	1389.82	890.4	70.3	1.09	3.71	264.44	10.69	17.58	0.089
<10000	4585.98	1039	29.13	1.92	5.27	265.18	12.37	21.59	16

6. Conclusions

We have proposed an intelligent technique for data sharing and event transition among personal devices of a user that proactively executes appropriate tasks across multiple devices by recognizing user intention based on the observed actions of the user for specific devices. A series of experiments supports that the PES system we have implemented can provide appropriate personalized services with a high degree of satisfaction to the user in an effective and efficient manner.

As a future work, we are planning to personalize the CPU time threshold by assigning different threshold values to different users according to their personal working styles and preferences. We are also seeking a method of applying the personalized CPU time threshold to event monitoring by dynamically updating the threshold for each user.

References

[1] T. Pessemier, T. Deryckere, K. Vanhecke and L. Martens, "Proposed architecture and algorithm for personalized advertising on iDTV and mobile devices," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 2, pp. 709-713, 2008. Article (CrossRef Link)

[2] H. Lee and J. Kim, "An approach for content sharing among UPnP devices in different home networks, *IEEE Transactions on Consumer Electronics*, vol. 53, no. 4, pp. 1419-1426, 2007. Article (CrossRef Link)

[3] T. Nawaz, M. Mian and H. Habib, "Infotainment devices control by eye gaze and gesture recognition fusion," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 2, pp. 277-282, 2008. Article (CrossRef Link)

[4] H. Shin, M. Lee and E. Kim, "Personalized digital TV content recommendation with integration of user behavior profiling and multimodal content rating," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 3, pp. 1417-1423, 2009. Article (CrossRef Link)

[5] R. Want, T. Pering and D. Tennenhouse, "Comparing autonomic and proactive computing," *IBM Systems Journal*, vol. 42, no. 1, pp. 129-135, 2003. Article (CrossRef Link)

[6] A. Salovaara and A. Oulasvirta, "Six modes of proactive resource management: A user-centric typology for proactive behaviors," in *Proc. of 3rd Nordic Conf. on Human-Computer Interaction*, pp. 57-60, 2004. Article (CrossRef Link)

[7] O. Schrempf, U. Hanebeck, A. Schmid and H. Worn, "A novel approach to proactive human-robot cooperation," in *Proc. of IEEE International Workshop on Robot and Human Interactive Communications*, pp. 555-560, 2005. Article (CrossRef Link)

[8] E. Horvitz, J. Breese, D. Heckerman, D. Hovel and K. Rommelse, "The Lumiere project : Bayesian user modeling for inferring the goals and needs of software users," in *Proc. of 14th Conf. on Uncertainty in Artificial Intelligence*, pp 256-265, 1998. Article (CrossRef Link)

[9] J. Budzik and K. Hammond, "User interactions with everyday applications as context for just-in-time information access," in *Proc. of International Conf. on Intelligent User Interfaces*, pp. 44-51, 2000. Article (CrossRef Link)

[10] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman and E. Ruppin, "Placing search in context: The concept revisited," *ACM Transactions on Information Systems*, vol. 20, no. 1, pp. 116-131. 2002. Article (CrossRef Link)

[11] L. Hongen, "A proactive web agent for information browsing and extracting," in *Proc. of 6th Asia-Pacific Web Conf.*, pp 879-882, 2004. Article (CrossRef Link)

[12] T. Joachims, D. Freitag and T. Mitchell, "WebWatcher: A tour guide for the World Wide Web," in *Proc. of International Joint Conf. on Artificial Intelligence*, pp. 770-775, 1997. Article (CrossRef Link)

[13] H. Lieberman, "Letizia: An agent that assists web browsing," in *Proc. of International Joint Conf. on Artificial Intelligence*, pp.475-480, 1995. Article (CrossRef Link)

[14] W. Zhang and J. Wu, "A learning search based intelligent assistant of Web browser," in *Proc. International Conf. on Wireless Communications, Networking and Mobile Computing*, pp. 5629-5631, 2007. Article (CrossRef Link)

[15] N. Stefanov, A. Peer and M. Buss, "Online intention recognition in computer-assisted teleoperation systems," in *Proc. International Conf. on Haptics: Generating and Perceiving Tangible Sensations*, pp. 233-239, 2010. Article (CrossRef Link)

[16] O. Schrempf, D. Albrecht and U. Hanebeck, "Tractable probabilistic models for intention recognition based on expert knowledge," in *Proc. of IEEE/RSJ International Conf. on Intelligent Robots and Systems*, pp. 1429-1434, 2007. Article (CrossRef Link)

[17] L. Pereira, and T. Han, "Intention recognition via causal bayes networks plus plan generation," in *Proc. of the 14th Portuguese International Conf. on Artificial Intelligence*, pp. 138-149, 2009. Article (CrossRef Link)

[18] P. Kiefer, and C. Schlieder, "Exploring context-sensitivity in spatial intention recognition," in *Proc. of Workshop on Behavior Monitoring and Interpretation*, pp. 102-116, 2007. Article (CrossRef Link)

[19] L. Pereira and H. Anh., "Elder care via intention recognition and evolution prospection," in *Proc. of the 18th International Conf. on Applications of Declarative Programming and Knowledge Management*, 2009. Article (CrossRef Link)
[20] B. Rhodes and P. Maes, "Just-in-time information retrieval agents," *IBM Systems Journal*, vol. 39, no. 3-4, pp. 685-704, 2000. Article (CrossRef Link)
[21] J. Hong and J. Landay, "A context/communication information agent," *Personal and Ubiquitous Computing*, vol. 5, no. 1, pp. 78-81, 2001. Article (CrossRef Link)
[22] P. Chirita, C. Firan and W. Nejdl, "Pushing task relevant web links down to the desktop," in *Proc. of the 8th ACM International Workshop on Web Information and Data Management*, pp. 59-66, 2006. Article (CrossRef Link)
[23] H. Holz, H. Maus, A. Bernardi and O. Rostanin, "From lightweight, proactive information delivery to business process-oriented knowledge management," *Journal of Universal Knowledge Management*, vol. 0, no. 2, pp. 101-127, 2005. Article (CrossRef Link)
[24] D. Billsus, D. Hilbert and D. Maynes-Aminzade, "Improving proactive information systems," in *Proc. of the 10th International Conf. on Intelligent User Interfaces*, pp. 159-166, 2005. Article (CrossRef Link)
[25] K. Wickramasinghe, C. Guttmann, M. Georgeff, H, Gharib, I. Thomas, S. Thompson and H. Schmidt, "Agent-based intelligent collaborative care management," in *Proc. the 8th International Conf. on Autonomous Agents and Multiagent Systems*, vol. 2, pp. 1387-1388, 2009. Article (CrossRef Link)

Hocheol Jeon received his M.S. degree in Computer Science and Engineering from Hanyang University, Korea, in 1999. He is currently a doctoral student at the Computer Science and Engineering department, Hanyang University, Ansan, Korea. His research interests include intelligent agent, information retrieval, information extraction, and context-aware user modeling.

Taehwan Kim received his M.S. degree in Computer Science and Engineering from Hanyang University, Korea, in 2007. He is currently a doctoral student at the Computer Science and Engineering department, Hanyang University, Ansan, Korea. His research interests include Web data mining, Web intelligence, artificial intelligence, and Semantic Web & ontology.

Joongmin Choi is a professor in the Department of Computer Science and Engineering, Hanyang University, Korea. He is the director of Web Intelligent Consortium (WIC) Korea Center. He received his B.S. and M.S degree in computer engineering from Seoul National University, Korea, in 1984 and 1986, respectively, and the Ph.D. degree in computer science from the State University of New York at Buffalo, USA, in 1993. His research interest focuses on Web Intelligence, which is a somewhat broad concept covering the areas of Web information extraction, Web data mining, Semantic Web and ontology, and other intelligent techniques for manipulating Web information.