Chinese Journal of Electronics
Vol.21, No.3, July 2012

Context-Aware Task Allocation for Quick Collaborative Responses*

SUN Yuqing[1], Matthias Farwick[2], Patrick C.K. Hung[3], Dickson K.W. Chiu[4] and JI Guangjun[1]

(1.*School of Computer Science and Technology, Shandong University, Jinan 250100, China*)

(2.*Institute of Computer Science, University of Innsbruck, Austria*)

(3.*Faculty of Business and Information Technology, University of Ontario Institute of Technology, Canada*)

(4.*Dickson Computer Systems, Hong Kong*)

Abstract — Under some emergencies, persons are required to arrive quickly at the scene and collaborate on sensitive tasks. To ensure effective performance of these tasks and be compliant with business regulations, user context should be considered and security requirements are desired. In this paper, we tackle this challenging problem of quick collaborative response from the view of task allocation with user authorization. The considered Quick-response task allocation problem (QTAP) answers how to find a user-task allocation solution, the group of qualified users who can fastest arrive at the scene to fulfill the collaborative processes and satisfy the required security constraints. We study the computational complexity of this problem and solve it by the reduction to the well-studied scheduling problem. We further discuss an important extension of QTAP that supports task dependencies and propose an algorithm to solve it.

Key words — Collaboration, Task allocation, Context awareness, Authorization.

I. Introduction and Motivation

Under some emergencies, persons are required to quickly arrive at the scene and collaborate on sensitive processes, which are unknown in advance and may change at any moment. For example, in the aviation industry, highly complex and delicate processes have to be executed on a routine basis in order to guarantee proper maintenance of aeroplanes before a flight. A maintenance technician first checks the important functionalities of the planes, often requiring a second independent check by another technician. Also the technicians need to be agile while inspecting, such as moving in and around the plane. This makes the use of handheld devices appealing, which allows the technicians to perform certain tasks of the process at specific locations, such as at a safe position or the only position where they can effectively perform the required tasks. When exceptional vulnerabilities of parts have been detected, the maintenance process needs to be quickly adapted without violating security constraints of the original inspection process. In some cases, task performers need to be warned or called to the scene for assistance. During these processes, quick collaborative response is required since any delay would cause huge loss, possibly in the range of one to several hundred thousand dollars a day for an airline company. Other examples of emergent situations include nature disasters, terrorist attack, and healthcare services, in which the requirements of quick collaborative response are quite similar.

In such emergency applications, the required tasks are unknown or unpredictable in advance. To effectively schedule the task performers for quick collaborative response, the access control system of the process engine should have the collaborative capabilities to select appropriate users and support corresponding authorizations concerning necessary security and business factors. Firstly, security constraints are desired on authorizations. Since multiple persons may be involved in an urgent collaboration playing different roles for sensitive tasks, security constraints require enforcement on their authorizations. For example, the widely adopted Separation of duty (SoD) constraint requires a user not being authorized to more than a certain number of tasks to a sensitive process in order to prevent fraud[1]. Secondly, performers need to possess certain qualifications in order to ensure effective performance or regulation compliance. For instance, the routine double check of an airplane should be accomplished by two engineers. Thirdly, both user and system contexts should be considered. When persons are involved in emergency processes, the elapsed time for finishing a task includes the time for a performer to attend the location plus the time for executing the task. If there is more than one candidate for a task, it is desirable to select the most suitable candidate from the view of quick collaborative response.

In this paper, we tackle the quick collaborative response problem from the view of task allocation by user authorizations. Based on our previously proposed Context-aware authorization model (CAM)[2], we study the Quick-response task allocation problem (QTAP) to return a solution of user-task assignments such that the selected qualified users can fastest arrive at the scene and fulfill the urgent collaborative tasks, while satisfying security constraints. We analyze the computational complexity of this problem and show that it is NP-hard in general. A solution to QTAP is presented by the reduction to the well studied scheduling problem under some construction tricks so as to benefit from the previous research results. We also discuss a useful extension of QTAP considering task dependencies and provide an algorithm to solve it.

*Manuscript Received Apr. 2011; Accepted Oct. 2011. This work is supported partly by the National Natural Science Foundation of China (No.61173140), the Science Foundation of Shandong Province (No.Y2008G28) and the Independent Innovation Foundation of Shandong University (No.2010JC010), The Opeh Foundation of the Satate Key Lablratory of Computer Arehiteeture, Chinese Academy of Sciences.

The paper is organized as follows. Section II compares the related works. After reviewing the Context-aware authorization model in Section III, we discuss the QTAP problem and present a solution in Section IV. Next, we discuss the QTAP extension considering task dependencies, together with a heuristic algorithm and experiments in Section V. Finally, we discuss the merits and some implementation details of our solution in Section VI and conclude the paper in Section VII.

II. Related Work

The existing literature provides a variety of techniques addressing secure authorizations in context-aware task allocation[3]. Sun *et al.* propose and enforce flexible security policies for active cooperation[4]. Hong *et al.*[5] considers various computing context, user context, and physical context, and propose a conceptual model and methodology for adapting existing enterprise services into ubiquitous ones. The authors in Ref.[6] discuss the problem of authorizations under qualification and constraints, as well as to present an algorithm to find a feasible authorization solution. Although the idea of user attributes based authorization in these works is similar with the consideration of user qualification in task-user assignments in our work, there is a distinct difference. They do not consider the geographic information about users and event sites. The task performers in their solutions may not be the most suitable group to quickly respond in emergency situations.

Our work is also related with the business process management. Some discuss how to assign tasks to users under pre-specified constraints[7]; and some discuss the authorization management in business processes based on Web services[8,9]. Mendling *et al.*[10] describe how SoD constraints can be expressed for BPEL4People workflows. Paci *et al.*[9] investigate the resiliency problem in WS-BPEL business processes as well as considering separation of duty and binding of duty constraints. In Ref.[11], Wang and Li proposed the Role-and-relation-based access control (R2BAC) model for workflow systems considering both user's relationships and role memberships when making access decision. However, from a conceptual standpoint, our considerations and method are significantly different. Their works mostly support the user task assignments for a predefined workflow, while we focus on the secured task allocation in an active collaboration, where the required tasks and the candidates for each task are unknown in advance and may vary at any moment. Specially, although the workflow satisfiability problem under multiple constraints in Ref.[11] seems similar with our problem, a distinct difference is that we take users' geographic position into consideration for the authorization decision such that the optimal authorization solution could be found for quick response, which was not discussed in their work.

Recently, the concept of user location has received attention by access control researchers. This has constituted in a new class of access control models called Location-based access control (LBAC). LBAC technologies allow taking users physical location into account when determining their permissions. Ardagna *et al.*[12] integrate location-based conditions into access control by checking the requester's location as well as credentials. Ray *et al.* extend the Mandatory access control (MAC) by incorporating user location information in determining whether a subject can gain access to a given object[13]. The widely adopted Role based access control model (RBAC)[14] is also extended to support certain geographic requirements[15], such as GEO-RBAC[16]. What these works concerned is determining whether to authorize users access rights when they are in a specific position by evaluating the predefined location based conditions. However, under emergencies, quick enforcement of tasks is desired and the performers are required to arrive quickly at the scene. The most important requirement in such scenario is how to coordinate users according to their physical positions rather than to make an authorization decision after a user is already there, which was not addressed in their work. They either do not provide an access control mechanism supporting authorizations for quick collaborative response. Although the quick response problem was discussed in the preliminary version of this paper[2] as well as in the Refs.[17, 18], the authors did not consider the duration of each task as a limiting factor. In practice, each task in a business process may last a period of time and the sequence between them influences the total response time. Thus, the problem and the proposed algorithms in this paper are quite different with regards to this.

III. The Quick-response Task Allocation Model

In this section, we overview our proposed Context-aware authorization model (CAM for short), which supports the runtime allocation of tasks for urgent instance[2]. This is an extension to the most widely adopted Role-based access control model (RBAC)[14]. In CAM, roles are granted with access rights to perform some tasks and are hierarchically inherited. Users are the subjects to be assigned responsibilities to perform certain job functions through roles memberships. Let U and R respectively denote the set of users and roles in a given system. The user role assignments $UR \subseteq U \times R$ are implemented by evaluating a user's attributes against a role's qualification conditions. This mechanism can ensure the flexibility of authorization management since one may adjust the assignments by redefining the qualification conditions. By role hierarchies, the users who are the members of a senior role are also the members of its juniors, while all permissions assigned to a junior role are inherited by its seniors. However, whether a user definitely execute a task through the assigned or inherited role in an instance depends on the event and user context, as well as the specified security constraints.

Each task includes a set of necessary permissions, namely the approvals of a user to operate on one or more protected objects like a database table when achieving the task objective. Let A represent the set of tasks in a system. We define the predicate $dur(a:A)$ to return the average required duration of fulfilling task a, which can be estimated according to daily experiences. We adopt $\mathrm{DUR}_A = \{(a, dur(a)) | a \in A\}$ to denote the set of durations for tasks in A. We also define the following two predicates $ActU(u:U) \rightarrow 2^A$ and $UserA(a:A) \rightarrow 2^U$ to calculate the tasks that user u is allowed to perform and to calculate all the users who are authorized to perform task a, respectively.

Constraints are a fundamental mechanism in RBAC systems to enforce high-level security requirements. In CAM, security constraints are specified on tasks since they are often the objects of the fundamental operations in a system, and should be satisfied on both static task-role assignments and runtime task-user allocation. The most frequently considered constraints are Mutual exclusion (ME) and Binding of duty (BD), which have been well established in previous literature Refs.[1, 9, 6]. A Mutual Exclusion constraint is defined as $ME(A_s)$, where $A_s \subseteq A$, that requires a user being allowed to perform only one task in A_s. For example, $ME(\{Issue_{check}, Init_{check}\})$ enforces that a user is only allowed to execute either initiate a check or issue the check. This constraint is a powerful means to limit the distribution of critical permissions and is widely used to support the Separation of Duty policy or to enforce the conflict-of-interest policy. A Binding of Duty constraint, defined as $BD(A_s)$, where $A_s \subseteq A$, restricts the set of tasks A_s being performed by the same user. The purpose of such constraint is to simplify the management of users and roles by requiring a user taking on a series of related responsibility. For example, $BD(\{Init_{proj}, Modi_{proj}\})$ requires that any user who initiates a project must have the right to modify the project.

Since there often coexist multiple constraints in a system, we need to ensure no conflict between them. For example, if a ME constraint requires that no user is authorized for both tasks a_1 and a_2, yet a_1 and a_2 are also associated with a BD constraint (*i.e.* they have to be performed by the same user), it is impossible to assign these tasks to users without violating these constraints. Therefore, the condition $|A_s \cap A'_s| \leq 1$ must hold for any pair of constraints $ME(A_s)$ and $BD(A'_s)$ so as to ensure the consistency of constraints. Furthermore, to avoid the ambiguity of multiple ME constraints, we require no common task existing between two ME constraints. Overall, the above entities and relationships together present a system environment, which is called the configuration.

Definition 1 [Configuration] A configuration is given as a tuple $\langle U, R, RH, UR, A, C, DUR_A \rangle$, where U is a set of users, R is a set of roles, $RH \subseteq R \times R$ defines role hierarchies, UR is the set of user-role assignments that any $(u, r) \in UR$, where $u \in U$ and $r \in R$, represents u is qualified for r, A represents the set of tasks, C is a set of security constraints that should be followed in any instance and each constraint is in the form $ME(A_s)$ or $BD(A'_s)$, A_s, $A'_s \subseteq A$, and DUR_A is the set of task durations.

In CAM, an instance of urgent collaborative process is defined as a dynamically customized set of tasks, denoted as DA, under the system context of event location and user location. To coordinate the task performers, access control systems have to be flexible and efficient in runtime task allocation via authorizations. When more than one available candidate is qualified for a task, the fastest candidate should be selected for quick response purpose. In practice, the most appropriate users are highly relevant with the context of users and events, such as the event location, users' location, time, users' status (ready or busy), *etc.* A possible solution of computing user delay is by the location context, which is crucial for the purpose of quick response and can be easily obtained by sensors such as Global positioning system (GPS) technologies. Portable mobile devices such as cell phones, PDAs, and notebooks can also provide the abundant context by collecting and interpreting user's interaction response[5,19]. Previous literature has well investigated the location based predicates and use them to predict user delay under consideration of other environment contexts such as traffic[12]. In the following, we adopt the predicate $Delay(u)$ to denote the delay of user u before arriving at the current scene.

As a result, whether a user is assigned to a task in an emergency event is based on four aspects: (1) user's available status at that moment, (2) user delay, (3) user qualification, and (4) security constraints. We adopt the set $Context^U_{DA} = \{(u, Delay(u)) | u \in U \wedge u.status = ready\}$ to denote the delay for each available user in U arriving at some point under event DA, which reflects the first two aspects of context. In the next section, we discuss how to identify such appropriate users for the quick collaborative response in an event.

IV. Towards Quick Collaborative Response

In this section, we discuss the Quick-response task allocation problem (QTAP) to find a task-user allocation solution that aims at the execution of an emergent process instance in the quickest time. We begin our discussion with an example.

1. An illustrative example and runtime task allocation

Example 1 Here is an illustrative process of airplane maintenance. The given dynamically customized set DA of tasks and their average duration time are listed below.

$DA = \{a_1, a_2, a_3, a_4, a_5\}$, where

a_1 : Check and maintain the weather radio and GPU

a_2 : Check and maintain the engine and landing gears

a_3 : Check and maintain the engine and TAT probe *etc.*

a_4 : Generate the report of airplane condition.

a_5 : Recheck the findings of vulnerability.

$DUR_A = \{(a_1, 20\text{mins}), (a_2, 10\text{mins}), (a_3, 30\text{mins}), (a_4, 25\text{mins}), (a_5, 15\text{mins})\}$.

Here, $(a_1, 20\text{mins})$ means the average period of fulfilling task a_1 is 20 minutes, also denoted as $dur(a_1) = 20$. The set C of involved security constraints are also given. $C = \{ME(a_2, a_3), ME(a_4, a_5), BD(a_1, a_4)\}$.

On the assumption that all qualified users are on the event site and each task is performed by a different person, these tasks can be started immediately and performed in parallel. Thus the elapsed time of fulfilling all tasks, denoted as $Elapsed\_time(DA)_{on\_site}$, is the maximum of each task duration, namely $Elapsed\_time(DA)_{on\_site} = \max\{dur(a_1), dur(a_2), dur(a_3), dur(a_4), dur(a_5)\} = \max\{20, 10, 30, 25, 15\} = 30(\text{mins})$.

However, due to some unavoidable reasons, some users may delay their attendance and some may be unavailable at the moment of the event. Thus, the elapsed time (since event happens) of fulfilling a task should include both the task duration and the user delay. If a user performs more than one task, the elapsed time would be the summation of these tasks' durations with his attendance delay.

For simplicity, we omit the concrete user set, role set and role hierarchies for the whole system. Instead, we are supposed to give the available qualified users for DA, denoted as $UserA_{ready}$. Obviously, $UserA_{ready}(a)$ is a subset of the set of qualified users for a, that is

$UserA_{ready}(a) \subseteq UserA(a)$.

$UserA_{ready}(a_1) = \{Alice, Bob\}$,

$UserA_{ready}(a_2) = \{Alice, Caro\}$,

$UserA_{ready}(a_3) = \{Caro, Dan\}$,

$UserA_{ready}(a_4) = \{Alice, Bob\}$,

$UserA_{ready}(a_5) = \{Dan, Bob\}$.

The given context is $Context^U_{DA} = \{(Alice, 15\text{mins}), (Bob, 20\text{mins}), (Caro, 40\text{mins}), (Dan, 30\text{mins})\}$.

Under the above system configuration and context, all feasible solutions of task allocation that satisfy the security constraints are enumerated in Table 1. The elapsed time of fulfilling the tasks in DA under each solution is also calculated. For example, in solution s_1, we have $(Alice, a_1)$ (*i.e.* Alice performs a_1), $(Alice, a_2)$, $(Caro, a_3)$, $(Alice, a_4)$, and (Bob, a_5). Since *Alice* has to perform three tasks a_1, a_2 and a_4 one by one in this solution, her total elapsed time is then: $delay(Alice) + dur(a_1) + dur(a_2) + dur(a_4)$. Overall, the elapsed time of fulfilling all tasks in DA under s_1, denoted as $Elapsed\_time(DA)_{s_1}$, should be the maximum of each user and is calculated as:

$Elapsed\_time(DA)_{s_1} = \max\{delay(Alice) + dur(a_1) + dur(a_2) + dur(a_4), delay(Caro) + dur(a_3), delay(Bob) + dur(a_5)\} = \max\{15 + 20 + 10 + 25, 40 + 30, 20 + 15\} = 70(\text{mins})$

Table 1. Task allocation and the elapsed time of fulfilling DA

Solution ID	Task allocation in each solution a_1, a_2, a_3, a_4, a_5	Elapsed time
s_1	$Alice, Alice, Caro, Alice, Bob$	70
s_2	$Alice, Alice, Caro, Alice, Dan$	70
s_3	$Alice, Alice, Dan, Alice, Bob$	70
s_4	$Alice, Alice, Dan, Alice, Dan$	75
s_5	$Alice, Caro, Dan, Alice, Bob$	60
s_6	$Alice, Caro, Dan, Alice, Dan$	75
s_7	$Bob, Alice, Caro, Bob, Dan$	70
s_8	$Bob, Alice, Dan, Bob, Dan$	75
s_9	$Bob, Caro, Dan, Bob, Dan$	75

For other task-user assignment solutions, the elapsed time are listed in Table 1. From this example, we observe that the elapsed time for fulfilling the same set of tasks varies with the task allocation, which is influenced by the user delay and the number of tasks that a user performs. To achieve the quick-response purpose, we should select the group of users under security constraints with the minimum elapsed time for them to fulfill the tasks. Now, we introduce the notation of Quick-response solution for this purpose.

Definition 2 (Quick-response solution) Given a configuration $\langle U, R, RH, UR, A, C, DUR_A\rangle$, a set DA of tasks, and a context $Context^U_{DA}$, $URA \subseteq U \times A$ is called a Quick-response solution (QR Solution for short) if and only if the following four conditions hold: (1) Every task $a \in DA$ is assigned to one user; (2) For each $(u, a) \in URA$, $u.status = ready$ and $u \in UserA(a)$; (3) No constraint in C is violated, and (4) The elapsed time for the selected users in URA to fulfill the tasks in DA is minimal under $Context^U_{DA}$.

The elapsed time of fulfilling all tasks under QR Solution is called the optimal time in this paper. Thus, a meaningful problem is to verify whether there exists a feasible task-user allocation solution for the given DA under configuration $\langle U, R, RH, UR, A, C, DUR_A\rangle$; and when there are multiple feasible solutions, which one is the QR Solution under $Context^U_{DA}$.

2. The quick-response task allocation problem

Definition 3 (QTAP) Given a set $DA \subseteq A$ of tasks, a configuration $\langle U, R, RH, UR, A, C, DUR_A\rangle$, and a context $Context^U_{DA}$, the Quick-response task allocation problem (QTAP) is looking for the QR Solution for DA.

Theorem 1 QTAP is NP-hard.

Proof This theorem can be proven by reducing the NP-complete Partition Problem to QTAP. In Partition problem, we are given a number of positive integers $S = \{n_1, n_2, \cdots, n_t\}$ and $m = 1/2\sum_{j=1}^{t} n_j$ and are asked whether there exists two disjoint subsets S_1 and S_2 of S such that $\sum_{j\in S_i} n_j = m$ for $i = 1, 2$. Now we construct a configuration $\langle U, R, RH, UR, A, C, DUR_A\rangle$, a set $DA \subseteq A$ of tasks, and $Context^U_{DA}$ as follows:

Let $U = \{u_1, u_2\}$, $R = \{r_1, r_2\}$, $RH = \emptyset$, $C = \emptyset$, $A = DA = \{a_1, a_2, \cdots, a_t\}$. Each user is qualified for every role, namely $UR = \{(u_1, r_1), (u_1, r_2), (u_2, r_1), (u_2, r_2)\}$. That is to say, the set of tasks has the same size with the integer number in the Partition Problem. Each task is assigned to every role. So we could conclude that each user is qualified for every task, namely $UserA_{ready}(a_i) = \{u_1, u_2\}$, $i \in [1, t]$. Let $DUR_A = \{(a_1, n_1), (a_2, n_2), \cdots, (a_t, n_t)\}$. That is to say, the execution duration of a_i is the same with the corresponding integer, $dur(a_i) = n_i (i \in [1, t])$. $Context^U_{DA} = \{(u_1, 0\text{mins}), (u_2, 0\text{mins})\}$. Intuitively, each integer n_i placing in set S_1 in the Partition Problem corresponds to task a_i assigning to user u_1 in QTAP; Otherwise, n_i placing in S_2 indicates task a_i assigning to u_2 in QTAP;

Now, we prove that there is a partition of $S = \{n_1, n_2, \cdots, n_t\}$ if and only if we could find the QR Solution for DA and the optimal time is half of the sum of all task durations. On one hand, assume there is a partition S_1 and S_2 of S satisfying the requirements. We now construct a QR Solution for DA. For every $i \in [1, t]$, if $n_i \in S_1$, we assign a_i to u_1; otherwise, if $n_i \in S_2$, we assign a_i to u_2. Since S_1 and S_2 are disjoint, each task is assigned to only one user, either user u_1 or user u_2. According to above construction, the elapsed time for u_1 fulfilling his/her assigned tasks is $t_{u_1} = delay(u_1) + \sum_{n_i\in S_1} dur(a_i)$; while the elapsed time for u_2 fulfilling his/her assigned tasks $t_{u_2} = delay(u_2) + \sum_{n_j\in S_2} dur(a_j)$. Obviously, $t_{u_1} = t_{u_2}$ and this is the fast way to fulfill all the tasks in DA. Therefore, the assignment solution is optimal for QTAP.

On the other hand, assume there exists a QR Solution for DA, whose optimal time is half of the sum of all task durations. Since there are only two candidate users to perform the tasks in DA and the delay time for them being the site is same, the most efficient way to fulfill all the tasks is to equally assigned the tasks to them according to the task duration (if it exists). The QR Solution must be in the case that DA are divided into two sets, where the sum of task durations in them are equal, namely $delay(u_1) + \sum_{n_i\in S_1} dur(a_i) = delay(u_2) + \sum_{n_j\in S_2} dur(a_j)$. Now we construct the partition of S. For every $i \in [1, t]$, if a_i is assigned to user u_1 in QR Solution, we place integer n_i in the set S_1; otherwise, if a_i is assigned to user u_2, we place integer n_i in the set S_2. Since in QR Solution each task is assigned to only one user, the corresponding integer n_i is placed into either S_1 or S_2, namely S_1 or S_2 are disjoint. This indicates S_1 and S_2 are the partition of set S.

QTAP is intractable means there exist instances that take exponential time in the worst case. However, many encountered instances may be efficiently solvable. Our ultimate goal is to find the QR Solution for a given set of tasks DA under configuration $\langle U, R, RH, UR, A, C, DUR_A\rangle$ and $Context^U_{DA}$. Practically, not every configuration has a feasible task assignment solution for a set DA of required tasks, let alone a choice for quick response. For example, if Alice is the only available user qualified for tasks a_1 and a_2 and there is a constraint $ME(\{a_1, a_2\})$, then there is no way to assign both tasks to a user without violating the mutual exclusion constraint. So, it is necessary to check whether there is a feasible assignment solution that could perform DA. Furthermore, if there are more than one choices the following question arises: which one is the optimal.

3. The solution to QTAP

In this subsection, we present a solution to QTAP by reducing it to the classical scheduling problem. Scheduling deals with the allocation of scarce resources to tasks over time. It is a decision-making process with the goal of optimizing one or more objectives[20]. A scheduling problem can be describe by a triplet $\alpha|\beta|\gamma$, where the α field denotes the machine environment, the β field provides details of processing characteristics and constraints, and the γ field describes the objective to be minimized. One of the classical forms of scheduling problem $p_m|d_j, p_{ij}|C_{\max}$ is on the unrelated parallel machines with deterministic parameters and all tasks are available for processing at time zero, denoted as PMAC. Although PMAC is NP-hard, there are a large number of efficient approximation algorithms in the Ref.[20]. This class of problems can be described as follows. There are m machines $M = \{M_i|i = 1, \cdots, m\}$ and n jobs $T = \{t_j|j = 1, \cdots, n\}$ jobs. Each job t_j has processing time p_{ij} on machine i and is associated with a due date d_j representing the committed completion date. The objective is to find a schedule that minimize the makespan.

The reason that we solve QTAP by reduction method rather than designing a specific algorithm is that we found the common characteristics between these two problems. If we make some tricks on the construction, QTAP can transformed to the scheduling problem and therefore we can benefit from the decades of research results of solving the scheduling problem. In the following, we would first make a sanitary check so as to get rid of those unfeasible cases. Then we present the details on the reduction and discuss how to find the optimal one on the feasible cases. Please refer the algorithm in Fig.1 also.

In QTAP, we are given a configuration $\langle U, R, RH, UR, A, C, DUR_A\rangle$, a dynamically generated set DA of tasks and $Context^U_{DA}$. Now we generate the PMAC scheduling problem according to these parameters. Intuitively, users are regarded as machines and task t_j assigning to machine M_i in the scheduling problem indicates that user u_i is assigned the task a_j in the resulting QR Solution.

(1) **Sanitary check and pre-process** For each task $a \in DA$, we enumerate all available qualified users $UserA_{ready}(a)$. If one of such set is empty, currently there is no available qualified user for a. This indicates there is no feasible task assignment solution for DA. To handling the processing time of each task, we may give an approximate bound since there is no strict deadline. The release time can be set time zero, namely each task can be processed at the beginning. The possible maximum of the elapsed time for fulfilling all tasks in DA is $\max\_due = \max_{(u_j\in U\wedge u_j.status=ready)}\{delay(u_j) + \sum_{a_i\in(ActU(u_j)\cap DA)} dur(a_i)\}$, namely the maximum of the summation of tasks' durations that each user is qualified for. The upper bound of due date for each task can be set $\max\_due$ since a user is allowed to take on more than one tasks in DA and to perform them sequentially.

(2) Handling of users and tasks

For each user $u_i \in U$, we set it as a machine M_i and create a

```
For each task $a \in DA$, do
    $UserA_{ready}(a) = \{u \in UserA(a) \wedge u.state = ready\}$;
    If $UserA_{ready}(a) = \emptyset$ Then return False;
$\max\_due = \max_{(u_j \in U \wedge u_j.state=ready)}\{delay(u_j) +$
    $\sum_{a_i \in (ActU(u_j) \cap DA)} dur(a_i)\}$; $M = \emptyset$.
For each user $u_i \in U$, do
    Set it as a machine $M_i$, $M = M \cup \{M_i\}$
    create a new task $t_{i'}$ denoting $u_i$. Set $d_{t_{i'}} = \max\_due$;
    $p_{ii'} = Delay(u_i)$; $p_{ji'} = \infty$ for $j \neq i$
For each BD constraint $c = BD(A_s)$, do
    Create a new task $a'$; $dur(a') = \sum_{a \in A_s} dur(a)$;
    $UserA_{ready}(a') = \bigcap_{a \in A_s} UserA_{ready}(a)$;
    If $UserA_{ready}(a') = \emptyset$ Then return False;
    For each $ME(B_s)$ constraint
        If $A_s \cap B_s \neq \emptyset$, Then replace $ME(B_s)$ with
        $MRE((B_s - A_s \cap B_s) \cup \{a'\})$
    $DA = (DA - A_s) \cup \{a'\}$;
For each task $a_j \in DA$, do
    Set $d_j = \max\_due$.
    For each $M_i \in M$
        If $u_i \in UserA_{ready}(a_j)$ then set $p_{ij} = dur(a_j)$;
        otherwise set $p_{ij} = \infty$.
For each $ME(A_s)$ constraint, do
    For each task $a_j \in A_s$ do $d_j = due(a_j)$.
```

Fig. 1. The reduction of QTAP to the scheduling problem under the given DA, $Context^U_{DA}$, and $\langle U, R, RH, UR, A, C, DUR_A\rangle$

new task $t_{i'}$. Let $d_{t_{i'}} = \max\_due$; $p_{ii'} = Delay(u_i)$; and $p_{ji'} = \infty$ for $j \neq i$ (∞ can be a very larger integer than the task duration and user delay). This denotes the special task $t_{i'}$ can be performed only on M_i. For each task $a_j \in DA$, if user u_i is qualified for it, namely $u_i \in UserA_{ready}(a_j)$, we set the processing time of t_j on the corresponding user machine M_i is $p_{ij} = dur(t_j)$; otherwise $p_{ij} = \infty$. This ensures that a task being allowed to perform only by qualified users. Let $d_j = \max\_due$.

(3) Handling of security constraints Since we have discussed the constraint consistency in last section, here we assume there is no conflict existing in the constraint set C. For each binding of duty constraint $BD(A_s)$, we check the candidate user sets for all tasks in A_s. If there is no intersection among them, namely $\bigcap_{a \in A_s} UserA_{ready}(a) = \emptyset$, it is unfeasible to make a task assignment solution for DA without violating this BD constant since the binding constraint requires these tasks to be always performed by the same user. In this case, we return answer False. Otherwise, when the intersection set is not empty, we generate a new task say a' to replace all the tasks in A_s, and set its duration to the sum of task duration in A_s as these tasks only can be performed one by one sequentially. Formally, $dur(a') = \sum_{a_i \in A_s} dur(a_i)$. At the same time, we replace every occurrence of each task $a_i \in A_s$ in other ME constraint with a'. For example, if there is a constraint $BD(\{a_1, a_2\})$, the qualified users for a_1 and a_2 are $UserA_{ready}(a_1) = \{u_1, u_2, u_3\}$ and $UserA_{ready}(a_2) = \{u_4, u_2, u_3\}$, we would create a new task a' and set the intersection set $\{u_2, u_3\}$ as its candidate users, as well as $dur(a') = dur(a_1) + dur(a_2)$. If there is also a constraint $ME(a_1, a_3)$, it would be replaced by $ME(a', a_3)$.

For each $ME(A_s)$, we need to ensure each task $a \in A_s$ being assigned to a different machine in the scheduling problem. The processing time of these tasks should be assigned overlapped such that a candidate machine can only process one of them at the same time. So, for each task $a_j \in A_s$, we specify the due date $d_j = dur(a_j)$ such that any two of them cannot be processed by the same machine. Since the tasks involved in any two ME constraints are different, above proposal guarantees that the ME constraint always satisfiable.

Obviously, this reduction can be processed with the polynomial time complexity $O(|DA| * |U| + Max^2_{A_s})$, where $|U|$ and $|DA|$ are the size of the available user set and the size of DA. Max_{A_s} is the max set size $|A_s|$ of $ME(A_s)$ and $BD(A_s)$ constraints, namely $Max_{A_s} = Max_{ME(A_s), BD(A_s) \in C}\{|A_s|\}$. In general, if an optimal solution is found for the scheduling problem, we construct the optimal task assignment for QTAP in the following way: if task t_j is arranged to machine M_i, we put (u_i, a_j) into the resulting Quick-response solution.

V. QTAP Extension

In this section, we present a useful extension to the Quick-response task allocation problem. As we have mentioned in Section IV, the emergency process is defined as a dynamically customized set of tasks without associated specific relationships. However, in practice, there may exist dependency relationships between tasks, such as sequential, parallel, or elective. To capture this, we extend our definition to introduce the partial order relationship on DA, denoted as $BP = \langle DA, \lhd\rangle$, where $\lhd \subseteq DA \times DA$, which is widely adopted in the domain of business process management. The notation $a_i \lhd a_j$, where $a_i, a_j \in DA$, means task a_i should be performed before a_j. a_i is called the previous task of a_j, while a_j is called the subsequent task of a_i. The common assumption under this definition is that there exists no circle in $\langle DA, \lhd\rangle$, *i.e.*, relationships like $a_1 \lhd a_2$, $a_2 \lhd a_3$, and $a_3 \lhd a_1$ do not coexist. Previous work has discussed multiple methods to check and remove a circle, such as Refs.[7, 11].

An illustrative example is shown in Fig.3, which is the extension of Example 1 with the dependency relationships between tasks, where $DA = \{a_1, a_2, a_3, a_4, a_5\}$ and $\lhd = \{(a_1, a_2), (a_1, a_3), (a_2, a_4), (a_3, a_4), (a_4, a_5)\}$. The security constraints are $C = \{ME(a_2, a_3), ME(a_4, a_5), BD(a_1, a_4)\}$. In Fig.3, the integers in the squares associated with each task are the task duration, such as a_1 requiring 20 minutes. The available qualified users for each task, namely $UserA_{ready}(a)$, are given in the brackets beside the task and are associated with user delay. For example, $(Alice, 30)$ beside three tasks a_1, a_2 and a_4 denotes $Alice$ is qualified for these tasks and 30 minutes is required for her to be present.

Now, we discuss how the extension may affect the results presented in the previous section. It is easy to see that the problem QTAP remains NP-hard since it is at least as hard as the case without dependencies between tasks. Although the solution in to QTAP presented in the last section can support the preemption relationships between tasks, it is difficult to express the elective requirements. So in this section, we present a different way to solve the QTAP extension problem.

1. Heuristic algorithm

The proposed algorithm for the extension of QTAP is an ant colony optimization algorithm. Please refer to Fig.2. In the preprocess, we introduce a new predicate $Pre\_act(a)$ to enumerate all directly previous tasks of each task a, and define two variables $URAopt$ and URA to record the current optimal solution for QTAP and the temporarily generated feasible solution. We also perform the heuristics on $BD(A_s)$ constraints by setting the set of available candidate users for $a \in A_s$ to the intersection of their available user sets.

In the process of generating an optimal solution, we introduce another heuristic for reducing the computation, to verify the elapsed time of fulling any task being worse than the optimal. The predicates $pre\_time(a)$ and $elapse\_time(a)$ are respectively denote the elapsed time of fulfilling a's previous tasks and the elapsed time of finishing a after the whole process started. We first randomly select a candidate solution URA (*i.e.* user-task allocation) satisfying all security constraints to perform the tasks in DA and record the elapsed time in a temporal optimum C_{opt}, as well as the assignment solution $URAopt$. Then, we enumerate all candidate solutions one by one and see whether it is better than C_{opt}. As soon as we find the elapsed time for fulfilling any task a in DA is larger than the temporal optimum C_{opt}, namely $elapse\_time(a) > C_{opt}$, we discard it immediately. Otherwise, C_{opt} and $URAopt$ are updated when we have a better solution, as shown in step 6.

To calculate the elapsed time of DA under each URA, we start from the tasks with no previous task, namely $Pre\_act(a) = \emptyset$. For each such task a and its available performer u, we have $pre\_time(a) = 0$ and $elapse\_time(a) = dur(a) + delay(u)$. Then we consider the tasks whose previous tasks having been processed, namely $mark(a) = 1$. For any such task a, while its previous task being executed, the performer of a is on his way to the event. This implies there is overlap between the user delay time $delay(u)$ and the elapsed time of fulfilling its previous tasks $pre\_time(a)$. If $pre\_time(a) > delay(u)$, then u is already at the event when the previous tasks end. We do not need to take the performer's delay into account, and therefore $elapse\_time(a) = pre\_time(a) + dur(a)$. Otherwise, $elapse\_time(a) = dur(a) + delay(u)$. When there are multiple tasks directly previous to a, namely $|Pre\_act(a)| > 1$, either being performed in parallel or electively, the elapsed time of fulfilling them should be: $pre\_time(a) = \max\{elapsed\_time(a')|a' \in Pre\_act(a)\}$.

Considering the example from Fig.3, we list all the attempted solutions in Table 2. The evolution of URA_{opt} includes s_1, s_2, and s_4. From these results, we can see that many candidate solutions are discarded before we complete the calculation, since we found them resulting a worse elapsed time than the current optimal one.

Overall, the computational complexity of our algorithm in Fig.2 is $O(|DA| * N_{\max} + |DA|^{N_{\max}})$, where $N_{\max}$ is the maximum size of the available qualified user set for each task. Formally, $N_{\max} = \max\{|UserA_{ready}(a)|,\ a \in DA\}$. Although it is still in exponential complexity, in practice the adoption of above heuristics can greatly reduce the attempts of computing the possible task allocation solutions and thus increase the rate of success. We would also mention that this algorithm can definitely solve the ordinary QTAP problem without dependencies. In that case, the predicate $Pre\_act(a)$ is empty for each task a indicating that a has no previous task. And the whole process of solving QTAP just enumerates all possible user task solutions and find the optimum.

```
Pre-process:
  $URA = \emptyset$; $C_{opt} = \infty$; $URA_{opt} = \emptyset$
  For each task $a$ in DA, do
      $UserA_{ready}(a) = \{u \in UserA(a) \wedge u.state = ready\}$;
      If $UserA_{ready}(a) = \emptyset$ return False;
      Compute the directly previous tasks of $a$:
     $Pre\_act(a) = \{a'|a' \lhd a \wedge a' \in DA\}$;
  For each BD constraint $c = BD(A_s)$, do
      $tmp\_set = \cap_{a_i \in A_s} UserA_{ready}(a_i)$;
      For each $a_i \in A_s$
         $UserA_{ready}(a_i) = tmp\_set$;
Generate an optimal solution:
  Loop: Do
      1. For each task $a \in DA$ do
         $mark(a) = 0$; $elapsed\_time(a) = 0$; $pre\_time(a) = 0$;
      2. Generate a candidate user-task allocation for DA by
      enumeration, i.e. $URA = \{(u,a)|a \in DA \wedge u \in$
     $UserA_{ready}(a)\}$
      3. If $URA$ does not satisfy all security constraints then goto
      step 2 for next candidate
      4. For each $(u,a) \in URA \wedge (Pre\_act(a) = \emptyset \vee (\forall a' \in$
      $Pre\_act(a) \wedge mark(a') = 1))$ do
         $pre\_time(a) = \max\{elapsed\_time(a')|a' \in Pre\_act(a)\}$;
         $mark(a) = 1$;
         If $pre\_time(a) >= delay(u)$
             then $elapsed\_time(a) = pre\_time(a) + dur(a)$;
             else $elapsed\_time(a) = delay(u) + dur(a)$;
         If $elapsed\_time(a) > C_{opt}$
             Then goto loop
      5. $elapsed\_time(DA) = \max\{elapsed\_time(a)|a \in DA\}$
      6. If $elapsed\_time(DA) < C_{opt}$ then $C_{opt} =$
      $elapsed\_time(DA)$; $URA_{opt} = URA$;
      End Loop: Until all possible user-task combination have
      been enumerated
```

Fig. 2. The heuristic algorithm to the QTAP extension problem under the given $\langle DA, \lhd\rangle$, $\langle U, R, RH, UR, A, C, DUR_A\rangle$, and $Context^U_{DA}$

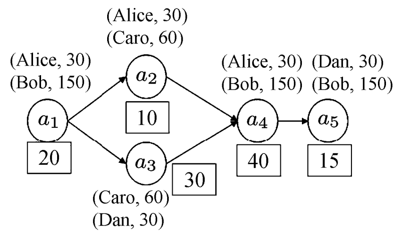


Fig. 3. An example of a business process

Table 2. The elapsed time under different task allocation solutions

Solution ID	Authorization activation solution $a_1,\ a_2,\ a_3,\ a_4,\ a_5$	Elapsed time
s_1	*Alice, Alice, Caro, Alice, Bob*	165
s_2	*Alice, Alice, Caro, Alice, Dan*	145
s_3	*Alice, Alice, Dan, Alice, Bob*	discarded
s_4	*Alice, Alice, Dan, Alice, Dan*	135
s_5	*Alice, Caro, Dan, Alice, Bob*	discarded
s_6	*Alice, Alice, Caro, Alice, Dan*	discarded
s_7	*Bob, Alice, Caro, Bob, Dan*	discarded
s_8	*Bob, Alice, Dan, Bob, Dan*	discarded
s_9	*Bob, Caro, Dan, Bob, Dan*	discarded

2. Experiments

We performed some experiments to verify the efficiency of our algorithm by comparing it with the non heuristic approach. The environment is a personal computer with a dual-core Intel Core 2 Duo E6550 2.33GHz CPU and 1GB RAM, running on the Windows XP SP3 operating system. We programed the prototype with Visual Studio 2008 SP1, MySQL 5.1 and C++.

We generated a set of test cases for our experiments, in which the parameters were chosen to be close to practical cases. In particular, both the number of tasks involved in DA and the number of security constraints should not be very large. However, the delay for users' attendance may quite different. The number of tasks involved in DA ranges from 3 to 20, the number of security constraints ranges from 3 to 6, and the number of candidate users for each task ranges from 3 to 10. The users' delay and task durations vary from 10 to 100 (mins).

We evaluate the impact of the heuristics to reduce the computation complexity. Specifically, we have considered four different test cases: (1) the number of tasks varies from 3 to 10; (2) the max number of potential users for each task varies from 1 to 5; (3) the number of potential user-task assignment combinations varies from 50 to 300; and (4) the number of security constraints varies from 1 to 6. The experimental results are reported in Fig.4. For each aspect, we measured the execution time (in milliseconds) of the two algorithms. Moreover, for each case we have executed more than ten trails and calculated the average response time. From the results, we can see that the heuristic algorithm greatly reduces the execution time than the non-heuristic algorithm. Specially in Fig.4(d), the response time of the heuristic algorithm decreases when the number of security constraints increases. It is easy to understand that for

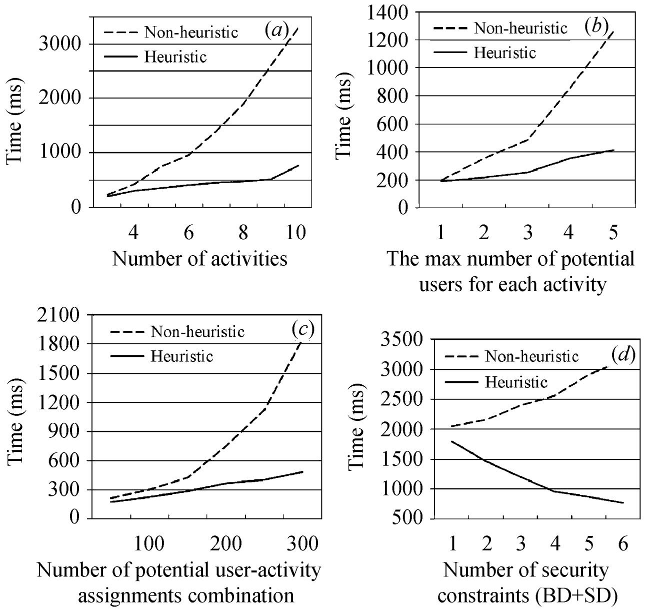


Fig. 4. Non-heuristic approach vs. heuristic approach. (*a*) First test case; (*b*) Second test case; (*c*) Third test case; (*d*) Fourth test case

the same system configuration, the more security constraints are given, the more candidate performers for each task are restricted and discarded. For example, originally $\{Alice, Bob\}$ are candidates for task a_1, while $\{Alice, Carly\}$ are candidates for a_2. If a constraint $BD(a_1, a_2)$ is added, then only *Alice* is the eligible candidate for these tasks. Our algorithm makes a prior check, which markedly reduces the computation.

VI. System Implementation and Discussion

In our preliminary work[2], we have proposed a system architecture by extending a BPEL4People engine with an access control architecture for human tasks. Here we refine this architecture and present some implementation details on important parts. We implement our prototype on a PC with operating system WINDOWS XP SP2, a 2Gz E5200 processor and 2GB RAM. It is based on the BPEL engine ActiveVos 7.1.2 and a MySQL 5.0 database. The key step is the dynamic binding of users and tasks, *i.e.* appointing which user performs each task via which role. In general, for each task of a business process it is required to appoint the performer before the whole process execution. However, in our model the optimal user task assignment solution is generated at runtime. So we designed a Web service that generates the optimal solution according to our algorithms. We also use the Identity service in ActiveVos to retrieve the solution and select task performers according to it. The clients for the users are implemented as Android applications. Specific classes of the Android SDK provide means to retrieve the user's location. The position is denoted by spatial coordinates and is represented as a single point in the Cartesian coordinate. It is then associated with a certain place in the real world. Currently, a user delay is calculated only according to the distance between the user and an event site. More sophisticated context can be simulated if desired. For user notification, we implement two methods. By configuring the parameters of POP3 and SMTP in the mail box of ActiveVos, we can use emails for user notification. We also integrated short message notification for mobile clients.

There are quite a few merits of our solution on the quick collaborative response. Firstly, our method supports portable intelligent equipments, like intelligent mobile phone or PDA and it is suitable to emergent situation. Since in an urgent event, the event location and required tasks are unknown in advance and we could not arrange well all qualified user at the site. Our method integrates the location position technologies and coordinates appropriate users to be there and fulfill the tasks in the fatest manner. Secondly, our model and algorithm can be integrated into a practical business process application since many modern open-source BPEL engines like ActiveBPEL already support human tasks as described in the BPEL4People specification. We also have prototyped a system integrating a business process management system. Another merit of our solution is the integration of the RBAC model since it is widely adopted in practical systems and supports flexible authorization management.

VII. Conclusions

Under emergency situations, quick response is urgently required for persons to arrive at the scene to perform sensitive processes without compromising security policies, where the required tasks may be dynamically customized and the performers are unknown in advance. In this paper, we study the Quick-response task allocation problem (QTAP) to find a task-user allocation solution with consideration of context and security constraints. We study the computational complexity of QTAP and propose an algorithm to select the qualified users for fulfilling the tasks in the fastest manner, while satisfying security constraints. We also discuss an extension of QTAP with the consideration of the dependency relationships between tasks and present a heuristic algorithms to solve this problems. Our experiments have demonstrated the effectiveness and efficiency of these heuristics.

References

[1] N. Li, M.V. Tripunitara, Z. Bizri, "On mutually exclusive roles and separation of duty", *ACM Transaction on Information and System Security*, Vol.35, No.2, pp.1094–9224, 2007.

[2] Y. Sun, M. Farwick and D.K. Chiu, "Constraint-based authorization management for mobile collaboration", in *Proceeding of 2009 IEEE Congress on Services*, Los Angeles, USA, pp.22-29, 2009.

[3] Z. UYao, B. Li and S. Liu, "Role based collaboration authorizing by using ontology", *Chinese Journal of Electronics*, Vol.20, No.3, pp.389–394, 2011.

[4] Y. Sun, B. Gong, X. Meng, Z. Lin and E. Bertino, "Specification and enforcement of flexible security policy for active cooperation", *Elsevier Information Sciences*, Vol.179, No.15, pp.2629–2642, 2009.

[5] D. Hong, D. Chiu, S. Cheng, V. Shen and E. Kafeza, "Ubiquitous enterprise service adaptations based on contextual user behavior", *Information Systems Frontier*, Vol.9, No.4, pp.36–44, 2007.

[6] Y. Sun, Q. Wang, N. Li, E. Bertino, M. Atallah, " on the complexity of authorization in rbac under qualification and security constraints", *IEEE Transaction on Dependable and Secure Computing*, Vol.8, No.6, pp.883–897, 2011.

[7] Y. Sun, J.Z. Huang, X. Meng, "Integrating constraints to support legally flexible business processes", *Springer Information Systems Frontiers*, Vol.13, pp.171–183, 2011.

[8] F. Paci, E. Bertino, J. Crampton, "An access-control framework for ws-bpel", *International Journal of Web Service Research*, Vol.5, No.3, pp.20–43, 2008.

[9] F. Paci, R. Ferrini, Y. Sun, E. Bertino, "Authorization and user failure resiliency for ws-bpel business processes", in *Proceedings of the 6th International Confrence on Service Oriented Computing*, Sydney, Australia, pp.116–131, 2008.

[10] J. Mending, K. Ploesser, M. Strembeck, "Specifying separation of duty constraints in bpel4people processes", in *Proceeding of the 11th International Conference on Business Information*

Systems, Innsbruck, Austriaa, pp.273–284, 2008.

[11] Q. Wang, N. Li, "Satisfiability and resiliency in workflow systems", in *Proceeding of the European Symposium on Research in Computer Security* (*ESORICS*), Dresden, Germany, pp.90–105, 2007.

[12] C.A. Ardagna, M. Cremonini, E. Damiani, S.D.C. di Vimercati, P. Samarati, "Supporting location-based conditions in access control policies", in *Proceeding of the ACM Symposium on Information, Computer and Communications Security Table of Contents*, New York, USA, pp.212–222, 2006.

[13] I. Ray, M. Kumar, "Towards a location-based mandatory access control model", *Computers and Security*, Vol.25, No.1, pp.343–358, 2006.

[14] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman, "Role-based access control models", *Computer*, Vol.29, No.2, pp.38–47, 1996.

[15] I. Ray, M. Toahchoodee, "Aspatio-temporal role-based access control model", in *Proceedings of the 21th Annual IFIP WG 11.3 Working Conference on Data and Apptications Security*, Berlin, Gemany, pp.211-226, 2007.

[16] M.L. Damiani, E. Bertino, B. Catania, P. Perlasca, "Geo-rbac: A spatially aware rbac", *ACM Transaction on Information System Security*, Vol.10, No.1, pp.1–42, 2007.

[17] Y. Sun, D.K. Chiu, "Conyext-aware scheduling of workforce for multiple urgent events", in *The Fourteenth Interational Conference on Computer Supported Cooperative Work in Design* (*CSCWD 2010*), Shanghai, China, pp.629–634, 2010.

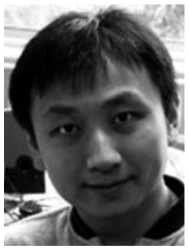
[18] Y. Sun, D.K. Chiu, B. Gong, X. Meng, P. Zhang, "Scheduling mobile collaborating workforce for multiple urgent events", *Journal of Network and Computer Applications* (*JNCA*), 2011.

[19] N. Marsit, A. Hameurlain, Z. Mammeri, F. Morvan, "Query processing in mobile environments: a survey and open problems", in *Proc. of the First International Conference on Distributed Framework for Multimedia Apptications* (*DFMA-05*), Besanon, France, pp.150–157, 2005.

[20] M.L. Pinedo, *Scheduling: Theory, Algorithms, and Systems* (*3rd edition*), Heidelberg, Germany: Springer, 2008.

SUN Yuqing received B.S., M.S. and Ph.D. degrees in Computer Science from Shandong University, China. She is currently a professor in the School of Computer Science and Technology at Shandong University. Her research interests include system security and privacy, social network, Web services, and business computing. (Email: sun_yuqing@sdu.edu.cn)

Matthias Farwick received M.S. degree in Computer Science from the University of Innsbruck, Austria, in 2010. He has worked in several research projects in the area of model-driven security configuration and access control in the Austria, Germany, China, Canada and USA. Currently, he pursues Ph.D. studies at the University of Innsbruck in the area of automated enterprise architecture model maintenance and works as a consultant in the field of IT-Landscape management for QE-LaB Business Services.



Patrick C.K. Hung is an Associate Professor at the Faculty of Business and Information Technology in University of Ontario Institute of Technology in Canada and an Adjunct Professor at the State Key Laboratory of Software Engineering in Wuhan University in China. He is also a Guest Professor at Institute of Computer Science in University of Innsbruck, Austria. He is a founding member of the IEEE International Conference of Web Services (ICWS) and IEEE International Conference on Services Computing (SCC). He is an associate editor of the IEEE Transactions on Services Computing, International Journal of Web Services Research and International Journal of Business Process and Integration Management.

Dickson K.W. Chiu received the M.S. degree in 1994 and the Ph.D. degree in Computer Science from the Hong Kong University of Science and Technology in 2000. He started his own computer company while studying part-time. He has been teaching at several universities in Hong Kong. His research interest is in service computing. The results have been widely published in over 150 papers in international journals and conference proceedings. He is the founding Editor-in-chief of the International Journal on Systems and Service-Oriented Engineering and serves in the editorial boards of several international journals. He co-founded several international workshops and co-edited several journal special issues. He also served as a program committee member for over 70 international conferences and workshops. He is a Senior Member of both the ACM and the IEEE, and a life member of the Hong Kong Computer Society.

JI Guangjun received B.S. degree in Computer Science from Jinan University in 2008 and M.S. degree in Computer Science from Shandong University in 2011. Now he works in the Research Institute of the State Grid Electric Power, which is affiliated to the State Grid Corporation of China (SGCC).