Personal Workflow Management in Support of Pervasive Computing

San-Yih Hwang, Jeng-Kuen Chiu, Wan-Shiou Yang

Department of Information Management
National Sun Yat-Sen University, Taiwan
{syhwang,jack,ryang}@mis.nsysu.edu.tw

1. Introduction

With the advent of new wireless technologies, mobile personal computers (MPCs) are quickly gaining their popularity. Concurrently an increasingly number of services for MPCs have been developed. However, current services view PMCs as a vehicle to perform individual activities. No attempt has been made to provide *process services* that reach mobile users via their PMCs. In the real world, many personal applications are process-oriented, e.g., planning a trip, holding a party, applying a new job, etc. We expect PMCs to help manage and execute these processes in the future, serving the role of planning, coordinating, and reminding task executions so as to accomplish predefined personal goals. On the other hand, traditional workflow management systems are mainly used to coordinate business processes in enterprises. These processes must be repetitive and have well-formed structures. Personal processes differ from their business counterparts in that they are subject to change and rigid regulations on control flow are seldom imposed. As a result, personal activities are related mainly due to their data dependencies and their coordination must be flexible. These differences demand a new personal workflow model as well as the innovative design of a personal workflow system (PWFS).

2. The Process Model

We model a process as a set of tasks, each of which is associated with several attributes such as name, input data, output data, the durations and places in which this task can be executed. We can view a PWFS as a function that, when time and place allow, maps a set of input data and a set of tasks into a set of output data. Formally, this function can be specified as

$$2^{D_{data}} \times 2^{D_{task}} \rightarrow 2^{D_{data}} \cup \{\perp\}, \text{where} \perp \text{indicate invalid execution}$$

Viewing a PWF as a function described above allows us to place the following inquiries:

1. Given a set of input data items and a set of tasks to be executed, what data items can be generated?
2. Given a set of input data, what tasks should be executed in order to produce a specific set of output data?
3. To execute a specific set of tasks, what input data items are needed?

Therefore, we propose the following three operations that intend to answer the above three queries:

ProduceData: inputset × taskset → outputset
InvolvedTask: inputset × outputset → taskset
RequiredData: taskset × outputset → inputset

With the above three operations, many queries about the execution of a personal process can be specified. For example, suppose the user would like to identify the set of tasks that can be co-executed with C after both A and B complete, the following SQL statement is specified:

```
SELECT   t.name
FROM     TASK  t, c
WHERE    t  IN  InvolvedTask(ProduceData(avail,('A','B')),ANY)
AND      c.name = 'C'
AND      t.time  OVERLAPS  c.time
AND      t.place  OVERLAPS  c.place;
```

3. Implementation Issues

We propose to adopt metagraphs [1], a formal model designed to express data dependencies of processes in model bases community, to specify personal processes. The analytical capability of metagraphs paves the way for efficiently implementing the proposed operations. Besides, implementing a personal workflow system involves many issues. To name a few: the storage of personal processes, the design of graphical interface, the presentation of worklist, the development of triggering system, the design of temporal and spatial operations, and so forth. These issues will be investigated when we come to stage of implementation.

A unique feature about business processes is its flexibility. It is quite often that mobile users execute tasks that produce unexpected results or even engage in some totally unplanned tasks. In this case, rather than rejecting this change, the PWFS can do nothing but to accept the consequence. In return, the PWFS examines the impact of this change and adjusts the unexecuted part of the task definitions according to user's decision. For example, the execution of a task may generate fewer data items than expected. As a result, some subsequent task may become unexecutable due to the lack of required input data. However, while the entire task may not be executed, part of the task may still be executable, resulting in the splitting of a task. On the other hand, executing a task may sometimes produce extra data items. These extra data items may in turn enable the early execution of some tasks or make others obsolete. Operations for measuring the effect of unexpected task execution have been proposed but are not presented here due to space limitations.

References

[1] A. Basu and R.W. Blanning, "A Formal Approach to Workflow Analysis," *Information Systems Research*, 11(1), 2000, pp.17-36.