IEICE TRANS. INF. & SYST., VOL.E94–D, NO.7 JULY 2011

PAPER

Personal Event Management among Multiple Devices Based on User Intention Recognition Using Dynamic Bayesian Networks

Hocheol JEON[†a)], **Taehwan KIM**[†b)], *Nonmembers*, *and* **Joongmin CHOI**[†c)], *Member*

SUMMARY This paper proposes a proactive management system for the events that occur across multiple personal user devices, including desktop PCs, laptops, and smart phones. We implemented the Personal Event Management Service using Dynamic Bayesian Networks (PEMS-DBN) system that proactively executes appropriate tasks across multiple devices without explicit user requests by recognizing the user's device *reuse intention*, based on the observed actions of the user for specific devices. The client module of PEMS-DBN installed on each device monitors the user actions and recognizes user intention by using dynamic Bayesian networks. The server provides data sharing and maintenance for the clients. A series of experiments were performed to evaluate user satisfaction and system accuracy, and also the amounts of resource consumption during intention recognition and proactive execution are measured to ensure the system efficiency. The experimental results showed that the PEMS-DBN system can proactively provide appropriate, personalized services with a high degree of satisfaction to the user in an effective and efficient manner.

key words: *personal event management service (PEMS), user intention recognition, proactive task execution, dynamic Bayesian networks (DBN)*

1. Introduction

Recently, people are increasingly using multiple personal devices, including desktop PCs, notebooks, smart phones, and MP3 players, and many of them are working with these devices in different places, such as home, office, school, or public transportation. More often, there are situations when users want to seamlessly use the contents in multiple devices to achieve continuity of tasks by using data and status information transition. For example, it is common that users who use their office PCs during the daytime want to continue their work using home PCs at night, or on their way home in the subway train using smart phones. Also, users want an MP3 or video file played on the PC to be played again on their smart phones from the point at which it was suspended. However, some manual preparatory work which is necessary to accomplish these tasks could be complicated and inefficient, especially if more devices are involved.

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 the cumbersome preparatory work. The main idea of our proposed system is to proactively execute tasks by monitoring user behavior and probabilistically recognizing the user's *device reuse intention* [16], [17]. The data obtained as a result of the user intention recognition is shared among personal devices so that the user task performed in one device can be continued seamlessly in other devices. A preliminary work for user intention recognition and proactive execution has been proposed and implemented as the Personal Event Service (PES) system [5], and this paper proposes an enhanced method using dynamic Bayesian networks to probabilistically recognize the user's reuse intention.

We have implemented the Personal Event Management Service using Dynamic Bayesian Networks (PEMS-DBN) system, which proactively executes appropriate tasks across multiple devices without explicit user requests by sharing the data used by the user and by recognizing the user intention based on the observed actions of the user for specific devices. The client module of PEMS-DBN installed on each device monitors the user actions and recognizes the user intention using dynamic Bayesian networks. The server manages the shared data and events and provides maintenance for the clients. A series of experiments were performed to evaluate user satisfaction and system accuracy, and the results of these experiments showed that PEMS-DBN can provide appropriate, personalized services with high satisfaction to the user. Also, to ensure the system efficiency and effectiveness, we measured the amounts of resource consumption during intention recognition and proactive execution, and the results show that all of the processes are executed with minimum overload to the system.

2. Related Work

In recent decades, a number of studies in various fields have been devised to recognize user intentions based on the observation of user behaviors, implicit user feedback, and some context information, and to predict the users' next actions and proactively execute suitable tasks without user intervention. Some of related studies include the following: 1) a method to improve the performance of the user intention recognition and to minimize the uncertainty in a desktop environment [6], 2) a probabilistic approach to provide advanced assistance and guidance to users while the users interact with Microsoft Office applications [4], 3) a system to use the users' context information such as recognizing currently opened documents to monitor web browsing behaviors [2], [3], 4) a system to serve users with appropriate assistance in computer-assisted teleoperation [8], [15], 5) a

Manuscript received October 25, 2010.
Manuscript revised February 2, 2011.

[†]The authors are with the Department of Computer Science and Engineering, Hanyang University, Ansan, Korea.
a) E-mail: hochuls@chol.com
b) E-mail: kimth@islab.hanyang.ac.kr
c) E-mail: jmchoi@hanyang.ac.kr

DOI: 10.1587/transinf.E94.D.1440

proactive approach to resource management [10], and 6) an intelligent information agent to collect user preferences and learn these preferences to plan and reason proactively [9].

As robot utilization is increasing, an understanding of user intention is a very worthy and important issue in the robotics field to ensure intelligent behaviors through interactions with users. Reflecting this trend, there are a number of studies in the past few years for recognizing user intentions from a robot perspective [1], [11]–[13].

The dynamic Bayesian network (DBN) is a representative way that can express and infer knowledge for an environment with insufficient information. A system using the DBN model can appropriately respond to uncertain context information and provide semantic representation for objects and context information for the context-aware service. In the past few years, many methods have been proposed to probabilistically recognize user intention [6], [7], [14], [16].

Most of these techniques can be performed on only one device, and the events that have happened at other devices do not affect any user behaviors at the current device. In our study, however, we considered all of the devices of each user, and PEMS-DBN can execute events that have happened at other devices when the reuse intention for the event is recognized. In other words, when a sign for an event is observed, PEMS-DBN is able to perform the estimated next behavior of the user.

3. System Architecture

Figure 1 shows the system architecture of PEMS-DBN which employs a client-server structure. The server side consists of *Server Event Manager*, *DB Manager*, and *Server Data Manager*. The client side consists of *Client Data Manager*, *Client Monitoring Manager*, *Client Intention Recognizer*, *Client System Manager*, *Client Estimator*, and *Preference Manager*.

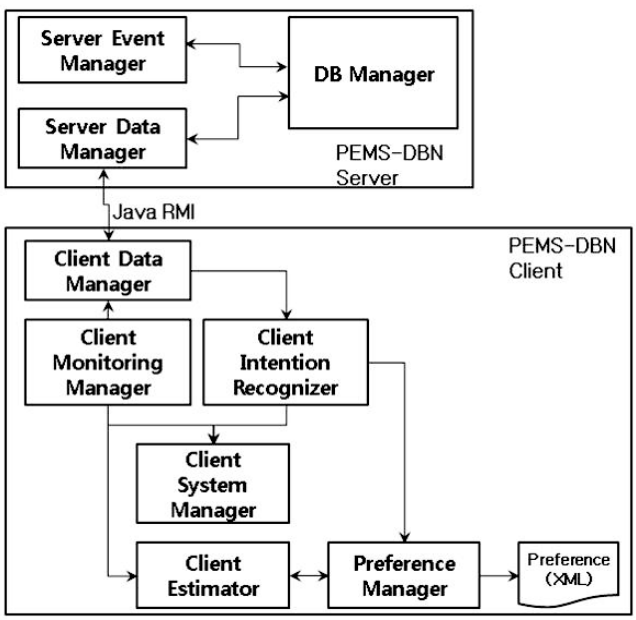

Fig. 1 System architecture of PEMS-DBN.

The *Server Event Manager* executes periodically, deleting expired event information from the database and updating the usability of devices. The event information that is periodically received from a client through the *Server Data Manager* is stored in the database by the *DB Manager*, which also manages the user information, device information, and task information for each device. The *Server Data Manager* manages the connection with a client, stores monitoring results to the database, and saves the data files used by applications to the server file system. The network connection between the server and each client is implemented using Java Remote Method Invocation (RMI). Through this connection, a client periodically sends the data of the applications that are executed by the user to the *Server Data Manager*, which stores this data to the database.

The *Client Data Manager* requests and manages the connection with the server. It receives information about the events performed by the user in other devices from the server, and sends the monitoring results, which are gathered periodically, to the server. The *Client Intention Recognizer* monitors the current processes to recognize the reuse intention of the user, gathers information about the data files and processes that are highly likely to be reused, and transmits them to the server through the *Client Data Manager*. The reuse probability, which is calculated by the *Client Estimator*, varies according to various environmental factors, including the event type, the time that the event occurs, and the device on which the event occurs. The *Client Monitoring Manager* constantly monitors the processes corresponding to the events that are performed by the user in the current device. When the intention is detected, the *Client Monitoring Manager* proactively executes the event. Afterward, the *Preference Manager* updates the preference information for each situation according to whether the event is reused or not. The *Client System Manager* finds the applications that can be monitored in the device and writes the absolute paths of the execution files in XML format.

To detect user's reuse intention, the *Client Estimator* and the *Preference Manager* play important roles. The *Client Estimator* estimates the CPU time with a maximum probability value for each event by considering the environment and context information, accesses the preference file via the *Preference Manager*, and calculates the probability value of the reuse intention for the currently used processes and files using the probability density function.

The functioning of the PEMS-DBN system can be explained by the following scenario for data and event sharing among personal devices, including a home PC, an office PC, and a smart phone. In this scenario, assume that Mr. Kim is a computer 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.

4. Intention Recognition by Using Dynamic Bayesian Network

4.1 A Dynamic Bayesian Network for Reuse Intention

We use the DBN to perceive the user's working behavior for each device and to calculate the probability value of the reuse intention for an event depending on the observed actions.

In general, we should perform some preparatory work to reuse the files or continue to do the work at another place. For example, users may copy or move files into an external storage device, such as a USB, upload the files via P2P applications, send an e-mail and attach the files by using e-mail applications, or execute remote access applications, such as pcAnyWhere or CarbonCopy. These actions are signs that imply the user's reuse intentions. In addition, these actions are repeated in order to continuously do the work at different places. Since the PEMS-DBN system regards the execution of the application as a sign for the given event, the PEMS-DBN Client proactively executes the event when the sign is observed. Based on these observations, we have built a DBN that is used in PEMS-DBN, as shown in Fig. 2.

In this DBN, we define 2 types of domain knowledge, 4 types of context knowledge, 1 intention which is the reuse intention, 5 actions, and 5 measurement processes.

Domain knowledge covers the environmental influences, and includes *Time* and *Device*. *Time* indicates the time of the event creation, and can be assigned one of {*morning*, *afternoon*, *night*}. *Device* denotes the place at which an event occurs and PEMS-DBN deals with 5 device types including desktop PCs, notebooks, smart phones, mobile players, and IPTVs. The reason why we used coarse-grained indicators such as morning or afternoon rather than fine-grained indicators such as 1pm or 7am is that we needed to maintain small number of cases to quickly calculate the probability density value for the reusability of the file. In other words, we needed a set of discrete values to apply observed values to the Bayes' rules more easily, and coarse-grained indicators can reduce possible number of cases.

Context knowledge reflects the influence of the internal state of the device on which the event is occurring, and includes some factors that can affect user satisfaction about proactive execution for the given event. The types of context knowledge are *Application Type*, *CPU Time*, *Repeat*, and *Co-Execute Application*. Here, *Application Type* is the type of application program, and in PEMS-DBN, there are 4 types, including *Editor*, *Reader*, *Viewer*, and *Player*. Table 1 shows some example application programs that belong to each application type with a description of the program characteristics. *CPU Time* is the CPU running time of a process during the interaction between a user and an application. *Repeat* indicates that the same event occurs repetitively and continuously at the same device or several devices, and it means the work is done over a long period of time. *Co-*

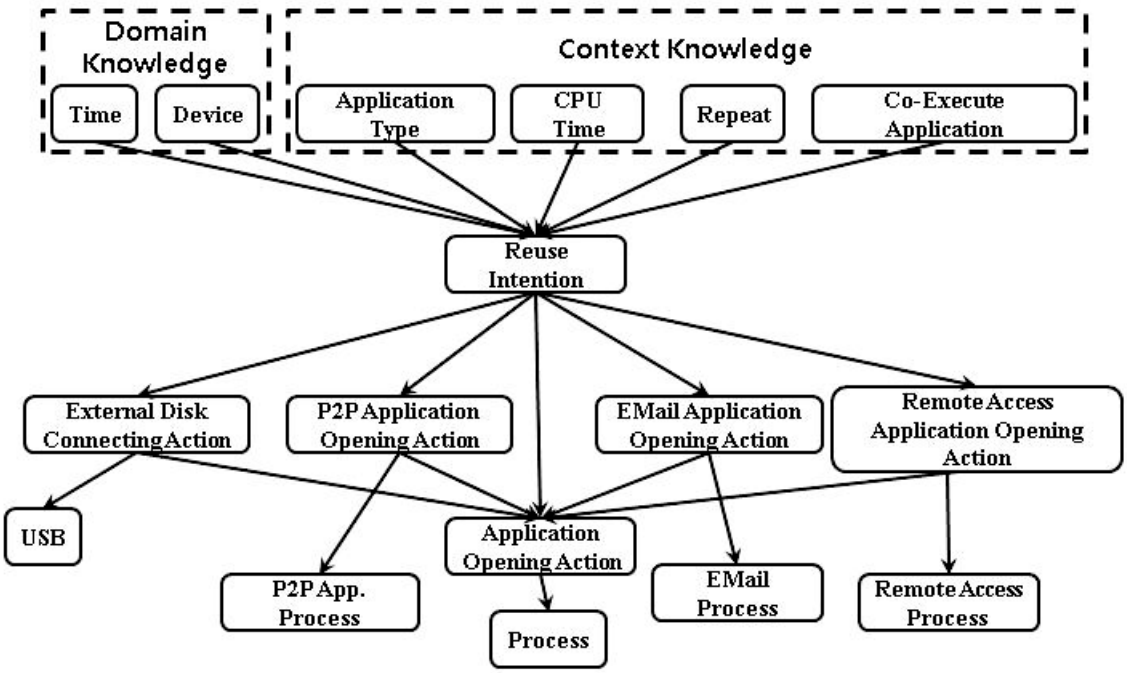

Fig. 2 A DBN used in the PEMS-DBN system.

Table 1 Application types.

Type	Applications	Description
Editor	Hwp (Korean word processor), MS-Word, MS-PowerPoint, MS-Excel, Notepad, UltraEdit, ERWin, StarUML	Readable and writable applications
Reader	Acrobat Reader, Ghost View	Only readable applications
Viewer	MSPaint, AlSee	Only viewable applications
Player	Window Media Player, Gom Player, Gom Audio	Only playable applications

Execute Application means multi-tasking, such as listening to music while doing word processing. It is used to reflect the influence of multi-tasking on the satisfaction degree of the proactive execution.

As mentioned before, there is one intention, the reuse intention, and to recognize this intention, 5 actions are used: External Disk Connecting Action, P2P Application Opening Action, EMail Application Opening Action, Remote Access Application Opening Action, and Application Opening Action.

These five actions are used in the intention recognition phase by representing the occurrence of each of them as an element value of the measurement vector. Thus, the measurement vector has five elements, each of which denotes the occurrence of each action. The vector is used as a sign to recognize the reuse intention, and the value of each vector element is set to 0 initially and becomes 1 at the occurrence of the corresponding action. For example, when the 'External Disk Connecting Action', which denotes the action of the user's connecting a USB to a personal device, is observed by the monitoring processes, the measurement vector becomes $\langle 1, 0, 0, 0, 0 \rangle$.

4.2 Events

In PEMS-DBN, an event is a fundamental processing unit to represent user behavior. We define an event E as $E = \langle D, C, F, U \rangle$, where D is domain knowledge, C is context knowledge, F is file information, and U is user information.

D includes the time and device information, consisting of the time at which the event occurs and the unique sequential number of the device. C is defined as $C = \langle AID, CPUTime, CoExec, Repeat \rangle$, where AID is the application id, $CPUTime$ is the running time of the application, $CoExec$ indicates whether different types of applications are being executed, and $Repeat$ indicates whether repetitive events occur for the file. F is defined as $F = \langle name, path, file \rangle$, where $name$ is the name of the used data file, $path$ is the local absolute path of the file, and $file$ is a byte array corresponding to the data file. User information includes personal details such as the user id. The same events that have occurred on a device by the same user are continuously updated to ensure the consistency of file contents.

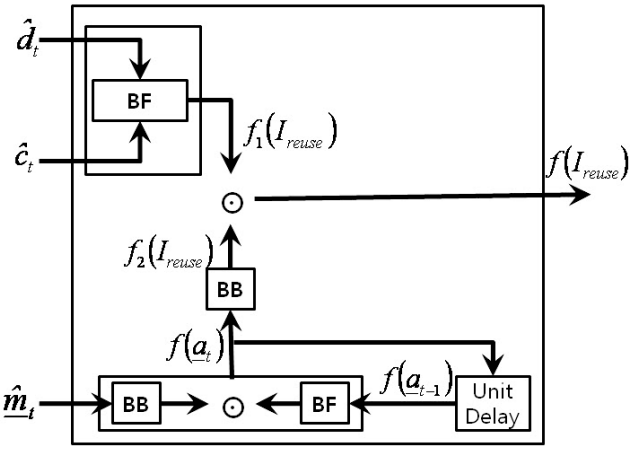

Fig. 3 The Client Estimator for calculating probability density.

4.3 Intention Recognition

User intention recognition focuses on *what*, boiling down to the problem of "what events are reused by the user?". We used two conditions to recognize the reuse intention of users for a given event: one is that the CPU Time of a process must be greater than the appropriate CPU Time that was calculated by the *Client Estimator*, and the other is that the probability density function $f(I_{reuse})$ for the reuse intention has to satisfy the predefined threshold condition. An event is created only if a process satisfies these two conditions, and then it is registered to the server and proactively executed by the PEMS-DBN Client.

The *Client Estimator* archives the indices for CPU Time candidates by using the domain knowledge and context knowledge, and finds an appropriate CPU Time with the maximum reuse probability value for the archived indices. The model used by our *Client Estimator* to calculate the probability density function is shown in Fig. 3, which adopts and modifies the model defined in [11]. It computes a probability density over the reuse intention given the measurement vector $\underline{\hat{m}}_t$, the domain knowledge $\hat{d}_t$, and context knowledge $\hat{c}_t$. The BF and BB blocks indicate the Bayesian forward and Bayesian backward inference, respectively. In order for an event to be created for a process, the probability density $f(I_{reuse})$ for the process has to satisfy the condition $f(I_{reuse}) \geq 0.5$ or $|f(I_{reuse}) - (1 - f(I_{reuse}))| \leq \alpha$ for the threshold α.

The condition $f(I_{reuse}) \geq 0.5$ implies that the reuse probability $f(I_{reuse})$ is larger than the no-reuse probability $1 - f(I_{reuse})$, so the event is created naturally. The other condition $|f(I_{reuse}) - (1 - f(I_{reuse}))| \leq \alpha$ is dealing with a situation when the reuse probability is smaller than the no-reuse probability but the absolute difference between them is extremely slight. For example, if the reuse probability is 0.49, the no-reuse probability must be 0.51, and the absolute difference between them is 0.02 which is regarded as negligible. In this case, we expect that there is a chance of the file being reused in other devices, so the event is created as

Algorithm 1 Reuse Intention Recognition

```
procedure IntentionRecognition()
    e[] = getEvents();
    user-info = loadUserInfo();
    device-info = loadDeviceInfo();
    for each event e do
        p[] = getProcesses(e);                    ▷ executing processes for e
        for each process p do
            t = findMAXCPUTime();
            d = estimateProbDensity(p);
            f = usedDataFile(p);                  ▷ data file used in p
            if CPUTime(p) ≥ t and d ≥ α then      ▷ α is threshold
                path = findPath(f);               ▷ directory path of f
                usedFile = convertToByteArray(path);
                sendFileToServer(usedFile);
                sendEventToServer(e, user-info, device-info);
            end if
        end for
    end for
end procedure
```

Algorithm 2 Proactive Execution

```
procedure ProactiveExecution()
    e[] = getEventsOtherDevices();
                    ▷ get events happened in other devices previously
    p[] = getJustExecutedProcesses();
                    ▷ get processes just executed in the current device
    for each process p do
        for each event e do
            if processName(e) == processName(p) then
                if the same program k exists for p then
                    executeProcess(findPath(k), e);
                else if there is a compatible program m for p then
                    executeProcess(findPath(m), e);
                else
                    displayWarningWindow(e);
                end if
            end if
        end for
    end for
end procedure
```

well.

As shown in Fig. 3, the *Client Estimator* calculates $f(\underline{a}_t)$, $f_1(I_{reuse})$, and $f_2(I_{reuse})$ based on the domain knowledge, context knowledge, and measurement vector by using the BF and BB inferences. Eventually, the probability density $f(I_{reuse})$ is computed through the intermediate densities $f(\underline{a}_t)$, $f_1(I_{reuse})$, and $f_2(I_{reuse})$.

Algorithm 1 shows a pseudo-code for the reuse intention recognition. The algorithm begins with the user information and device information, and records the information of all of the processes that are currently being executed in the current device to the event array. For each event e, the PEMS-DBN Client finds the maximum CPU Time through the *Client Estimator* based on the given environment and context information. If the CPU Time of e exceeds the maximum CPU Time and the probability density of the process p is greater than the computed threshold α, then it searches for the absolute path of the used data file and converts the file into a serialized format. Finally, it sends e with user information and device information to the server.

4.4 Proactive Execution

Proactive execution focuses on *when*, boiling down to the problem of "when do users want to reuse the given events?". To observe a sign for the reuse intention, the PEMS-DBN Client periodically performs process monitoring. If the detected process is the same as (or is compatible with) the process of the event, then the PEMS-DBN Client proactively executes it. However, if the process is not the same and there is no compatible application, the PEMS-DBN Client notifies the user via warning popup windows.

Algorithm 2 shows a pseudo-code for proactive execution. The algorithm begins with the received events and data files that were previously used in all registered devices. It then loads the information of all of the installed programs and compatible programs, including their absolute paths, program names, and executable file names. The algorithm searches for all of the processes that are being executed in the current device, and for each event e, it proactively executes the same or compatible applications to reuse the data files. If no compatible applications are available, then the system notifies the user through warning windows.

5. Experiments and Analysis

5.1 Experiment Environment

We collected 1037 events from 6 users who volunteered for the experiment, and the data collection spanned approximately 4 weeks. All participants of the experiment are male with their ages ranged from 20 to 40, and they are working for IT-related companies. Four participants used 2 PCs, and two participants used 2 PCs and 1 notebook for the experiment. In the experiment, because most of the participants listen to music while doing work, the satisfaction and accuracy for the *Player* type were very high, with more than 98% for most of them. Hence, there is not much difference among users' satisfaction, so the analysis of the results for the *Player* type events is omitted.

Each user registered all of his/her devices with the server and downloaded and installed the PEMS-DBN Client on all devices. Also, each user manually recorded all data files that were used and written and marked the reuse value for each file with *true* or *false*, indicating the correct reuse intention for each file that was used by the user. Based on this data, we measured the user satisfaction and system accuracy for each user. Equations (1) and (2) are the satisfaction measure and the accuracy measure, respectively.

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

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

In these equations, $\#Event$ is the number of all events, $\#TrueEvent$ is the number of *True* events, $\#SavedTrueEvent$

Table 2 Satisfaction and accuracy for the *Editor* type events.

(a) User satisfaction (%)

Week \ User	User1	User2	User3	User4	User5	User6
1st	100	100	100	100	87.5	100
2nd	100	100	100	94.1	88.8	100
3rd	100	75	100	93.7	92.8	100
4th	100	100	100	92.8	100	92.8

(b) System accuracy (%)

Week \ User	User1	User2	User3	User4	User5	User6
1st	87.5	52.6	30	63.1	68.4	53.8
2nd	82.3	78.9	36.8	80	50	56.2
3rd	93.7	75	57.8	84.2	66.6	42.8
4th	94.7	92.8	92.8	93.3	92.8	93.7

Table 3 Satisfaction and accuracy for the *Reader* type events.

(a) User satisfaction (%)

Week \ User	User1	User2	User3	User4	User5	User6
1st	33.3	100	100	100	100	100
2nd	88.8	100	100	100	100	100
3rd	100	100	100	100	100	100
4th	100	100	90	81.8	83.3	100

(b) System accuracy (%)

Week \ User	User1	User2	User3	User4	User5	User6
1st (%)	75	36.3	70	90	66.6	70
2nd (%)	57.1	62.5	66.6	91.6	75	66.6
3rd (%)	100	87.5	90	100	88.8	66.6
4th (%)	100	100	81.8	84.6	77.7	84.6

is the number of *Saved* events which are also *True* events, and #*UnsavedFalseEvent* is the number of *Unsaved* events which are also *False* events. Here, *True* events indicate those events that were reused by the user and *False* events indicate the not-reused events. Also, *Saved* events indicate those events that were saved and stored to the server by the system for proactive execution, and *Unsaved* events indicate the ignored and not-saved events.

5.2 Evaluation of User Satisfaction and System Accuracy

Each PEMS-DBN Client gathers satisfaction information to determine the user's reuse intention for every proactive execution. Note that we add the event to the statistics only when normal proactive execution occurs.

The experiment results for 6 users based on diverse context information for each application type are shown in Tables 2 and 3. Tables 2 (a) and (b) show user satisfaction and system accuracy for the *Editor* type events, respectively, and Tables 3 (a) and (b) show the same evaluation results for the *Reader* type events.

Figure 4 shows the average satisfaction and accuracy for all participants. The satisfaction degree is 96.6% for *Editor* and 94.8% for *Reader*, which can be considered very high, while the accuracy degree is 71.7% for *Editor* and 78.7% for *Reader*, which are relatively low. This result of low accuracy for both types of applications was mainly

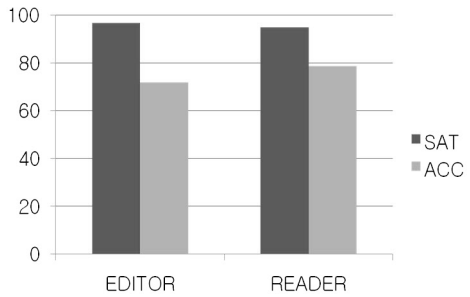

Fig. 4 Average satisfaction and accuracy.

caused by the high ratio of false positive events. In other words, many events that would not be reused in the future are saved to the server due to some miscalculation for the events in the intention recognition phase. One of the main reasons for this miscalculation is the problem of multiple opened files. Some application programs, such as MS Word and MS PowerPoint, maintain only one process for a situation when the application opens several files, so when the PEMS-DBN Client sends an event to the server, a data file that is unrelated to the user intention might be attached accidentally. We are currently working on resolving the multiple opened file issue as a future work by using the *image* of a process which indicates the information similar to an entry displayed on the *Applications* tab of the *Task Manager* (taskmgr.exe) program in MS Windows. For example, when multiple files are opened by MS PowerPoint, the *Processes* tab of *Task Manager* displays only one process but the *Applications* tab displays several entries, each of which corresponds to each opened file. Hence, we might be able to measure the CPU Time of each image of the process so that the system can discriminate the correct data file related the user intention.

Besides the average measures for satisfaction and accuracy, we can analyze how the measures are changing as time goes by from Tables 2 and 3 by considering the *Week* attribute. Note that the satisfaction measures are very high regardless of the progress in weeks, whereas the accuracy measures are mostly increasing as weeks go by. This phenomenon implies that the system is somehow able to accumulate the experience of recognizing the user's reuse intention and apply it to similar situations later on.

5.3 Evaluation of System Efficiency

At this point, we want to show that the PEMS-DBN system operates in an effective and efficient manner while it showed reasonable satisfaction and accuracy measures. The efficiency of the system is mainly evaluated by estimating the resource consumption such as CPU and memory usage during intention recognition and proactive execution processes.

First, we measured how much resources are used during the intention recognition (*IR*) process, and the results are recorded in Tables 4, 5, and 6 for the *Reader* type, the *Editor* type, and the *Player* type events, respectively.

The size of the used data files are classified into 7 cate-

Table 4 Resource consumption during intention recognition (IR) for the *Reader* type.

File Size Type (Kbyte)	Avg. File Size (Kbyte)	Avg. *IR* Time (ms)	Avg. CPU Usage (%)	Avg. Heap Mem Usage (Kbyte)	Avg. Non-Heap Mem Usage (Kbyte)
<10	3.05	8	0	0	2.75
<100	57.61	41.27	0.55	93.29	2.03
<500	267.64	142	0.65	605.2	2.61
<1000	679.86	333.14	0.57	1371.13	9.93
<2000	1432.75	681.81	0.91	2049	24.95
<10000	4121.21	1843.84	1.32	1844.27	14.47
>10000	16817.91	8016	1.5	842.63	138.22

Table 5 Resource consumption during intention recognition (IR) for the *Editor* type.

File Size Type (Kbyte)	Avg. File Size (Kbyte)	Avg. *IR* Time (ms)	Avg. CPU Usage (%)	Avg. Heap Mem Usage (Kbyte)	Avg. Non-Heap Mem Usage (Kbyte)
<10	8.704	31	0	0	5.5
<100	58.17	56	0.556	180.67	2.35
<500	238.25	163.29	0.427	560.82	2.22
<1000	681.79	481.31	0.64	141.62	4.83
<2000	1386.49	869.92	0.75	1962.97	11.6
<10000	3402.08	2700.35	0.78	1627.73	9.64

Table 6 Resource consumption during intention recognition (IR) for the *Player* type.

File Size Type (Kbyte)	Avg. File Size (Kbyte)	Avg. *IR* Time (sec)	Avg. CPU Usage (%)	Avg. Heap Mem Usage (Kbyte)	Avg. Non-Heap Mem Usage (Kbyte)
<5000	3797.92	1.85	2.5	303.92	13.46
<10000	8688.41	3.95	2.7	1771.92	18.1
>10000	10736.51	4.83	3.1	1356.02	11.76

gories from "less than 10 KB" to "greater than 10 MB" (see the *File Size Type* attribute in Tables 4 and 5). However, we used only 3 categories of file size for the *Player* type events (see the *File Size Type* attribute in Table 6) because audio and video files are generally large-sized. Note that, as indicated by *Avg. IR Time* attribute values in the tables, intention recognition is executed in a short period of time (i.e., 1.58 sec for *Reader*, 0.71 sec for *Editor*, and 3.54 sec for *Player*, in average), and the time is increasing proportional to the file size regardless of the event types. Moreover, the amounts of CPU and memory usage during intention recognition are also quite small (i.e. 1.36% CPU usage, about 1 MB heap memory and 27.76 KB non-heap memory usage, in average) regardless of the event types. These observations support that the PEMS-DBN system manages its intention recognition process efficiently.

We also measured similar data of resource usage for the proactive execution process. Tables 7, 8, and 9 show the status of resource consumption during proactive execution (*PE*) for the *Reader* type, the *Editor* type, and the *Player* type events, respectively.

Proactive execution proceeds in two phases; the *monitoring phase* and the *file execution* phase. In the monitoring phase, the system checks whether the files that were used before in other devices by the user will be reused in the current device. In the file execution phase, the system executes the files by launching compatible applications. (In the tables, the *Mon* attribute denotes for monitoring and the *FE* attribute denotes for file execution.)

Note that, as indicated by *Avg. PE Time* attribute values in the tables, proactive execution is also executed in a short period of time (i.e., 0.38 sec for *Reader*, 0.622 sec for *Editor*, and 0.41 sec for *Player*, in average). For the *Reader* type events, proactive execution consumes very small amount of CPU and memory resources, whereas the *Editor* type and the *Player* type events need relatively large amount of resources for proactive execution due to the characteristics of the *Editor* type and *Player* type applications programs which generally occupy large portion of resources during execution.

These experiment data indicate that the PEMS-DBN system consumes very small amount of resources, and also the processes of intention recognition and proactive execution are carried out in a short period of time so that the user is able to accomplish his/her jobs without any delay. In summary, these experimental results support our claim that the PEMS-DBN system is effective and efficient.

6. Conclusions

We have proposed a novel personalized system to provide appropriate services to users by employing an intelligent mechanism using the proactive execution of tasks based on user intention recognition. We exploited a dynamic Bayesian network to probabilistically recognize users' device reuse intention by measuring appropriate CPU times and calculating the probability density for a given event. In addition, we considered various environment and context information for detailed personalization. From a series of experiments, the users' average degree of satisfaction was very high, which was enough to satisfy our expectations. Furthermore, our system achieved this degree of user satisfaction in an effective and efficient manner by minimizing the resource consumption. Although the average accuracy degree is relatively low because of the multiple opened file problem and other reasons, we are convinced that, if the experiment for personalization is continued for a long period of time, the accuracy will increase. We are currently working on resolving the low accuracy issue by solving the multiple opened file problem using the *image* of a process so that the system can discern the file related to the user intention.

Acknowledgments

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, Korea).

Table 7 Resource consumption during proactive execution (PE) for the *Reader* type.

File Size Type (Kbyte)	Avg. File Size (Kbyte)	Avg. *PE* Time		Avg. CPU Usage (%)		Avg. Heap Mem Usage (Kbyte)		Avg. Non-Heap Mem Usage (Kbyte)	
		Mon (sec)	FE (ms)	Mon	FE	Mon	FE	Mon	FE
<10	3.049	0.57	30.5	0	0	298.44	49.32	50.74	0.032
<100	56.68	0.297	11.52	0.569	1.908	489.69	141.11	6.6	0.78
<500	230.64	0.307	12.54	0.121	1.215	400.75	5.16	8.45	0.98
<1000	666.55	0.315	8.92	0.067	0.73	393.13	11.98	7.86	0.93
<2000	1432.75	0.338	9.91	0.058	2.353	272.43	11.21	17.03	2.12
<10000	3568.32	0.298	11.83	0.389	1.769	271	8.22	15.66	1.94
>10000	16817.91	0.547	16	0.69	2.491	257.54	8.21	50.74	2.32

Table 8 Resource consumption during proactive execution (PE) for the *Editor* type.

File Size Type (Kbyte)	Avg. File Size (Kbyte)	Avg. *PE* Time		Avg. CPU Usage (%)		Avg. Heap Mem Usage (Kbyte)		Avg. Non-Heap Mem Usage (Kbyte)	
		Mon (sec)	FE (ms)	Mon	FE	Mon	FE	Mon	FE
<10	8.704	0.75	16	0.592	17.2	317.18	0	94.48	0.064
<100	60.89	0.79	41.18	0.145	3.918	348.24	4.169	14.72	0.056
<500	240.64	0.593	28.5	0.526	2.804	423.59	10.55	10.295	0.77
<1000	720.57	0.472	20.296	0.797	4.193	390.94	8.316	20.84	0.402
<2000	1324.15	0.584	15.25	0.18	2.019	254.97	11.498	26.05	0.68
<10000	3471.35	0.547	17.17	0.606	4.085	529.24	8.071	19.99	1.379

Table 9 Resource consumption during proactive execution (PE) for the *Player* type.

File Size Type (Kbyte)	Avg. File Size (Kbyte)	Avg. *PE* Time		Avg. CPU Usage (%)		Avg. Heap Mem Usage (Kbyte)		Avg. Non-Heap Mem Usage (Kbyte)	
		Mon (sec)	FE (ms)	Mon	FE	Mon	FE	Mon	FE
<5000	3797.9	0.59	7.5	0.11	1.78	249.29	0	50.74	0.03
<10000	8628.6	0.33	12.3	0.14	4.76	264.7	15.46	18.54	0.11
>10000	10955.5	0.33	12.5	0.13	5.11	261.7	20.27	22.64	0.07

References

[1] D. Aarno, "Intention recognition in human machine collaborative systems," Ph.D. dissertation, KTH School of Computer Science and Communication, 2007.

[2] J. Budzik, K. Hammond, C. Marlow, and A. Scheinkman, "Anticipating information needs: Everyday applications as interfaces to Internet information servers," Proc. World Conf. of the WWW, Internet and Intranet (WebNet 98), Orlando, USA, Nov. 1998.

[3] J. Budzik and K. Hammond, "User interactions with everyday applications as context for just-in-time information access," Proc. Intl. Conf. on Intelligent User Interfaces, pp.44–51, New Orleans, USA, Jan. 2000.

[4] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse, "The Lumiere project: Bayesian user modelling for inferring the goals and needs of software users," Proc. 14th Conf. on Uncertainty in Artificial Intelligence, pp.256–265, Madison, USA, July 1998.

[5] H. Jeon, T. Kim, and J. Choi, "Proactive task execution using data sharing and event transition among personal devices," submitted to KSII Trans. on Internet and Information Systems, 2010.

[6] P. Krauthausen and U. Hanebeck, "Intention recognition for partial-order plans using dynamic Bayesian networks," Proc. 12th Intl. Conf. on Information Fusion, pp.444–451, Seattle, USA, July 2009.

[7] L. Pereira and H. Anh, "Intention recognition via causal Bayes networks plus plan generation," Proc. 14th Portuguese Intl. Conf. on Artificial Intelligence, Lecture Notes in Artificial Intelligence, vol.5816, pp.138–149, 2009.

[8] L. Pereira and H. Anh, "Elder care via intention recognition and evolution prospection," Proc. 18th Intl. Conf. on Applications of Declarative Programming and Knowledge Management (INAP 09), Evora, Portugal, Nov. 2009.

[9] B. Rhodes and P. Maes, "Just-in-time information retrieval agents," IBM Syst. J., vol.39, no.3–4, pp.685–704, July 2000.

[10] A. Salovaara and A. Oulasvirta, "Six modes of proactive resource management: A user-centric typology for proactive behaviors," Proc. 3rd Nordic Conf. on Human-Computer Interaction, pp.57–60, Tampere, Finland, Oct. 2004.

[11] O. Schrempf and U. Hanebeck, "A generic model for estimating user-intentions in human-robot cooperation," Proc. 2nd Intl. Conf. on Informatics in Control, Automation and Robotics (ICINCO 05), pp.251–256, Barcelona, Spain, Sept. 2005.

[12] O. Schrempf, U. Hanebeck, A. Schmid, and H. Worn, "A novel approach to proactive human-robot cooperation," Proc. IEEE Intl. Workshop on Robot and Human Interactive Comm., pp.555–560, Nashville, USA, Aug. 2005.

[13] O. Schrempf, D. Albrecht, and U. Hanebeck, "Tractable probabilistic models for intention recognition based on expert knowledge," Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS 07), pp.1429–1434, San Diego, USA, Oct. 2007.

[14] P. Shenoy and R. Rao, "Dynamic Bayesian networks for brain-computer interfaces," Advances in Neural Information Processing

Systems (NIPS 2005), vol.17, pp.1265–1272, 2005.
[15] N. Stefanov, A. Peer, and M. Buss, "Online intention recognition in computer-assisted teleoperation systems," Proc. 7th Intl. Conf. on Haptics: Generating and Perceiving Tangible Sensations, pp.233–239, Amsterdam, The Netherlands, July 2010.
[16] K. Tahboub, "Intelligent human-machine interaction based on dynamic Bayesian networks probabilistic intention recognition," J. Intelligent Robotics Systems, vol.45, no.1, pp.31–52, Jan. 2006.
[17] R. Want, T. Pering, and D. Tennenhouse, "Comparing autonomic and proactive computing," IBM Syst. J., vol.42, no.1, pp.129–135, Jan. 2003.

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 agents, 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 and ontology.

Joongmin Choi is a professor in the Department of Computer Science and Engineering, Hanyang University, Korea. He is the Director of the Web Intelligent Consortium (WIC) Korea Center. He received his B.S. and M.S. degrees in Computer Engineering from Seoul National University, Korea, in 1984 and 1986, respectively, and 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.