Modelling Workflow Executions under Role-based Authorisation Control

Ligang He[1], Kewei Duan[2], Xueguang Chen[3], Deqing Zou[3], Zongfen Han[3], Ali Fadavinia[1] and Stephen A. Jarvis[1]
1. Department of Computer Science, University of Warwick, Coventry, UK
2. Department of Computer Science, University of Bath, Bath, UK
3. School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China.
liganghe@dcs.warwick.ac.uk

Abstract— **Workflows are often used to represent enterprise-type activities, and authorisation control is an important security consideration in enterprise-level applications. Role-Based Access Control (RBAC) is a popular authorisation control scheme under which users are assigned to certain roles, and the roles are associated with permissions. This paper presents a novel mechanism for modelling workflow execution in cluster-based resource pools under Role-Based Access Control (RBAC) schemes. Our modelling approach uses Coloured Timed Petri-Nets, and various authorisation constraints are modelled, including role constraints, temporal constraints, cardinality constraints, Binding of Duty and Separation of Duty constraints, etc. The interactions between workflow authorisation and workflow execution are also captured in the model. In this paper, the modelling mechanism is developed in such a fashion that the construction of the authorisation model for a workflow can be automated. This feature is very helpful in modelling a large collection of authorisation policies or complex workflows. A Petri-net simulation tool, the CPN-Tool, is utilised to implement the developed modelling mechanism and simulate the constructed model. Both system-level performance (e.g., utilisation of resource pools) and application-level performance (e.g., workflow response time) can be obtained from model simulations. This work can be used to plan system capacity and investigate the impact of authorization policies on system and application performance.**

Keywords: Modelling, Authorisation, Workflow, RBAC

I. INTRODUCTION

Business processes or *workflows* are often used to model enterprise applications [6]. A workflow consists of multiple activities or tasks with precedence constraints. When we design workflow management/scheduling strategies, or evaluate the performance of workflow execution on target resources, it is often assumed that when a task is allocated to a resource, the resource will accept the task and start the execution once the processor becomes available. In reality, however, authorisation policies may be deployed in the organisations and used to specify who is allowed to perform which tasks at what time. When these authorisation schemes are taken into account, the situation can become complex.

A number of authorisation schemes have been presented, see [1][16]. The RBAC (Role Based Access Control) scheme is one of the most popular authorisation schemes. Under the RBAC scheme, users are assigned to certain roles, while the roles are associated with prescribed permissions. Therefore, the organisations can control the users' permissions through these roles.

The following example illustrates RBAC deployed in banking [3]. A bank often uses a variety of computing applications to support its business; these applications may be deployed on a central resource pool (e.g., a computing cluster) at the bank. A workflow may consist of tasks such as credit data checks, automated signature approval, risk analysis and so on. In each task, a particular application has to be launched to perform the corresponding business functions. Under RBAC, an application can only be launched by certain users (i.e., the employees in the bank) assuming certain roles (i.e., official positions such as branch manager or financial advisor). The following authorisation constraints are often encountered in such scenarios: 1) Role constraints: An application may only be run by assuming a particular set of roles; 2) Temporal constraints: A role or user is only activated during certain time intervals (e.g., an employee only covers morning shifts in a particular role); 3) Cardinality constraints: The maximum number of applications running simultaneously under a role is N; 4) Separation of Duty constraints: If Task A is run by a role (or a user), then Task B must not be run by the same role (or user); 5) Binding of Duty constraints: If Task A is run by a role (or user), then Task B must be run by the same role (or user).

It is common to find such authorisation constraints and workflow scenarios in several application domains, such as healthcare systems [16] and in the manufacturing community [8]. All these authorisation constraints may affect the execution behaviors of applications and impact on both application and system performance (e.g. mean response time of workflows, utilisation of the resource pool, etc). The focus of this paper is to model the authorisation and execution of workflows in cluster-based resource pools. The constructed models can then be analysed and/or simulated to obtain system performance in terms of certain metrics. Various types of authorisation constraints are modelled in this paper, including role constraints, temporal constraints, cardinality constraints, Binding of Duty (BoD) and Separation of Duty (SoD) constraints. Various performance metrics can be analysed and obtained from the constructed models, including those for system-oriented performance (e.g., utilisation and throughput) and for application-oriented performance (e.g., response time of workflows).

In this paper, the Colour Timed Petri-net formalism [8][19] is applied to model workflow authorisation and execution. A Petri-net simulation tool, the CPN Tool [25], is utilised to implement and simulate the model. Performance data is then obtained from running model simulations. The work presented in this paper can be used to plan system capacity or estimate application performance in the presence of authorisation policies. This work can also provide insight into how to tune performance by adjusting authorisation policies so as to achieve balance between performance and security overhead.

The remainder of this paper is organised as follows: Section 2 discusses related work; Section 3 introduces the Colour Timed Petri-Net formalism applied in this paper; workflow authorisation and execution is modelled in Section 4; model simulations are provided in Section 5 and, Section 6 concludes the paper.

II. Related Work

Workflow management has been extensively studied and as a result is well documented in related literature [3][6][11][18]. Much of this research is aimed at automating the execution, and enhancing the performance, of workflows in parallel and distributed systems [11][19]. Some of this research has also utilised Petri-nets to model workflow execution, however we note that their work does not formally investigate the performance of workflow execution under authorisation constraints.

Research has also been conducted on the topic of security and authorisation constraints in workflow execution [3][7][19][20]. Some of this research also uses Petri-nets to model authorisation constraints. The work presented in this paper differs from that research in the following respects: First, the work found in [3][20] does not capture the temporal constraints of the roles' availability; in this work, the roles' temporal constraints are modelled. Second, it is assumed in [3][7][14] that a task can only be run under one role. This assumption simplifies the modelling process; however, the assumption is not always true. It may well be the case that a task is allowed to run under a range of roles. The relaxation of this assumption is especially necessary when temporal constraints of roles' availability are taken into account. In so doing, when one role is not available, another activated role may be assigned to run the task, so that the workflow execution can still progress. In this work, the task-role assignments and the task-user assignments are modelled in a flexible fashion, which allows a task to be run under a selection of roles/users. Third, the work in [5][22] does not capture the interactions between workflow authorisation and workflow execution. The work presented in this paper models resource competition and interactions between the authorisation module and the execution module. Finally, previous work only models the execution or authorisation of a single workflow [21][24]. The modelling mechanism developed here however, is able to model the simultaneous execution of multiple workflows.

In previous work [10], we have applied Generalised Stochastic Petri-Net (GSPN) theory to model workflow executions under Role-based Authorisation Control, and then used standard Petri-net analysis techniques to theoretically calculate the performance from the constructed models. Although GSPN is adequate to model the scenarios investigated in that work, it struggles to meet the following new requirements: First, although the temporal constraints are modeled in [10], the temporal constraints have to be regular due to the nature of GSPN. In this paper, the tokens' temporal attributes, the time stamps and the transition guards in the CTPN formalism, are all combined to capture any type of temporal constraints. Second, duty constraints are also modelled in [10]. However, it is assumed that two tasks with duty constraints have to be run in sequence. In this paper, the duty constraints between parallel tasks are modelled. Moreover, Separation of Duty and Binding of Duty are modelled in a uniform fashion in this paper, which enables the assembly of authorisation modules to be automated. Third, the maximum number of workflow instances has to be known in advance before constructing the models and the number cannot be varied after the models have been constructed. In this work, different workflow instances are represented by different token colours and the number of workflow instances is an adjustable model parameter. Finally, it is very difficult for the GSPN modelling scheme in [10] to achieve automated component assembly. It becomes tedious and error-prone when we need to model a large collection of authorisation policies. In this work, the model is constructed in a modular fashion and individual authorisation modulars can be assembled automatically to form the authorisation model for the entire system.

Another major difference between the work in [10] and this work is that in [10] we applied a theoretical approach to calculate performance metrics from the constructed models. In this paper, the developed modelling mechanism has been implemented and performance data can be obtained by running model simulations. These two approaches complement each other, but the simulation approach is more amenable to evaluating large-scale and irregular models.

III. Colour Timed Petri-Nets

The formal definition of a Colour Petri-Net (CPN) differs depending on the source literature [3][25]. The CPN formalism applied in this paper can be formally defined as in Eq.(1) [25],

$$CPN = (P, T, I, O, CS, CF, A, G) \quad (1)$$

where P is a finite set of places; T is a finite set of transitions; I: T → P^{∞} is the input function mapping from a transition to a multiset of places, which are termed *input places* of the transition; O: T → P^{∞} is the output function mapping from a transition to a multiset of places, which are termed the transition's *output places*; CS is a set of colours[1] (a colour in the CPN in [25] is represented as a data type, which can be a primitive data type or a derived data type); CF: P → 2^{CP} is the colour function mapping from places to a subset of the colour set[2]; A: (T × P) ∪ (P × T) →*f*(CF(P)) is the arc function, mapping a directed arc (from a place to a transition or from a transition to a place) to a function of the colours in the place P; G: T → $f(\bigcup_{P\in(IN(T)\cup OUT(T))} CF(P))$ is a guard function mapping a transition to a function of the colours in all places associated with T. The guard function defined in [25] represents the substantive difference from CPN definitions found in other literature [3].

A CPN model consists of places (defined in P) and transitions (defined in T), and a number of directed arcs (defined in IN and OUT). Each place can be marked with a number of tokens, and each token has a data value, which is termed the *token color*. The data value can be a primitive data type, such as an integer and a string, or a complex structure consisting of other primitive data types or complex structures. The number of tokens and the token colours in each place, called a *marking*, represents the state of the model. A place is associated with a data type (termed a *colour set*), which is defined in CS. A place's colour set determines the set of token colours that the tokens in the place are allowed to possess. The model determines whether a token can be fired through the arc functions associated with the arcs (defined in A) and/or guard functions associated to transitions (defined in G). An arc function evaluates to a set of tokens, which determine the type and number of tokens that can pass through the arc. An arc function or a guard function can contain a number of variables as well as the operations (e.g. comparison) and logical operators (e.g. if-else branch) on these variables. Therefore, an arc function or a guard function may evaluate to different values for different tokens.

A Colour *Timed* Petri-Net is an extension of a CPN. In a CTPN, a token can be associated with both a colour and a time stamp. Furthermore, a CPN model has a global timer representing the model time. The time stamp attached to a token indicates the earliest time when the token can be processed. In addition to the arc and guard function, whether a transition can fire or not in a CTPN model, is also controlled by the model's global timer and the tokens' time stamps. The rule is that besides satisfying the arc and the guard functions associated with a transition, the tokens must have time stamps which are no later than the value of the global timer. The model remains at a given model time as long as there are enabled transitions. If there are no such transitions, the global timer is advanced to the earliest model time at which at least one transition is enabled.

IV. Modelling Workflow Authorisation

In this section, various types of authorisation constraint and controls are modelled using CTPNs. These include: 1) Role constraints; 2) Temporal constraints; 3) Role and user assignment; 4) Binding-of-duty constraints; 5) Separation-of-duty constraints; 6) Cardinality constraints; 7) Workflow execution under authorisation control.

A. *Role and user assignment subject to temporal constraints*

As discussed in the banking example above, a constituent task of a workflow involves an application being launched by an employee with an official position. Generally speaking, a role (e.g., an official position) and a user (e.g., an employee) need to be assigned to handle a task of a particular type in the workflow. The model of assigning a role and a user to a task is illustrated in Figure 1. In this model, the transition T_{rt} assigns a role in the P_r place to a task in the P_t place, and deposits a token to the P_{rt} place, indicating the task-role assignment has been established. When a token is deposited into the place P_{urt}, it means that a user has been assigned to the task. We call the model of assigning a role and a user to a task a *role and user assignment module*. It will be shown in Subsection IV.C that these modules can be assembled to construct the authorisation control module for the entire workflow.

A Petri-net model can be formally defined using Eq.1. The remainder of this subsection is dedicated to determining the attributes CS, CF, A and G (ie., token colours, arc and Guard functions) for the role and user assignment module in Fig.1. These attributes together enforce the temporal constraints and role constraints. It is straightforward to determine the attributes P, T, I and O in the model. Therefore, they are omitted for brevity.

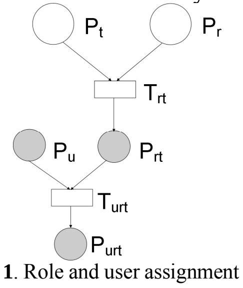


Figure 1. Role and user assignment module

1) *Token colours*

- A token in the P_r place, denoted as *r*, represents a role. The colour of the token *r* is defined as *r* = (*rid*, *D*), where *rid* is the role identifier and *D* is a set of durations in which the role is activated and can be assigned to tasks, i.e.,
$$D = \{[ld_i, ud_i] \mid i \in \aleph\}. \qquad (2)$$
- A token in the P_t place, denoted as *t*, represents a task. The token colour is defined as *t* = (*tid*, *wid*, *e*), where *tid* is the task identifier, *wid* is the identifier of the workflow that the

[1] A colour in the CPN defined in [25] is represented as a data type, which can be a primitive data type (e.g., an integer) or a derived data type (e.g., a record derived from other data types).
[2] 2^{CP} is the power set of the colour set.

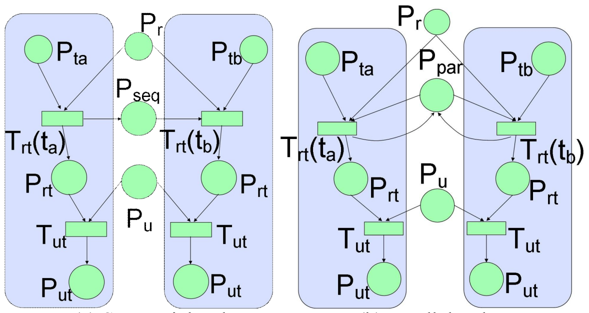
task belongs to, and e is the task's execution time. A time stamp ts is attached to the token t, which is symbolised as $t@ts$. The time stamp represents the earliest time at which the task can be processed. The time stamp of a new task is initialised as the task's arrival time.

- The colour of a token in the P_{rt} place, which is the combination of the color attributes of token t and r, is defined as follows:

$$rt = (tid, wid, e, rid, D)$$

- A token in place P_u represents a user, whose colour is defined as $u = (uid, rid)$, where rid is the role that the user belongs to.
- The colour of a token in the place P_{urt} is defined as

$$urt = (uid, rid, tid, wid, e, D)$$



(a) Sequential tasks (b) Parallel tasks

Figure 2. Duty constraints modules

2) *arc functions*

- The arc function A(P_r, T_{rt}) can be defined as "If $gt \sqsubseteq r.D$ then r"; This expression means that a role is allowed to be assigned to a task only when the value of the global timer is within one of the role's activation durations (denoted by the symbol "$\sqsubseteq$"), i.e., the role is available. It is assumed in this paper that the users have the same availability as their associated roles. But it is straightforward to extend the model to allow individual users to have different (more restricted) availability (e.g., in the case of different employees rotating to cover different shifts of the same official position).
- Other arc functions in the model are defined as A(P_t, T_{rt}) = "t", $A(P_u, T_{ut})$ = "u", $A(P_{rt}, T_{ut})$ = "rt" and A(T_{ut}, P_{urt}) = "urt"

3) *Guard functions*

The guard G, associated with transition T_{rt}, denoted as $G(T_{rti})$, is used to enforce role assignment restrictions. There are two types of restriction. One is the role constraint which specifies the set of roles that are allowed to run a task. This restriction is expressed as "$r.rid \in R(t_i)$", where $R(t_i)$ denotes the set of roles which can run t_i. The other type of restrictions specify that a role is assigned to a task only when the task can run to completion within one of the role's activation durations specified in Eq.2. This restriction is motivated by the fact that most existing workflow description languages, such as BPMN [26], assume that a task in a workflow is atomic. This restriction is formulated as "$[gt, gt+t.e] \sqsubseteq r.D$". Therefore,

$$G(T_{rti}) = r.rid \in R(t_i) \ \&\& \ [gt, gt+t.e] \sqsubseteq r.D. \qquad (3)$$

B. *Binding of duty and separation of duty constraints*

The duty constraints impose restrictions on the role assignments of two tasks in a workflow. Although Binding of Duty (BoD) and Separation of Duty (SoD) represent opposite authorisation behaviors, these two types of duty constraints are modelled in a uniform fashion so that they have the same model structure. We call these the SoD module and the BoD module. The differences between these are essentially in some arc and guard functions, as well as the colour set of some places. The benefit of doing this is that the models for individual duty constraints can be easily assembled to form the authorisation model for the entire workflow.

There are two types of relationship between two tasks in a workflow in terms of precedence constraints: sequential tasks and parallel tasks. Assume task t_a and t_b. If $t_b \in$ Pred(t_a) or $t_b \in$ Succ(t_a) (Pred(t_i) and Succ(t_i) denote the set of tasks which are task t_i's predecessors and successors, respectively), then t_a and t_b are sequential tasks and have to be run in the required order. If $t_b \notin$ Pred(t_a) and $t_b \notin$ Succ(t_a), t_a and t_b can be executed in parallel. Duty constraints are modelled in a different way for these two types of tasks. Their model structures are shown in Fig.2a and Fig.2b, respectively.

1) *Sequential tasks*

As shown in Fig.2a, the duty constraints module consists of the role and user assignment modules for t_a and t_b, connected by place P_{seq}. P_{seq} is one of the output places of transition $T_{rt}(t_a)$ and one of the input places of transition $T_{rt}(t_b)$. The attributes of the role and user assignment modules have been discussed in Subsection A. The attributes related to the new place, P_{seq}, are as follows.

- **Token colours**: The colour of a token in place P_{seq} is defined as $seq = (tid, wid, rid)$, which carries the information of which role has been assigned to task tid.
- **Arc functions**: $A(T_{rt}(t_a), P_{seq})$ and $A(P_{seq}, T_{rt}(t_b))$ are both defined as "seq".
- **Guard functions**: The guard functions associated with $T_{rt}(t_b)$ are different for SoD and BoD constraints. If it is a SoD constraint, $G(T_{rt}(t_b))$ is expressed as Eq.4. If it is a BoD constraint, $G(T_{rt}(t_b))$ is formulated as Eq.5. t in Eq.4 and Eq.5 is a token in place P_{tb}. The condition $seq.wid=t.wid$ is used to guarantee that the same workflow instance is referred to, since this model allows multiple instances of the same workflow to be processed simultaneously. Different instances of a workflow will have different values of wid.

$$seq.wid=t.wid \ \&\& \ seq.rid \neq r.rid. \qquad (4)$$

$$seq.wid=t.wid \ \&\& \ seq.rid = r.rid. \qquad (5)$$

2) *Parallel tasks*

Similar to Fig.2a, a new place, labelled P_{par}, is used in Fig.2b to interface between the role and user assignment module when t_a and t_b are parallel tasks. In the remainder of this subsection we first determine the attributes related to place P_{par}, and then use an example to illustrate the workings of the module.

- **Token colours**: There are two types of tokens in P_{par}: *par_init* and *par*. *par_init* is defined as $par\_init = (tid_a, tid_b, wid)$, while *par* is defined as $par = (tid, rid, wid)$.
- **Guard functions**: If a SoD constraint, the guard function of the transition $T_{rt}(t_a)$ (or $T_{rt}(t_b)$) is formulated as Eq.6. If a BoD constraint, it is expressed as Eq.7.

$$((t.tid = par\_init.tid_a \,||\, t.tid = par\_init.tid_b) \;\&\&\; t.wid = par\_init.wid) \,||\, (t.wid = par.wid \;\&\&\; r.rid \neq par.rid) \quad (6)$$

$$((t.tid = par\_init.tid_a \,||\, t.tid = par\_init.tid_b) \;\&\&\; t.wid = par\_init.wid) \,||\, (t.wid = par.wid \;\&\&\; r.rid = par.rid) \quad (7)$$

- **Arc functions**: $A(P_{par}, T_{rt})$ and $A(T_{rt}, P_{par})$ are defined in Eq.8 and Eq.9, respectively.

$$A(P_{par}, T_{rt}) = par\_init \,||\, par \quad (8)$$

$$A(T_{rt}, P_{par}) = \text{“}\textbf{if } part\_init \textbf{ then } par\text{”} \quad (9)$$

3) *Workings of the duty constraint modules*

The workings of the duty constraint modules and the above expressions are illustrated as follows. When performing the role assignment for a task (e.g., t_a), the model will check whether the other task (e.g., t_b) has been assigned a role. Assume there is a BoD constraint between t_a and t_b, then the place P_{par} will contain a *par_init* token in the model's initial marking. When the T_{rt} transition performs the role assignment for t_a, it will evaluate which token in P_{par} can satisfy Eq.6, and therefore can be fired by the transition. There are two possibilities:

a) If there is a corresponding *par_init* token in P_{par}, which means that t_b has not been assigned a role, then the first part of Eq.6 (i.e., the portion before the second "||") will be evaluated as true. Consequently, the $T_{rt}(t_a)$ transition will remove the *par_init* token from P_{par} and deposit a *par* token back to P_{par} as shown in Eq.9.

b) If there is a *par* token in P_{par}, this means that t_b has been assigned to a role. Further, if there is a role in place P_r whose identifier (i.e., *r.rid*) is the same as the role assigned to t_b (i.e., *par.rid*), the second part of Eq.6 will be evaluated as true. Thus, the BoD constraint is enforced.

C. *Assembling authorisation modules*

One of the biggest advantages of the modelling mechanism developed in this paper is that the authorisation model can be constructed automatically by assembling a set of interacting hierarchical modules. There are clear interfaces and hierarchy structure among the modules. As shown in Fig.2, the duty constraints module consists of the role and user assignment modules for t_a and t_b, interfacing via place P_{par} or P_{seq}. Generally, the assembly procedure is as follows.

Definition 1. $M_k = (P_k, T_k, I_k, O_k, CS_k, CF_k, A_k, G_k)$ is the role and user assignment module for task t_k;

Definition 2. $M = (P, T, I, O, CS, CF, A, G)$ is a module in which j tasks, $t_{i1}, t_{i2}, \ldots, t_{ij}$, have the duty constraints with task t_k;

Definition 3. $M'=(P', T', I', O', CS', CF', A', G')$ is the module that captures the duty constraints between the j tasks in module M and task t_k in module M_k.

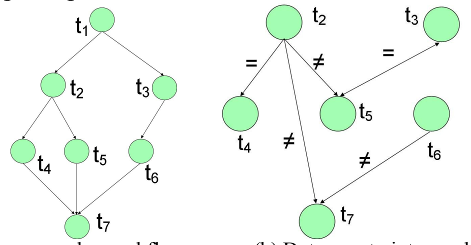
Then, the module M' in Definition 3 can be constructed by assembling M and M_k as shown in Theorem 1.

Theorem 1. Given M_k, M, and M' in Definition 1, 2 and 3, the attributes of M' can be computed as follows.

- $$P' = P \cup P_k \cup \{ \bigcup_{t_{ik} \in Pred(t_k) \cup Succ(t_k)} P_{seq}(t_{i_k}, t_k) \} \cup \{ \bigcup_{t_{i_k} \notin Pred(t_k)} P_{par}(t_{i_k}, t_k) \} \quad (10)$$
- $$T' = T \square T_k \quad (11)$$
- $$CS' = CS \cup CS_k \cup seq \cup \{par, par\_init\} \quad (12)$$
- $$CF' = CF \cup CF_k \cup CF(P_{seq}) \cup CF(P_{par}) \quad (13)$$
- A′ can be computed using Eq.14, where $P_{seq}(t_x, t_y)$ or $P_{par}(t_x, t_y)$ denote the P_{seq} or P_{par} place that is added to model the duty constraints between t_x and t_y.
- I', O' and G′ can be computed using Algorithm 1, where g_{SOD}^{seq}, g_{BOD}^{seq}, g_{SOD}^{par} and g_{BOD}^{par} are the guard expressions specified in Eq.4, Eq.5, Eq.6 and Eq.7, respectively.

Proof. 1) For each task in the module M that has the duty constraint with task t_k, a new place (either P_{seq} or P_{par} depending on whether they have the precedence constraint) is added in the new module M′. So P' can be computed as in Eq.10. 2) There is no need to add new transitions in M'. So T' can be computed as Eq.11. 3) The new colour set of M′ is the union of the colour sets of M_k and M, plus the colours of the tokens in place P_{seq} and P_{par}. So CS' can be computed as Eq.12. Similarly, CF′ can be computed using Eq.13. 4) The arc function set in M' is the union of the arc function sets of M_k and M plus the arcs added between place P_{seq} or P_{par} and the corresponding T_{rt} transitions, as shown in Fig.2. So A' can be computed using Eq.14. 5) Since there are no new transitions added in M', steps 1 to 3 in Algorithm 1 first initialise I', O', and G′ to be the union of I and I_k, O and O_k, and G and G_k respectively. Then in the for-loop, the algorithm adjusts the input, output and guard functions of the individual transitions that have the arc connections with the newly added P_{seq} and P_{par} places. □

The importance of Theorem 1 is that with the formalised equations and algorithm, the process of constructing the authorisation control model can be automated rather than being built manually, which can greatly speedup the modelling process.



(a) An exemplar workflow (b) Duty constraints graph

Figure 3. An exemplar workflow and its duty constraints graph

We use a case study to illustrate how to assemble the individual role assignment modules subject to the duty constraints. The exemplar workflow is abstracted from a loan lending process in a bank [28]. The roles which are involved in the process are listed in Table 1. The workflow consists of 7 tasks, whose topology is shown in Fig.3a. The tasks and role assignment constraints are shown in Table 2.

Table 1 Role descriptions in the loan lending workflow

Role	Description	Role	Description
SM	Second Market Official	BM	Bank Manager
FA	Financial Advisor	CL	Bank Clerk
LB	Loan Broker	UW	Underwriter

Table 2 Task descriptions and role constraints in the workflow

Task	Description	Role constraints
t_1	Updating products and rates	SM
t_2	Product and Rate decision engine	FA/LB/BM
t_3	Collecting data	FA/LB/CL
t_4	Analysing data of third-party 1	FA/LB/CL
t_5	Analysing data of third-party 2	FA/LB/CL
t_6	Analysing business rules	FA/BM
t_7	Underwriting	UW/BM

Assume that the following BoD and SoD constraints are imposed on the tasks, where $r(t_i)$ denotes the role assigned to task t_i.

C_1: $r(t_2) = r(t_4)$;C_2: $r(t_2) \neq r(t_5)$;C_3: $r(t_2) \neq r(t_7)$;
C_4: $r(t_6) \neq r(t_7)$;C_5: $r(t_3) = r(t_5)$;

These duty constraints can be represented as a *duty constraints graph* shown in Fig.3b. In the duty constraints graph, if there are duty constraints between two sequential tasks, a single-headed arrow is used to connect the predecessor to the successor. If there exists duty constraints between two parallel tasks, the two tasks are connected by a double-headed arrow. Applying the module assembly operations described in Eq.10-14 and Algorithm 1, the hierarchy of the authorisation control model for the entire workflow can be constructed as shown in Fig.4, where $M_{urt}(t_i)$ denotes a role and user assignment module for task t_i; please note that there should be a directed arc from the P_r place and the P_u place to every module as shown in Fig.2, these are omitted for the sake of clarity.

In this work we model the fact that the tokens in P_r and P_u are shared by all tasks. This is reasonable since the roles and users are global parameters and should be applied to all tasks in the system. Cardinality constraints, which specify the maximum number of tasks that can be handled at the same time by a role (or a user), can be modelled by the number of tokens representing the role (or the user) in P_r (or P_u).

D. Modelling workflow execution under authorisation

Fig.5 models the execution of the workflow in Fig.3a under authorisation control. The detailed hierarchy of the Authorisation Control Module (ACM) can be found in Fig.4. It can be seen that there are clear interfaces between the actual execution of the workflow and the ACM. When task t_i is ready (i.e. it has no predecessor or its predecessors have been completed), a token t is deposited into place P_{ti}, which is the same P_{ti} place in the ACM. This starts the authorisation process in the ACM for the task. After the authorisation is completed, a *urt* token is deposited by the ACM into the P_{urti} place. The task is now assigned a role and a user, and can undergo the resource allocation procedure.

Algorithm 1. **Calculating *I′*, *O′*** and ***G′*** **from *M* and *M_k***

```
1.  $I' = I \cup I_k$
2.  $O' = O \cup O_k$
3.  $G' = G \cup G_k$
4.  for each task $t_{i_s}$ in $M$ that has BoD or SoD with $t_k$ do
5.      If $t_{i_s}$ □ $Pred(t_k)$ then {
6.          $I'(T_{rt}(t_k)) = I'(T_{rt}(t_k)) \cup \{P_{seq}(t_{is}, t_k)\}$
7.          $O'(T_{rt}(t_{is}))=O'(T_{rt}(t_{is})) \cup \{P_{seq}(t_{is}, t_k)\}$
8.          if there is SoD between $t_k$ and $t_{is}$ then
9.              $G'(T_{rt}(t_k)) = G_k(T_{rt}(t_k))$ && $g_{SOD}^{seq}$
10.         else
11.             $G'(T_{rt}(t_k)) = G_k(T_{rt}(t_k))$ && $g_{BOD}^{seq}$
12.     }
13.     else if $t_{is}$ □ Succ($t_k$) then {
14.             $O'(T_{rt}(t_k)) = O'(T_{rt}(t_k)) \cup \{P_{seq}(t_{is}, t_k)\}$
15.             $I'(T_{rt}(t_{is}))=I'(T_{rt}(t_{is})) \cup \{P_{seq}(t_{is}, t_k)\}$
16.             if there is SoD between $t_k$ and $t_{is}$ then
17.                 $G'(T_{rt}(t_{is})) = G(T_{rt}(t_{is}))$ && $g_{SOD}^{seq}$
18.             else
19.                 $G'(T_{rt}(t_{is})) = G(T_{rt}(t_{is}))$ && $g_{BOD}^{seq}$
20.     }
21.     else { // $t_{is}$ and $t_k$ can be run in parallel
22.         $I'(T_{rt}(t_{is})) = I'(T_{rt}(t_{is})) \cup \{P_{par}(t_{is}, t_k)\}$
23.         $O'(T_{rt}(t_{is})) = O'(T_{rt}(t_{is})) \cup \{P_{par}(t_{is}, t_k)\}$
24.         $I'(T_{rt}(t_k)) = I'(T_{rt}(t_k)) \cup \{P_{par}(t_{is}, t_k)\}$
25.         $O'(T_{rt}(t_k)) = O'(T_{rt}(t_k)) \cup \{P_{par}(t_{is}, t_k)\}$
26.         if there is SoD between $t_k$ and $t_{is}$ then
27.             $G'(T_{rt}(t_k)) = G_k(T_{rt}(t_k))$ && $g_{SOD}^{par}$
28.             $G'(T_{rt}(t_{is})) = G(T_{rt}(t_{is}))$ && $g_{SOD}^{par}$
29.         else
30.             $G'(T_{rt}(t_k)) = G_k(T_{rt}(t_k))$ && $g_{BOD}^{par}$
31.             $G'(T_{rt}(t_{is})) = G(T_{rt}(t_{is}))$ && $g_{BOD}^{par}$
32.     }
```

$$A' = A \cup A_k \cup \bigcup_{t_{is} \in Pred(t_k)} A(T_{rt}(t_{i_s}), P_{seq}(t_{i_s}, t_k)) \cup \bigcup_{t_{i_s} \in Succ(t_k)} A(P_{seq}(t_{i_s}, t_k), T_{rt}(t_{i_s})) \cup$$
$$\bigcup_{t_{is} \notin Pred(t_k) \cup Succ(t_k)} (A(P_{par}(t_{i_s}, t_k), T_{rt}(t_{i_s})) \cup A(P_{par}(t_{i_s}, t_k), T_{rt}(t_k)) \cup A(T_{rt}(t_{i_s}), P_{par}(t_{i_s}, t_k)) \cup A(T_{rt}(t_k), P_{par}(t_{i_s}, t_k))) \qquad (14)$$

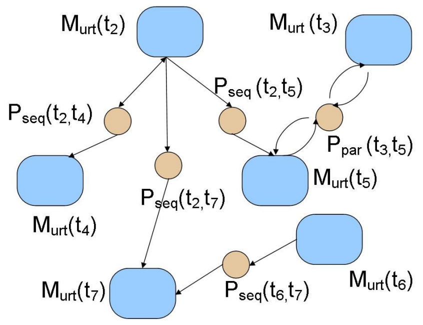


Figure 4. Hierarchy of the authorisation control module for the workflow in Fig.3a

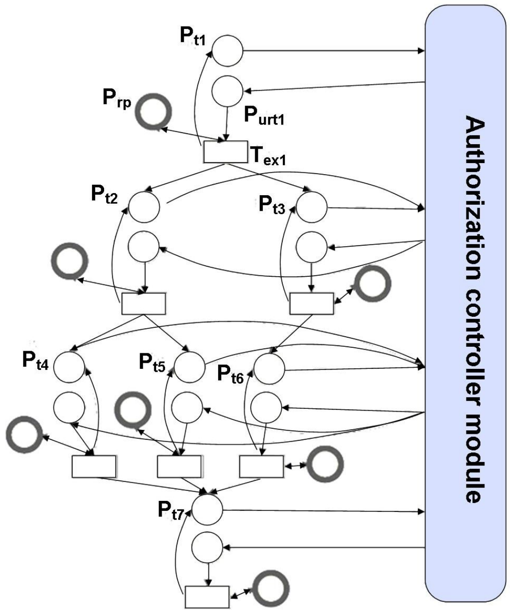


Figure 5. Modelling workflow execution under authorisation control; only the Petri-net components for task *t1* are labelled, and the labels for other tasks are omitted for the sake of clarity); the place *Pnp* represents the node pool and the individual *Pnp* places should be regarded as a single *Pnp* place.

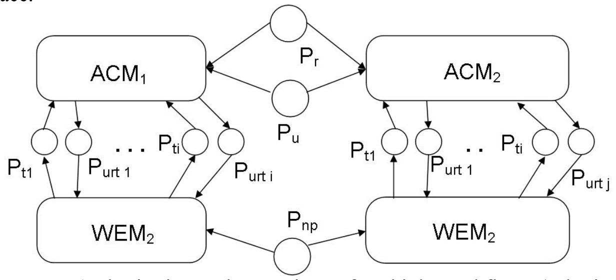


Figure 6. Authorisation and executions of multiple workflows (submitted by different clients); ACM stands for Authorisation Control Module, and WEM stands for Workflow Execution Module

In Fig.5, the place P_{np}, depicted as a bold circle, represents the resource pool. In this place, a token represents a resource (e.g., a compute node), and the number of tokens represents the number of resources currently available in the system. The T_{ex} transition represents the resource allocation and task execution. There is only one resource pool (therefore, a single P_{np} place) in the model, connecting to all T_{ex} transitions. Fig.5 is drawn so that every T_{ex} transition is associated with a separate P_{np} place; this is done for the sake of clarity (to avoid too many arcs crossing the figure). These individual P_{np} places should be regarded as a single P_{np} place. A token in the P_{np} place is defined as $np = nid$.

In this model, the execution of a task is modelled in the following way. If there are tokens in the P_{np} place (i.e., there are free compute nodes) and there is the urt_i token in P_{urti}, the T_{exi} transition fires immediately (indicating the start of t_i's execution) and a t token is deposited into the P_{tj} place (suppose t_j is t_i's child). The time stamp of the deposited token t will be set as the current global time plus t_i's execution time, that is, $t_j@ts = gt + t_i.e$. Therefore, the t_j token is not allowed to be fired until the time duration of $t_i.e$ has lapsed, which simulates the execution of task t_i.

Moreover, after a task is completed, the role and user assigned to the task need to be returned to place P_r and P_u, so that the role and the user can be assigned to other tasks. This procedure is modelled as follows. When T_{exi} fires, an r token is deposited back to the P_r place and a u token to the P_u place. The time stamps of these two tokens are both set as $gt+t_i.e$. Similarly, when T_{exi} fires, an np token is deposited back to the P_{np} place (expressed as double-headed arrows). $np@ts$ is also set as $gt+t_i.e$, which means that although the np token is back to the P_{np} place, the corresponding resource is only allowed to be allocated to other tasks when the simulated execution of t_i has been completed.

If multiple workflows are running (and each workflow can have many instances) in the system, an authorisation control module and an execution module need to be constructed for each workflow. The model structure for authorising and executing multiple workflows is shown in Fig.6. As can be seen from the figure, the different workflow authorisation modules only share place P_r and P_u, and the different workflow execution modules only share the P_{np} place.

V. MODEL SIMULATIONS

The modelling mechanism presented in this paper has been implemented using the CPN Tools [25]. The CPN Tools are a software platform that is able to construct and simulate Petri-net models. CPN provides a flexible mechanism that allows users to monitor a set of places and/or transitions. The runtime status of these places and transitions can be automatically collected during model simulations. CPN can evaluate a constructed Petri-net model in terms of various performance metrics. For example, resource utilisation can be evaluated by monitoring the number of tokens in the place corresponding to the resource pool (which is P_{np} in our model). Assuming the initial marking of the place is n, and during model simulations the average number of tokens in the place observed by CPN is avg_n, then the resource utilisation, denoted as ru, can be calculated as:

$$ru = 1 - avg\_n/n$$

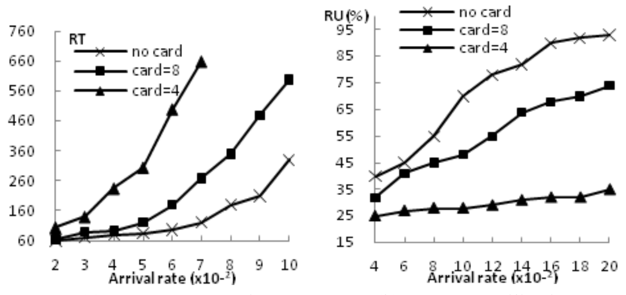
It is also straightforward to determine the response time of a workflow in CPN. The tool can extract the time stamp when a workflow arrives at the system, and the time stamp when the last task in the workflow is completed. The difference between these two time stamps is the response time of the workflow. Based on this information, we can easily calculate the mean response time of all workflows. Other performance metrics that the CPN tools are able to evaluate include throughput, deadline miss rate, etc. In this section, the simulation results are presented in terms of mean Response Time (RT) of workflows and Resource Utilisation (RU) of the resource pool, The performance in terms of deadline miss rate and throughput has the correlated relationship with response time and utilisation, respectively.

In the simulations, the CTPN model is constructed for the loan lending workflow given in Fig.3a. The workflow instances arrive following the Poisson process. The runtime of a task follows an exponential distribution and the mean runtime of task t_1-t_7 is set to be 10, 15, 5, 10, 10, 20 and 25 time units, respectively, based on runtime comparisons among the tasks in reality [28].

1) Impact of Cardinality constraints

Fig.7 shows workflows' RT and RU with the existence of cardinality constraints as the arrival rate of workflow instances increases. In order to investigate the impact of cardinality constraints, no other constraints are imposed except the duty constraints given in Fig.3b. Each role has 4 users. There are 8 homogeneous resources in the pool.

It can be observed from Fig.7a that when cardinality constraints are imposed, RT increases. This is because it is more likely that the tasks have to wait not only for resources, but also for the availability of roles. This qualitative analysis seems obvious. However, only through the modelling approach and simulation results we can acquire quantitative insight into how much impact a particular authorisation constraint can have. For example, suppose the workflows' RT is desired to be no more than 300 time units. When there is no cardinality constraint, the system can accommodate a workflow stream with a mean arrival rate of up to 0.095. However, when the cardinality constraint is 8 and 4, the workload level that the system is able to handle is reduced to approximately 0.075 and 0.045, respectively.



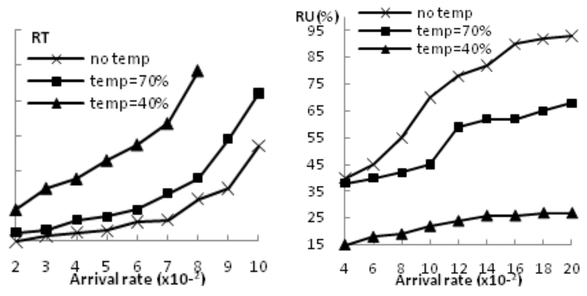
(a) Mean response time (b) Resource utilisation

Figure 7. Impact of cardinality constraints on mean response time and resource utilisation; "no card" means there are no cardinality constraints; "card=8" and "card=4" mean that the maximum number of tasks that a role can run simultaneously is 8 and 4, respectively.

As can be observed from Fig.7b, the cardinality constraints also have a negative impact on RU. Quantitatively, when there are no cardinality constraints, the resource utilisation is approximately 90% as the arrival rate increases. However, in the case of "card =8", the utilisation can only reach around 75%; the performance is even worse (approximately 35%) when the cardinality parameter is 4. These results suggest that even if there are free resources in the system, the tasks cannot make use of them because of the unavailability of roles due to the cardinality constraints. Through modelling and simulation, we are able to determine an optimised number of resources when we plan a system's capacity to support workflow executions under pre-specified authorisation constraints.

2) Impact of Temporal constraints

Fig.8 demonstrates the impact of temporal constraints on performance in terms of RT and RU as the workflow instances' arrival rate increases. The temporal constraints on roles are set in the following way in the simulations in Fig.8. For each role, a time duration is selected from a period of 100 time units. The selected time duration occupies the specified percentage of the 100 time units. The starting time of the selected duration is chosen randomly. For example, to select a time duration which is 70% of 100 time units, the starting point of the duration is randomly selected from 0 to 30%×100 time units.



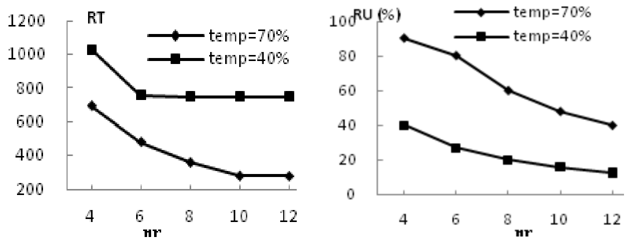
(a) Mean response time (b) Resource utilisation

Figure 8. Impact of temporal constraints on mean response time and resource utilisation; "no temp" means there are no temporal constraints; "temp = 70%" and "temp = 40%" means that the roles are available for 70% and 40% of the time, respectively.

It can be observed from Fig.8 that temporal constraints have a negative impact on performance in terms of both RT and RU, as to be expected. Another interesting observation is that under temporal constraints, the performance seems to be less affected compared with the performance under cardinality constraints. This may be because the roles' availability period may overlap. Therefore, when one role is not available, the task may be able to find another role.

3) Impact of the number of resources

Figure 9 shows the impact on performance of increasing the number of resources (*nr*) when there exist cardinality and temporal constraints. As can be observed from Fig.9a, when the temporal constraint is 70%, RT decreases as *nr* increases, until *nr* reaches 10. Also, when temp=40%, RT decreases until *nr* reaches 6. This is because when *nr* is greater than a threshold, the workflow executions are mainly hampered by the authorisation constraints. These results demonstrate that with the modelling and simulations, we can quantitatively investigate the impact of the deployed authorisation policies, and therefore can potentially balance the authorisation policy settings to remove performance bottlenecks. The trend in Fig.9b is different from that in Fig.9a. In Fig.9b, RU keeps decreasing as *nr* increases. A closer observation shows that RU decreases linearly after *nr* is greater than 10 and 6 in the case of temp=70% and temp=40%, respectively. This is because of the same reason discussed above, i.e., an execution bottleneck is now caused by authorisation and therefore, the increased resources will be largely idle. This result suggests that when planning system capacities, we should take authorisation into account and avoid over-provisioning unnecessary resources.



(a) Mean response time (b) Resource utilisation

Figure 9. Effect of increasing the number of resources (nr) in the existence of temporal and cardinality constraints; card=8; arrival rate is 7×10^{-2}

VI. CONCLUSIONS

This paper employs Colour Timed Petri Nets (CTPN) to model workflow execution under authorisation constraints in cluster-based resource pools. Various authorisation constraints are modelled in this paper, including role constraints, temporal constraints, cardinality constraints, separation of duty and binding of duty constraints. The model allows a task in the workflow to run under a selection of roles and users, and it also allows multiple workflows from different clients to be authorised and executed in the system simultaneously. The model is constructed in a modular fashion. Therefore, it is easy to model a large collection of authorisation policies and complex workflows using the modelling approach developed in this paper. The modelling mechanism has been implemented using the CPN Tools. Various performance metrics can be computed from the constructed model, including those for system-oriented performance and application-oriented performance.

VII. ACKNOWLEDGEMENTS

This work is partly sponsored by the Leverhulme Trust Research Project Grant (RPG-101), the National Natural Science Foundation of China (60773188 and 60973038), the Fundamental Research Funds for the Central Universities of China (HUST: 2010MS017).

REFERENCES

[1] G. Ahn and R. Sandhu, "Role-Based Authorization Constraints Specification," ACM Trans. Information and System Security, 3(4), 2000.
[2] R. Alfieri, et. al., "VOMS, an Authorization System for Virtual Organizations", 1st European Across Grids Conference, 2003.
[3] V. Atluri, "A Petri net based safety analysis of workflow authorization models", Journal of Computer Security, 8(2/3): 209-240, 2000
[4] R. Baker, L. Gommans, A. McNab, M. Lorch, L. Ramakrishnan, K. Sankar and M. Thompson, "Conceptual Grid Authorization Framework and Classification", in 7th Global Grid Forum Workshop (GGF7), March, 2003
[5] X. Zhao, Z. Qiu, C. Cai, H. Yang, "A formal Model of Human Workflow", the 2008 IEEE Intl. Conference on Web Services, pp. 195-202.
[6] D. Chakraborty, V. Mankar, A. Nanavati, "Enabling Runtime Adaptation of Workflows to External Events in Enterprise Environments", the 2007 IEEE International Conference on Web Services, pp.1112-1119.
[7] J. Crampton, "A reference monitor for workflow systems with constrained task execution", Proceedings of the tenth ACM symposium on Access control models and technologies, pp. 38-47, 2005
[8] Y. Jin, S. Reveliotis, "A generalized stochastic Petri net model for performance analysis and control of capacitated reentrant lines", IEEE Transactions on Robotics and Automation, 19(3): 474 - 480, 2003
[9] G. Della-Libera, P. Hallam-Baker, M. Hondo, T. Janczuk, et al. Web Services Security Policy Language (WS-SecurityPolicy), http://www-106.ibm.com/developerworks/library/ws-secpol/. 2002.
[10] L. He, M. Calleja, M. Hayes, S.A. Jarvis, Performance prediction for running workflows under role-based authorization mechanisms," Proc. of the 2009 IEEE International Symposium on Parallel & Distributed Processing, IEEE Computer Society Press.
[11] L. He, S Jarvis, D. Spooner, D. Bacigalupo, G. Tan, G. Nudd, "Mapping DAG-based Applications to Multiclusters with Background Workload", Proceedings of the 5th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05), 9-12 May 2005, Cardiff, UK
[12] L. He, S. Jarvis, D. Spooner, H. Jiang, D. Dillenberger, G. Nudd, "Allocating Non-real-time and Soft Real-time Jobs in Multiclusters", IEEE Transactions on Parallel and Distributed Systems, 17(2):99-112, 2006
[13] S. Indrakanti and V. Varadharajan, "An Authorization Architecture for Web Services", 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, pp. 222-236, 2005
[14] D. Ferrariolo, J. Barkley, and D. Kuhn, "A Role-Based Access Control Model and Reference Implementation within a Corporate Intranet," ACM Trans. Information and System Security, vol. 2, no. 1, pp. 34-64, 1999.
[15] I. Foster, C. Kesselman, J. Nick and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", in Open Grid Service Infrastructure WG (GGF), June, 2002.
[16] J. Joshi, E. Bertino, U. Latif and A. Ghafoor, "A Generalized Temporal Role-Based Access Control Model", IEEE Transactions on Knowledge and Data Engineering, 17(1): 4-23, 2005
[17] K. Keahey and V. Welch, "Fine-Grain Authorization for Resource Management in the Grid Environment", in 3rd International Workshop on Grid Computing, Baltimore, USA, November 18, 2002, pp. 199–206.
[18] S.H. Kim, J. Kim, S.J. Hong and S. Kim, "Workflow-based Authorization Service in Grid", in 4th International Workshop on Grid Computing, Phoenix, USA, November 17, 2003, pp. 94–100.
[19] S. Manolache, "Schedulability Analysis of Real-Time Systems with Stochastic Task Execution Times", Ph.D Thesis, Department of Computer and Information Science, IDA, Linkoping University
[20] K. Tan, J. Crampton and C. Gunter, "The consistency of task-based authorization constraints in workflow systems", In Proceedings of 17th IEEE Computer Security Foundations Workshop, pp. 155–169, 2004
[21] J. Wainer, P. Barthelmess, A. Kumar, "W-RBAC – A workflow security model incorporating controlled overriding of constraints", Intl. Journal of Cooperative Information Systems, 12(4), 455–486, 2003
[22] T. Ziebermayr and S. Probst. "Web Service Authorization Framework", The 2004 International Conference on Web Services, pp.614.
[24] W. Zuberek, "Timed petri nets in modeling and analysis of cluster tools," IEEE Trans. on Robotics and Automation, 17, pp.562–575, 2001
[25] K. Jensen, L. Kristensen, L. Wells, "Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems", Intl. Journal on Software Tools for Technology Transfer, 9(3), pp.213-254, 2007
[26] M. Owen and J. Raj, "BPMN and Business Process Management", http://www.bpmn.org/Documents/6AD5D16960.BPMN_and_BPM.pdf
[27] D. Schall, S. Dustdar and M. Blake, "Programming Human and Software-based Web Services", IEEE Computer, 43(7), pp.82-85, 2010.
[28]http://msdn.microsoft.com/en-us/library/bb330937.asp