Personal Workflows: Modeling and Management*

San-Yih Hwang and Ya-Fan Chen

Department of Information Management
National Sun Yat-sen University
Kaohsiung, Taiwan 80424
syhwang@mis.nsysu.edu.tw

Abstract. As mobile devices are getting powerful, reliable, and inexpensive, more and more personal services have been introduced to individuals via their mobile devices. These services enable users to perform a broad range of activities at any time and anywhere. While a lot of research and development effort has been focusing on extending the scope of activities that can be carried out on mobile devices, little attempt has been made to provide process-oriented services. In this paper, we propose a model for specifying and querying personal processes. This model is equipped with a set of operations that allow mobile users to express queries on their personal processes. Answers of these queries will help mobile users in planning their activities while they are on the move. We have designed and implemented a prototyped personal workflow system on Palm Pilot PDA based on the proposed model. Experiences with this implementation are provided.

1 Introduction

As microcomputers and wireless technologies are getting their maturity, personal mobile computers (PMCs) such as PDAs and mobile phones are becoming popular. The services provided to the PMCs reduce users' space and time constraints. There have been many applications developed for PMCs. As an example, there are more than 100,000 applications available solely for Palm Pilot PDAs [10]. Typical applications include calendar, memo, address book, and to do list. However, while these applications allow users to record and retrieve information about tasks and data, the relationships between tasks and data are left out. In fact, many of people's daily activities are not independent, and they are likely to be process-oriented.

Traditional workflow management systems (WFMSs) are designed to coordinate business processes in enterprises. These processes must be repetitive and have well-formed structures. In this paper, we focus on the specification and execution of processes that constitute personal tasks and data. We call these processes *personal processes* and the system that coordinates personal processes *personal workflow*

* This work is supported in part by the National Science Council in Taiwan under grant number NSC91-2219-E-110-003.

management system (PWFMS). The following example illustrates the merit a PWFMS may bring about.

An example

A college student Rita specifies two personal processes in her PWFMS: the class registration process and a party holding process. The class registration process follows the regulation of the college in registering classes that she plans to take in the coming semester and paying the required fee. It is made up of several tasks such as requesting registration forms, filling in forms, getting signatures from advisor and department head, paying course fee, etc. The party holding process is designed to prepare a party to be held in the near future and is composed of tasks like planning, finding the place, shopping, sending invitation letters, and so on. Once the two processes are successfully specified, her PWFMS is expected to provide a number of personal services such as the following:

1. When Rita walks close to a department office, her PWFMS reminds her of getting signatures from the department head, provided all data required to accomplish this task is ready.
2. When Rita decides on a number of tasks to perform, her PWFMS informs her to carry the data items required for these tasks.
3. When Rita determines to pay course fee at a bank, her PWFMS advises her that sending invitation letters (for the party holding process) can also be done nearby.
4. Suppose Rita plans to pay course fee today. Paying course fee requires an "invoice" from the registrar office. Her PWFMS will instruct her that a number of tasks need to be done first in order to receive the invoice.

To provide such personal services, we need to define a personal process model that allows users to specify individual tasks as well as their interdependencies. This model also provides a set of primitive operations. By properly combining several operations, users can inquire the current status of an ongoing process, determine the set of tasks that can be performed in the same trip, plan the future work for accomplishing a particular goal, and so forth.

However, traditional business process models and commercial WFMSs are not suitable in the context of personal process management due to the following reasons:

1. Each personal process instance has a unique structure. In other words, after a personal process is specified, most likely only one instance will be executed.
2. Personal tasks are primarily related by their executable time, executable places, and input and output data. Unlike enterprises that impose many regulations and procedures that need to be followed strictly by their processes, human beings seldom impose rigid rules on their personal tasks. As a result, the specification of control flow constructs such as or-split, and-split, or-join, and and-join, is rarely needed.
3. The main objective of a PWFMS is to remind or provide suggestions to a mobile user, rather than to enforce task executions as does a commercial WFMS. Therefore, traditional workflow scheduling issues, which address

how to determine the mapping between tasks and available resources, do not seem to exist in this context.

4. The coordination between steps of each process has to be flexible. It is quite often that a mobile user executes a task that produces unexpected results or even engages in a totally unexpected task. In this case, rather than rejecting this change, the personal workflow system examines the impact of this change and provides suggestions.

These unique features call for a novel design of personal process model and PWFMS.

In this paper, we focus on the model for specifying and querying personal processes. This model is equipped with a set of operations, which enable mobile users to place queries about the execution status of a personal process. We have also constructed a PWFMS prototype that implements three components: storage manager, query processor, and process recommendation system. Experiences of this prototyping effort are also provided.

Related work

Personal information services and applications (PISA) were revealed in [4] as a challenging area of future wireless computing. However, the issues examined mainly concern transactional services and cache consistency. In [7], an architecture called *Rome* was proposed to manage triggers at a centralized infrastructure. Mobile users will be alerted to do something when the trigger conditions have been satisfied. The main contribution of the architecture is an infrastructure- centric approach to the trigger management problem. In summary, these work intend to provide more services to personal activities rather than personal processes.

Issues for integrating mobile computing and workflow management technologies were discussed in [8]. The focus was how to conduct efficient resource management and provide convenient electronic document browsing to incorporate field workers into workflow management. A prototyped workflow system called WHAM that supports mobile applications was described in [9]. A novel workflow management model based on mobile agents located in wired or wireless networks was proposed by [6]. Although these work all addressed processes in a mobile environment, the goal was on extending workflow technologies to support mobile workforces for *business processes*, rather than handling activities for *personal processes*.

In [1], Abeta and Kakizaki designed a PDA system that accumulated and extracted operation records of workers working on the same project. Temporal relationships between work events were identified and served as basis for making recommendation for event browsing. The goal was not to help schedule personal processes.

Paper organization

This paper is organized as follows. Section 2 is devoted to the description of the proposed personal process model. Section 3 presents a set of primitive operations for formulating the queries. We have built a prototyped system on Palm Pilot PDA that implements the proposed model and several components proposed in the architecture.

This prototype provides a high level interface that allows the specification and querying of personal processes. Section 4 describes this prototype and the lessons we learned from this prototyping effort. Finally, in Section 5, we conclude this paper by reviewing what we have done and pointing out our future work.

2 Personal process model

In this section, we define the syntax of personal processes. A personal process is comprised of the following components:

- a set T of tasks,
- a set D of data,
- several functions that map a task to its name (Φ_n), the input data set(Φ_i), the output data set(Φ_o), the executable time intervals(Φ_i), the executable places(Φ_p):
  - Φ_n: $T \rightarrow$String
  - Φ_i: $T \rightarrow 2^D$
  - Φ_o: $T \rightarrow 2^D$
  - Φ_t: $T \rightarrow 2^{Time \times Time}$, where *Time* is the set of time.
  - Φ_p: $T \rightarrow 2^{Point \times Point}$, where *Point* is the set of geographical points, each of which is expressed as (latitude, longitude),
- a function Δ_n:$D \rightarrow$String that maps a data to its associated name.

In addition, there are attributes that record the execution status of tasks and data. We call these attributes *control* attributes. In this paper, we consider two control attributes, Φ_s and Δ_s, that are associated with tasks and data respectively. Φ_s: $T \rightarrow$ (UNEXECUTED, EXECUTING, COMPLETED) reveals the status of a task, which could be *unexecuted*, *executing*, or *completed*. Δ_s: $D \rightarrow$ (UNAVAILABLE, AVAILABLE) describes the availability of a data item.

The four functions Φ_i, Φ_o, Φ_t, Φ_p, are attributes pertaining to tasks. Φ_t and Φ_p are *time* and *place* attributes that specify respectively when and where the pertaining task can be performed. Φ_i and Φ_o are input and output attributes that specify the sets of data items that this task takes as input and output respectively. For representation purpose, we adopt meta graph for visualizing data dependencies. A metagarph is a graph-theoretic construct that captures relationships between pairs of sets of elements [2]. In its pictorial representation, a set of elements is surrounded by a small cycle, and the edges are arrows connecting the cycles. That is, an edge represents the direction of the input-to-output relationship between two element sets produced by a task. As an example, Figure 1 shows the metagraph of the party holding process described in Section 1.

We distinguish data in the input and output attributes into two kinds: *primitive* and *processed*. A primitive data is not produced by any task modeled in the system, and a processed data must be generated by at least one other task. A primitive data could be a data file, a blank form, a personal belonging (e.g., the ID card, credit card), or anything that is physically available somewhere. A processed data is available only when at least one task that is capable of producing it is completed. For example, the

task *sending invitation letters* takes two data items as input: 'Invitation Letters' and 'Credit Card'. The former is a processed data item, which can only be generated by *writing invitation letters*, while the latter is a primitive data item.

Note that in the proposed model, there is no rigid order on task executions. Tasks are associated by their respective attribute values, which may implicitly decide their execution orders. For example, if a task T_2 needs a data item that can only be produced by T_1, T_2 will not execute before T_1 terminates.

Similar to the other process models, a personal process model also has its constraints. Here we define two types of constaints: reachability and liveness. To address both constraints in the context of personal processes, we define the starting data set S=$\{d \in D$: d is a primitive data item$\}$ and a target data set $G \subseteq D$ for a process model, where the availability of each data item in G marks the successful termination of the personal process. Formally, reachability and liveness of a personal process $P(T, D)$ are defined as follows:

Definition 1: A personal process $P(T, D)$ is reachable if for every task $t \in T$, there exist a metapath that connects S and $\Phi_i(t)$ and a metapath that connects $\Phi_o(t)$ to at least one data item in G.

Definition 2: A personal process $P(T, D)$ is live if there exists a metapath that connects S to D.

If a personal process is not live, its execution will not achieve the goal of the process. If a personal process is not reachable, at least one of its tasks becomes redundant. In either case, the entire process definition is considered incorrect and needs modification. Details for checking the existence of a metapath can be found in [2], which also shows several other types of workflow analysis.

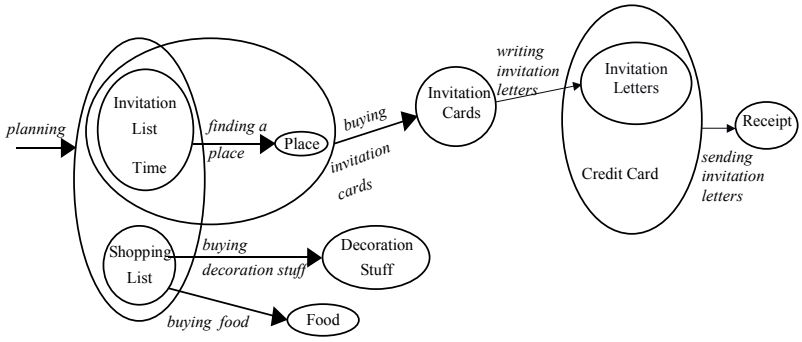

Figure 1: The meta graph of a party holding process

3 Algebraic operations

Since a personal process is modeled as a combination of task set and data set, traditional set operations like *intersection* ($\cap$), *union* ($\cup$), and *difference* ($-$) are applicable. Besides, as tasks and data are associated with a set of predefined attributes (functions), relational operator *selection* (σ) can also be applied. The combinations of these relational operations are powerful enough to answer many types of queries. However, the relational expressions are often lengthy and inefficient in handling queries frequently raised on personal processes, such as those described in the example of Section 1. We therefore propose several operations that enable easier specification and efficient execution of users' queries. Based on the types of data they needs, they can be classified into four categories: binary operations with operands $T \times T$, binary operations with operands $D \times D$, binary operations with operands $D \times T$, and unary operations with operand T.

- $T \times T \rightarrow T$: These are binary operations that take operands of type T. These operations include UNION, INTERSECTION, DIFFERENCE, TIME_OVERLAP, and PLACE_OVERLAP. UNION, INTERSECTION and DIFFERENCE operations are basic set operations. TIME_OVERLAP (PLACE_OVERLAP) is used for retrieving a subset of tasks in the first operand whose execution times (places) overlap with some task in the second operand.
- $D \times D \rightarrow D$ (T): UNION, INTERSECTION, and DIFFERENCE operations are also applicable to operands of type D. In addition, we propose a new operation NEED_TASK that returns a set of tasks that takes data items in the first operand as the input and produces the data set in the second operand.
- $D \times T \rightarrow T$:One operation MAKE_EXECUTABLE is proposed. It identifies a subset of executable tasks in the second operand while given the data items in the first operand as the input.
- $T \rightarrow D$: This category contains two operations, namely COMBINED_INPUT and COMBINED_OUTPUT, that return the aggregate input and output respectively for executing the tasks in the first operand.

More formally, the new operations are defined below:

DEFINITION (PLACE_OVERLAP, $\circ_p$): Given two sets S_1 and S_2 of tasks, PLACE_OVERLAP, denoted $\circ_p$, produces a subset of S_1 whose executing place overlaps with the executing place of some task in S_2. Formally,

$$S_1 \circ_p S_2 = \{t1 : t1 \in S_1, (\exists\, t2 \in S_2, \exists p1 \in t1.p, \exists p2 \in t2.p, RECTANGLE\_OVERLAP(p1, p2))\}$$

DEFINITION (TIME_OVERLAP, $\circ_t$): Given two sets S_1 and S_2 of tasks, TIME_OVERLAP, denoted $\circ_t$, produces a subset of S_1 whose executing time overlaps with the executing time intervals of some task in S_2. Formally,

$$S_1 \circ_t S_2 = \{t1 : t1 \in S_1, (\exists\, t2 \in S_2, \exists i1 \in t1.t, \exists i2 \in t2.t, INTERVAL\_OVERLAP(i1, i2))\}$$

DEFINITION (MAKE_EXECUTABLE, $\underset{dt}{\rightarrow}$): Given a set T of tasks and a set D of data, MAKE_EXECUTABLE, denoted $\underset{dt}{\rightarrow}$, returns a subset of T, each of which has the input data as a subset of D. Formally, $D \underset{dt}{\rightarrow} T = \{t : t \in T, t.i \subseteq D\}$.

DEFINITION (NEED_TASK, $\underset{dd}{\rightarrow}$): Given two data sets D_1 and D_2, NEED_TASK operation, denoted $\underset{dd}{\rightarrow}$, produces a set T of tasks that can be collectively executed by taking D_1 as the input and producing D_2 and has the lowest cost. Let $MinMetaPath(D_1, D_2)$ be a function that returns the minimum cost of all possible task sets that connect D_1 and D_2. $MinMetaPath(D_1, D_2)$ can be defined recursively as follows:

$MinMetaPath(D_1, D_2) = 0$ if $D_2 \subseteq D_1$

$MinMetaPath(D_1, D_2) = \underset{Output(T')=D_2}{Minimum} (MinMetaPath(D_1, Input(T')) + Cost(T'))$,

where $Input(T')$ and $Output(T')$ denote the aggregate input data set and the aggregate output data set of a task set T', respectively. Further $Cost(T') = \sum_{t \in T'} Priority(t)$, where $Priority(t)$ is the priority of task t assigned by the mobile user.

DEFINITION (COMBINED_INPUT, $\uparrow_i$): Given a set T of tasks, COMBINED_INPUT operation, denoted $\uparrow_i$, returns a set of data, each of which is an element of input data of some element in T but not in the output data of any element in T. Formally, $\uparrow_i (T) \equiv \{\cup t.i : t \in T\} - \{\cup t.o : t \in T\}$.

DEFINITION (COMBINED_OUTPUT, $\uparrow_o$): Given a set T of tasks, COMBINED_OUTPUT operation, denoted $\uparrow_o$, returns a set of data, each of which is an element of output data of some task in T. Formally, $\uparrow_o (T) \equiv \{\cup t.o : t \in T\}$.

The meaning and notations of set operations ($\cup$, $\cap$, and $-$) and select operation (σ) are defined the same as in the relational model [10] and omitted here for brevity. In the following, we illustrate the power of the proposed operations by showing several query examples:

1. Find the set of tasks that need to be done in order to produce 'receipt'.

   $\sigma_{s=available}(D)$ NEED_TASK $\sigma_{n='receipt'}(D)$

2. Find a set of tasks that can be co-executed with 'buying invitation cards', when 'planning' and 'finding a place' are both completed.

   $(((\sigma_{s=available} D) \cup (\uparrow_o (\sigma_{n='planning'} T) \cup (\sigma_{n='finding\ a\ place'} T)))$

   $\underset{dt}{\rightarrow}((\sigma_{s=unexecuted} T) - (\sigma_{n='planning'} T) - (\sigma_{n='finding\ a\ place'} T))$

   $\circ_p \sigma_{n='buying\ \ invitation\ \ cards'} T$

   $\circ_t \sigma_{n='buying\ \ invitation\ \ cards'} T$

3. Find the set of data that is needed to complete tasks 'finding a place', 'buying invitation cards', and 'buying decoration stuff'.

   $\uparrow_i (\sigma_{n='finding\ a\ place'}(T) \cup \sigma_{n='buying\ invitation\ cards'}(T) \cup \sigma_{n='buying\ decoration\ stuff'}(T))$

4. Retrieve the set of tasks with a given data item ('receipt') as part of its input.

$$\sigma_{i=\text{'receipt'}}(T)$$

5. Retrieve the set of tasks whose executions results in the generation of a specified data item ('receipt').

$$\sigma_{o=\text{'receipt'}}(T)$$

6. Find the set of tasks that can be executed immediately (Note that CURRENT is a system-defined dummy task that has the current time and the current place as the attribute values).

$$\sigma_{s=available}(D) \underset{dt}{\rightarrow} \sigma_{s=unexecuted}(T) \circ_t \text{CURRENT} \circ_p \text{CURRENT}$$

Note that there may not be a unique way for expressing a given users' inquiry. Two query expressions are said to be equivalent if they are destined to generate the same result based on any kind of data set. Query optimization aims to identify an expression with the least cost among all equivalent expression. In order to find such an optimal query expression, expression rules that generate equivalent expressions have to be consulted. An expression rule specifies how to transform an expression into a logically equivalent one. The set of equivalence rules are not enumerated here due to space limitation. Interesting readers are referred to [3] for a detailed coverage of the query optimization issues.

4 Implementation

To prove our concept, we have implemented a PWFMS prototype that includes three components, namely the storage manager and query processor at the client, and the process recommendation system at the server. The client program was implemented on a Palm Pilot by using J2ME-CLDC (Java 2 Micro Edition, Connected, Limited Device Configuration) as the develop tool. The process recommendation system at the server is a web-based system that was implemented by using PHP and Oracle 8 as the database.

Process definition

Personal processes can be either explicitly specified by mobile users or downloaded from the process recommendation system located on the server. To explicitly specify a personal process, a mobile user must supply information required for the process, including the tasks, the input/output data, and the mapping functions. Figure A-1 and A-2 in the Appendix display screenshots for defining a task and entering input data set respectively.

Specifying a personal process precisely could be a difficult job, especially when this process has to comply with regulations of some organizations. To deal with these difficulties, we have developed a personalized process recommendation system that provides personalized workflow templates in a particular domain. Every organization has its intended set of customers and must have some processes that interact with its customers. Each such a process can be visualized as (part of) a personal process when a customer needs to interact with the organization in order to achieve a particular goal.

Therefore, we advocate that an organization should organize these processes and provide them as personalized processes in a particular format. The provision of personalized processes can be seen as a personalization endeavor of an organization in that different customers with different background and/or interests may receive different personal processes even for accomplishing the same job. After downloading a personalized process, a mobile user will be reminded of things that need to be done at the right time and the right place with the help of his/her PWFMS.

We call each distinct process definition a *workflow template*. The designer of a personal process has to elaborate all workflow templates and, for each workflow template, indicate the associate attribute values. For example, suppose the process designer has identified that there are three distinct workflow templates T1, T2, and T3. The designer further discovers that only {A1, A2, A3} are relevant to template T1, {A2, A3, A5} are relevant to template T2, and {A2, A4} are relevant to template T3. These specifications are shown in Table 1.

Table 3: A possible template specification

Workflow template	A1	A2	A3	A4	A5
T1	a_{11}	a_{21}	a_{31}	any	any
T2	any	a_{21}	a_{32}	any	a_{31}
T3	any	a_{22}	any	a_{41}	any

A notable observation from Table 1 is that the conditions that are associated to any pair of workflow templates must be exclusive. Otherwise, it is possible that a mobile person with specific background may be eligible to two different templates, which will be confusing.

These workflow template specifications will be organized in a decision tree from which questions about users' interests or background can be posed. Formally, suppose there are n templates: T_1, T_2, …, T_n. Each template T_i, $0 \le i \le n$, has an access probability p_i and a condition specification on a subset of m attributes: A_1, A_2, …, A_m. The objective is to construct a decision tree such that

1. the path that leads to each template T_i, $0 \le i \le n$, in the decision tree must satisfy the condition corresponding to T_i, and
2. the function $\sum_{i=1}^{n} l_i \cdot p_i$ is minimized, where l_i is the length of the path that leads to a template T_i.

This problem could be NP-hard. Here we adopt a greedy heuristic that favors the attributes that are involved in the specification of more popular templates. More precisely, for each attribute A_j, we compute the aggregate probability P_j of the templates that involve A_j as follows: $P_j = \sum_{\text{the condition of } Ti \text{ involves } Aj} p_i$. The attribute with the highest aggregate probability is chosen as the first distinguishing attribute in the decision tree. Depending on the values of this attribute, workflow templates are clustered into a number of (non-exclusive) groups. The same method is recursively applied to each group to construct the decision tree. This procedure terminates when a

group contains only one template and the condition of the template has been fully specified along the path to the root.

Figure A-3 shows two screenshots for downloading the student registration process of our university, which involves eight attributes (including gender, student status, and loan requirement).

Query processing

To ease the query formulation, we have designed a language construct SELECT-FROM-GIVEN that is similar to the basic SELECT-FROM-WHERE blocks of SQL. The operation to be performed and its output attributes are specified in 'SELECT clause. The process(es) to which this operation is applied to is declared in 'FROM clause. The 'GIVEN clause describes the operands as well as the constraints on data, tasks, time, and place. However, it is still unlikely that an ordinary user will specify a query conforming to the proposed query syntax. We therefore predefined some frequently used queries that require a mobile user only to specify parameters. The screenshots for dynamically changing the status of a task (e.g., from un-executed to completed) and the invocation of pre-defined queries are shown in Figure A-4 and A-5 respectively.

5 Conclusions

We have introduced a personal process model and operations for expressing user's inquiries on their personal processes. A PWFMS prototype on Palm Pilot and a personalized process recommendation system on the server located on the fixed network have been implemented.

As web services are becoming popular, more and more business processes or tasks will be available on the Internet in the form of web services. We are in the course of extending the PWFMS framework to incorporate web services from various enterprises. The idea is that a mobile user can invoke the web service of a task in a personal process and automatically receive notification upon completion of the task. In our opinion, the PWFMS is an excellent platform for arranging related web services to achieve mobile users' personal goals. The identification of suitable business workflows and the synchronization between a personal workflow and business workflows within respective organizations will be explored.

References

1. A. Abeta, K. Kakizaki, "Implementation and evaluation of an automatic personal workflow extraction method," *Proc. of Int'l. Conf. on Computer Software and Application Conference* (*COMPSAC*), pp.206 –212, 1999.
2. A. Basu and R. W. Blanning, "A Formal Approach to Workflow Analysis," *Information Systems Research*, 11(1), 2000, pp.17-36.

3. Ya-Fang Chen, “The Research on Personal Workflow Systems in Support of Pervasive Computing,” *Master thesis,* National Sun Yat-sen University, July 2001.
4. A. Elmagarmid, J. Jing, and T. Furukawa, “Wireless Client/Server Computing for Personal Information Services and Applications,” *ACM SIGMOD RECORD*, 24(4), 1995, pp.16-21.
5. R. Elmasri, S. B. Navathe, *Fundamentals of Database Systems*, Ch. 7, Addison Wesley, Massachusetts, 2000.
6. W. Gang, W. Quanyuan, W. Huaimin, “A novel workflow management model based on mobile agents for internet electronic commerce,” *Proc. 36th Int'l. Conf. on Technology of Object-Oriented Languages and System* (*TOOLS*), pp.182–187, 2000.
7. A. Huang, B. Ling, S. Ponnekanti, “Pervasive Computing: What is it good for?”, *Proc. of ACM Int'l Workshop on Data Engineering for Wireless and Mobile Access*, pp.84-91, 1999.
8. J. Jing, K. Huff, H. Sinha, B. Hurwitz, B. Robinson, “Workflow and Application Adaptations in Mobile Environments," *Proc. of 2nd IEEE Workshop on Mobile Computing Systems and Applications,* pp.62-69, 1999.
9. J. Jing, K. Huff et al, "WHAM: Supporting Mobile Workforce and Applications in Workflow Environment", *Proc. of 10th Int'l .Workshop on Research Issues in Data Engineering*, pp. 31-38, 2000.
10. http://www.palmblvd.com.

Appendix: Screenshots of the PWFMS prototype

Figure A-1: Task definition

Figure A-2: Input data definition

Figure A-3: Downloading a personal process from the process recommendation system

Figure A-4: Changing the status of tasks

Figure A-5: Showing the pre-defined queries