Human Resource Constraints driven Virtual Workflow Specification

Roger Atsa Etoundi
University of Yaounde I
Department of Computer Science
PO.Box 812 Yaounde
ratsa@uycdc.uninet.cm

Marcel Fouda Ndjodo
University of Yaounde I
Department of Computer Science
PO.Box 812 Yaounde
marcel.fouda@uycdc.uninet.cm

Abstract

A virtual workflow specification is a formal description of business process in the real world and whose execution is done in a virtual organisation. Workflows are activities that involve the coordinated execution of multy tasks performed by different processing entities. Among the processing entities are human resources. These latter belong to an organisation or in a virtual organisation and are bound by strict constraints. Defining a correct execution of a workflow is very complex and error-prone, specially when dealing with human resources. Only a few works have addressed this problem and proposed some approaches for the verification of the correct execution of a workflow. These works do not tackel, in our known, the human resource issue. In this paper, we first identify the resource constraints in a workflow spacification. Then we prove an innovative approach to the checking of the human resource consistency in the virtual workflow specification. The formal description is done in the Raise Specification Language (RSL). The work reported in this paper provides a theoritical foundation for virtual workflow modeling and analysis in virtual workflow management.

Keywords: *Workflow execution, Workflow correctness, Human resource constraints, Virtual enterprise*

1. Introduction

Workflow management has emerged as an important technology to support the modeling, redesign and execution of business process [13]. According to workflow management coalition [6], *workflow* is defined as the computerised facilitation or automation of a business process, in whole or part. A *workflow management system* [5] is a system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic [6]. Since the execution of workflows is not always carried out by softwares, instate of using the world software in the definition of workflow management system, we use the term *agent* which denotes an entity capable in executing workflows.

In order to support their executions, business processes should be abstracted from the real world and then specified using the workflow specification language. The result is the definition of the workflow specification which contains information formally describing various abstractions of a workflow such as process, information and organisational aspect [2]. Building workflow specifications is very complex as stated in [13], and specially for the virtual one. For example lack of synchronisation among organisations in the use of human resources which introduce complex constraints such as mobility and availability.

Defining a correct execution of a workflow is very complex and error-prone, specially when dealing with human resources. Only a few works have addressed this problem and proposed some approaches for the verification of the correct execution of a workflow. These works do not tackel, in our known, the human resource issue.

The definition of a correct execution of a workflow remains problematic. Works that are carried out in this area are limited to time constraints. Resources aspects are neglected. Features - such as mobility, availability, qualification - associated to resources are not sufficiently used. In reality, the increasing number of demands of services or goods requires enterprises to manage human resources in order to satisfy the requirements of customers. These requirements continue to grow the need to programmatically validate the outgoing products or services attributes. Qualifies resources are therefore needed in order to deal with this challenge. The problem occurs when an organisation does not have qualified resources to deal with jobs, resources from another corporations are

solicited. This task is complex as the same resource may be solicitated by other corporations. To tackle this problem, we restrict ourself to enterprise human resources. We propose an approach to specify a workflow execution in a virtual enterprise based on the human resource constraints. The methodology used is based on the identification of the human resource constraints in a workflow spacification and the checking of the human resource consistency in the virtual workflow specification.

To handle rigorously customer's demands, enterprises cooperate by sharing their resources. In this respect, they are using internet to penetrate new markets, sharing supply chains in order to meet the challenge of the global market [17]. This integration has come out with new terms such as *virtual business process* and *Virtual Enterprise* concepts. A virtual business process of a virtual enterprise, also known as inter-organizational workflow [2], goes beyond a single corporation boundary: it is constructed by combining the services different enterprises provide. To share resources, communication among enterprises is fundamental, but it is preeminent in virtual organizations. Without communication, the boundary-spanning among virtual entities would not be possible. Given the burgeoning interest in this emerging phenomenon, it is surprising that very little empirical research exists on virtual organizations. Specially lacking are studies of communication processes within virtual organization settings. We extend our work with the design of an abstract protocol that enhances the communication among enterprises for better resource sharing.

The remainder of this paper is organised as follows. The next section introduces some settings on workflow specification. Section 3 discusses human resource constraints. After that, we address the analysis of human resources in the workflow specification in section 4, and define an abtract communication protocol to enhance the sharing of resources. Section 5 draws concluding remark.

2 The settings

2.1 Task

In the literature the notion of task is confusing. Still now, the defintion of task is not ontologically and formally clear [3]. In this work, we denote by *Task* an atomic activity which can be performed by a single resource in one step [20]. The concept of task is not quite new in the modeling of workflows. All the workflow modeling approaches consider this concept [4]. In most cases, the task concept is defined as a state transition function, but does not specify what is really changed in the state. Our approach to model task is guided by the works of *Allen* on time temporal logic in [1], and *Robert Goldblatt* on logic of time and computation in [10].

type
Task = State → State

By paraphrasing Leslie Lamport in [12], a *state* is an assignment of values to variables. One define variable as an observer that can be found in the execution environment of a task. Observers will be assigned values in order to reflect a concrete environment of execution of tasks. The observer concept can also be compared to variable defined in Lamport's Temporal Logic of Action [12]. From an observer *obv*, we define an assumption *propertz(obv)* of type *Condition* which may be satisfied by the state. A condition is described as a function from state to the boolean type *Bool*.

type
Condition = State → **Bool**
value
property : Observer → Condition

Given a task t and a state s the action $action(t, s)$ of t in s is the set of conditions CS, i.e $action(t, s) = CS$, whose values change by the execution of t in s. One can therefore by classified into orthogonal, concurrent, overlapping tasks [16].

2.2 Business process

We model a business process by the abstract type BP. Among its features, there is a set of tasks (*tasks(bp)*) to be executed, the environment (*env(bp)*) in which the execution will be carried out, the initial state from which the execution of tasks should start ($init(bp)$), the goals ($goal(bp)$) to be achieved, the place where each task t will be executed $place(bp)(t)$. Each business process is associated with a value $id(bp)$ which identifies it uniquely.

Some tasks need to be executed by the same agent, by different agents, or do not have any requirement about their execution. The features defined until now do not tackle this issue.

Definition 2.1 (Separation of Duties) *Let BP denote a business process, the separation of duties in BP defines a constraint among tasks of BP expressing the fact that some tasks of BP require different resources for their executions.*

This concept must be considered in the design of a business process in order to support its execution as it allows to check if there is enough resources to perform the activity to meet the deadline [7, 8]. To deal with the separation of duties according to an activity A, we define two relations between tasks. The first relation *separate* defines for each task *t* the set of tasks $separate(A)(t)$ that should not have

the same performance as *t* . This relation is required to be symmetric, non-reflexive as we do not allow any cyclic definition within the relations *succ* and *follow*. We also require that each task within the activity be associated with a set of tasks with which it can not share the same performance. In the same manner, some tasks will require the same resource for their execution like in [9].

Definition 2.2 (Execution sharing) *Given two tasks t and t' of an activity A, we say that t and t' have their execution shared if the two tasks require the same resource for their execution.*

If *t* is a task of A, $common(A)(t)$ defines the set of tasks whose executions require the same resource. This relation $common$ is required to be symmetric and transitive, and we require that all tasks be associated to a set of tasks with which they share the same performance.

In the definition of an activity, some tasks will require being executed only one at a time, others on the contrary will require being executed many times. Therefore, the models defined until now do not give information on the number of times a given task within an activity will be executed.

Definition 2.3 (Task's repetition) *Let t be a task of an activity a, the repetition of t defines the maximum number of times t may be executed.*

For a task *t* defined in the activity *a*, *repeat(a)(t)* denotes the number of times *t* should be executed. We also require that each successor and each direct successor of *t* be associated with a quorum, that is $succ(a)(t) \subseteq dom\ \ quorium(repeat(a)(t))$ and $follow(a)(t) \subseteq dom\ \ quorium(repeat(a)(t))$.

The execution of tasks does not only depend on the relation between them but also on some external events such as the availability of resources needed to be consumed during their execution. Therefore, defining the different relationships between tasks defined within an activity does not give all the information about the task execution.

There is an important difference between conditional execution and alternative execution which denote the execution depending on the execution of the workflow, and the execution depending on the environment and the execution of the workflow [8]. Therefore, we define, two categories of conditions: partial and execution conditions.

Definition 2.4 (Partial Condition of a task) *The partial condition of a task is a condition whose value only depends on the execution of its direct precedents.*

Let A denote a business process and *t* a task of $tasks(A)$, the partial conditions are defined for every successor and follower of t that is $dom\ \ partial\_constraint(a)(t) = succ(a)(t) \cup follow(a)(t)$.

Definition 2.5 (Triggering Constraints) *The triggering constraints define the conditions in which tasks in the business process may be executed.*

The execution of a task, except a minimum task, depends on the execution of its precedents. The triggering conditions are built from partial constraints using the boolean constructors. If t is a task of an activity a, that is $t \in tasks(a)$, and s a state of a such that t is triggered in s i.e $val(execution\_constraint(a)(t), s)$ and t' is a follower of t then t' is triggered in $t(s)$ i.e $val(execution\_constraint(a)(t'), t(s))$.

2.3 Workflow model

Organizations will have to come to deliver products and services in the global marketplace within time and resource constraints [2]. The trends for e-commerce along with increased global networking economies are real and are accelerated. We believe that the way organizations tackle this issue will define the difference between them. A number of concepts must be taken into account if organizations are to efficiently deal with this new challenge. The management of all this information depends in part on the enterprise workflow specification.

A workflow is represented by two main concepts: the process description which is composed of work items, the ordering between them, the conditions under which work items are executed, the set of agents involved in executing these works items, the starting time for its performance and the deadline for delivering goods or services. Formally a workflow wf is defined by a business process to be run i.e $bp(wf)$, and the time constaints associated to this performance, that is the starting time $startime(wf)$ and the completion time i.e $deadline(wf)$. Since these two last values may not be defined for all workflows, they are values of the variant time concept $TimePlus$ which better handles this situation. We denote by the abstract type $WFID$ the identity of a workflow.

The concept of *Timeplus* is specified as follows $TimePlus == undt|time(construct : Time)$ where the value *undt* expresses the undefined time, and values of the form $time(t)$, where *t* is obtained from the time generator *construct* which is a partial function defined from *Timeplus* to *Time*, is of type $TimePlus$. The generator *construct* is partial which means that it is defined for the values of *Timeplus* different from *und*.

2.4 Workflow Implementation

Once the business goal has been defined, it is neccessary to check how it can be fulfilled. This is defined by the execution path [15] also called procedure [21] which satisfies the associated business goal. A business goal may be associated to several execution paths. The designer of the business process must ensure that the associated business process will always be met if the associated resources are available. This may not be true in general. This is why in other approaches, the soundness property [19] has been defined. To deal with the soundness property, we should ensure that the business process can be executed. To meet this requirement, we first define what we understand by an implementation of a business goal.

Definition 2.6 (Implementation)

An implementation, the type $Implementation'$, is a list of couples composed each by a state S and a set of tasks TS such that their execution does not overlap in S that is $orthogonal(TS, S)$.

type
Implementation = (State × Task-**set**)*

When I notes an implementation, S and S' are the k^{th} and the $(k+1)^{th}$ state respectively, then S' is obtained from the action - *compute* - of the orthogonal tasks TS in S, i.e $S' = compute(TS, S)$.

Definition 2.7 (Execution) *An execution E, defines the resources dealing with the achievement of tasks, that is if t is a task then $exec(E)(t)$ defines the resource in charge of its execution, and the associated time intervals and $execperiod(E)(t)$ within which the achievements are performed in one shift.*

3 Human resource constraints specification

For a job to be executed in an enterprise, some human resources (agents) are needed. An agent is modeled as an entity that has the ability (skill) to perform work items. This consideration is based on the achievement of work items which itself is based on the notion of skill and delegation. In the modeling of agents, we will be consedering other aspects such as mobility [18]. The agent concept allows to deal with the organization perspective of a workflow design [17].

The concept of skill [11, 14] is one of the main concepts in enterprise business process modeling and human resource management. To ensure the performance of jobs requiring human resource in the enterprise, the manager should check not only the available of employees but also those who have the skill to perform them. Thus, each skill s should is associated to a none empty set of tasks $tasks(s)$. One defines the relation among skills.

Definition 3.1 (Heritage) *The heritage defines the relation among skills. If s and s' are two skills, such that s' herites from s then the set of tasks associated to s is contained in the associated set of tasks of s' i.e $tasks(s') \subseteq tasks(s)$, we then write $herite(s, s')$.*

One requires each agent ag to be associated with a non empty set of skills $dom\ \ skills(ag)$. Moreover, we require each skill $s \in dom\ \ skills(ag)$ to be associated with the set of tasks that ag can execute i.e $tasks(s) \subseteq skills(ag)(s)$.

In some enterprises, like in [9], experienced employees are required to deal with work items. In some other enterprises, managers are embarassed to choose an employee to perform a job when dealing with many employees having the same skill. One of the solution is to choose at random an agent. But this does not always work as the less experienced employee will take more time to perform the task [22]. One way to tackle this problem is to consider the experience of the employees in the workflow specification.

Definition 3.2 (Experience) *Let ag and ag' be two agents and s denotes their common skill, we say that ag is more experimented than ag' regarding s, and we write $experience(ag, ag', s)$ if $skills(ag')(s) \subset skills(ag)(s)$.*

To deal with the management of employees in the enterprise, the manager must be able to define the availability of employees at any moment, but also to deal with the unavailability of some employees due to sickness, holidays or resting which determine the stochastic and determinictic events [4]. To deal with this event, one needs to keep track of the schedules for each employee.

Definition 3.3 (Schedule) *A schedule E is characterised by time interval $period(E)$ and the purpose purpose(E) for which the time is used.*

The purpose can be a job to execute or an event - the type (*Onlive*) - that may make an agent to be off from the organisation.

Definition 3.4 (Diary) *A diary - the type $diary$ - defines an ordered sequence of schedules for a given agent.*

This sequence is ordered occording to the time intervals associated to schedules, and these slots of time do not overlap as an agent may not deal with more than one purpose at the same time.

An agent may move from one place to another. The specification of a workflow should consider the mobility as an agent can get to place from which the execution of a job becomes impossible. The system must the be able to detect this failure and look for another agent that can deal with the underlined job or inform the manager.

4 constraints consistency for virtual workflow specification

As mentioned earlier in this paper, resource conflicts are caused by inadequate representation between resources constraints and jobs defined in the workflow execution. These constraints are required to be identified and removed from the workflow specification.

Definition 4.1 *Within each shift in the workflow execution R, the execution is said to be valid, noted by $valid(R)$, if all the resources involved in the execution of tasks have the required skills for their achievement.*

When resources have been defined for the achievement of tasks within a workflow, it is not clear whether they are idle or not. Based on this limitation, one defines the *feasibility* of a workflow.

Definition 4.2 (Feasibility of a workflow) *A workflow wf is said to be feasible within an enterprise Org if all the resources involved in each run rn of wf are available i.e $allfree(org, rn)$.*

In general, workflow will not always be feasible as some resources involved in the execution may be occupied during the period of achievement due to sickness or events that prevent him to carry out the execution. One then defines the *relaxed feasibility* of a workflow execution.

Definition 4.3 (Relaxed feasibility) *Let Org be an enterprise, wf a workflow required to be carried out in Org, then wf is relaxed feasible if wf all the resources involved in the execution of tasks of an existing run are available, and belong to Org.*

Among events that can prevent resources to execute the assigned jobs, are their mobility. During the execution of a workflow, one should be able to define the location of each resource and compute the duration that made to move from one place to another. We assume that each resource when moving has a device that can communicate information to the organisation. However, during his mobility, he can be connected or disconnected from the organisation [23].

Given an enterprise Org dealing with roving agent MAg, and a workflow wf, the mobile agents involved in the achievement of of *wf* have the following properties:

- all the mobile agents of Org involved in the execution of wf are all connected i.e $connect(MAg, Org)$,
- All the mobile agents involved in the achievement of wf have enough time to potentially perform the jobs assigned to them in Org i.e $ontime(MAg, Org)$.

Definition 4.4 (Required Workflow) *A workflow wf is said to be required $required(Org, wf)$ within an enterprise Org dealing with the mobility of agents if all the mobile agents are connected, and have enough time to move from their current place to places they are required to perform jobs, that is $connect(MAg, Org)$ and $ontime(MAg, Org)$.*

The workflow will not always remains is this state as some mobile agents may be discinnected from the organisation. At this time, one cannot guarantee the completion of wf.

Definition 4.5 (Potential Workflow) *A workflow wf is said to be potential i.e $potential(Org, wf)$ within an enterprise Org scoping with the mobility of agents, if some mobile agents involved in the execution of wf are not reachable i.e $connected(Org, wf) = false$.*

Definition 4.6 (Unachievable Workflow) *A workflow wf is unachievable in an enterprise Org if some mobile agents involved in its execution do not have enough time to reach places where they are required to perform jobs i.e $connected(Org, wf)$ and $ontime(Org, wf) = false$.*

Based on what has been defined so far, one can notice that all the goals of the within an organisation cannot sometimes be met by considering agents of a single enterprise. Therefore, enterprises handle business goals rigourisly by communicating among them in order to share their resources (agents) [2]. The verification of the correctness of the execution in the resulting enterprise noted by a virtual enterprise is very complex as it may concern not only the assignment of jobs to agents but also the consistency of agent schedules. One must avoid that an agent be assigned more that one job within a time interval.

Definition 4.7 *Given a workflow wf to be executed within a virtual ve enterprise, we say that wf is correct if the resources involved in the achievement of the associated tasks have the skills to perform them, are idle within the execution period, and if the agent is not in the place where the execution will take place then there should be enough time to reach the place on time.*

Let Ag be an agent assigned a task T within a period P, $tcost(ve, place(ve, Ag), place(T))$ the duration to move from the place $place(ve, Ag)$, defining the current location

of Ag, to $place(T)$ the place where the execution should be carried out. The travel cost includes the duration jobs to be executed if existed before the performance of the ultimate job.

If Tm denotes the time during which Ag is in the place $place(ve, Ag)$ then Ag is on time to perform T means that $Tm + tcost(ve, place(ve, Ag), place(T)) \leq start(P)$.

The routing of tasks to resources in a virtual enterprise is very complex as most of them can be involved in more that one workflow execution at the same time. For correctness purpose, we extend this work by defining an abstract protocol dealing with the assignment of jobs to agents in a virtual environment.

5 Abstact communication protocol

When an enterprise fails to satisfy the goal of a given task for lack of agent, it sends a request to the enterprises forming the virtual organisation to get if possible an agent that can deal with this task based on its time constraints. Each enterprise reacts by sending the failure to get the required agent or information of the agent that can execute the job. Each enterprise provides a repository $inbox(Org)$ (a data base) to keep track of the requests sent by enterprises that need share the execution of their workflows.

The modelling of the framework of these databases and requests can be defined as follows, where the function $snd\_request$ denotes the primitive used to send a request. One requires that for a request rg to be sent by an enterprise org, that the goal of the associated task - $task(rq)$ - can not be met in org, this requirement is denoted by $unachievable(task(rq), org)$ if the following Raise's code.

The requests received are treated in order to get the required resource. The request is met if a resource has been defined for the achievement of the task, and there is enough time to travel, if required, in order to execute the task. In this case the enterprise from which the resource belongs, and the period of the execution of the task are returned. those concepts are captured in the abstract type $Assign$. In the case the request cannot be met, an undefined response und is returned. One defines a type *RP* to consider the two cases.

A request rq is satisfied by an enterprise org if one can define an instance of the type $Assign$. Therefore, an instance asg is defined for a request rq if a qualified resource rs has been found - that is *qualified(rs,task(rq),org)*-, can reach the place to execute the job ontime - that is if p and p' define the location of rs and the place to execute $task(rq)$ then $tcost(ve, p, p') \leq start(period(asg))$ - and rs is idle within the period $period(asg)$ that is *free(rs,period(asg),org)*.

In the case the request is satisfied, a notification is sent to its sender and to the enterprise from which the resource has been found. Thus, the system supporting the communication in the virtual enterprise must provide software components for the notification to the sender of the request and to the enterprise owner of the resource.

The modelling of these software components is given by the following functions *notify* and $notify\_owner$ descibed below. The definition of these function uses the term $share(org)$ to keep track on resources involved in the planning process of external workflows, while $get\_rp(ve, rq)$ denotes the identification of the resource chosen for the execution of the task $task(rq)$ associated to the demand rq.

Since different workflows can be planned concurrently, and a resource can be assigned tasks such that it becomes overloaded, the virtual enterprise manager must be able to detect it and find another resource which can deal with the execution of the resulting task of the request.

Requirement 5.1

The system dealing with the management of requests in the virtual enterprise level must a way of checking if a resource to which a task has been assigned within a time interval is also involved in the execution of another task whose period overlaps with the current period. In this case, there should be a way to determine, if possible, another resource for the achievement of the task.

Definition 5.1 (Reserveed Resource) *Given a resource rs in a virtual enterprise ve, and a request rq, rs said to be reserved for a the performance of $task(rs)$ if the period p within which rs is required to perform $task(rq)$ overlaps with a a period p' within which another task has been planned for execution.*

Definition 5.2 (Performable Task) *Let ve and rq denote a virtaul enterprise within which the resource chosen to perform the task of the request rq is reserved, we said that $task(rq)$ is perormable is another resource can be found for its execution.*

When the demand has been handled by the manager of the virtual enterprise the result of the request is sent to sender of the request and to the enterprise from which the resource, if any, has been found for the execution of the associated task. This is done by the the software component $notify$ described previously.

When the enterprise has received the response from its request of resource, if the required resource has been found, it is inserted in the run of the resulting workflow, else the goal pursued by the workflow cannot be achieved. If there is

no resource available to deal with the execution of a task in the virtual enterprise, the workflow should be rejected from the enterprise as the customer goal cannot be achieved, and its partial plan of execution made should be cancelled.

Property 5.1

When the partial plan of the execution of a workflow is cancelled, the resources used in this plan should be released.

Requirement 5.2

The decision support system for decision making in the virtual enterprise must provide a software component in charge of discharging resources from the execution of tasks whose periods overlap with the time intervals within which the execution of a task is confirmed.

In the case the execution of a workflow is satisfied within a virtual enterprise, the diaries of the external resources involved in its achievement are updated in their enterprises and in the virtual enterprise.

6 Conclusions and future work

Before a workflow specification is deployed and put into operation, it should be verified for its correctness. However, workflow specification contains information for representing various aspects of a workflow, which makes the workflow verification being too complex. Among them are information related to human resource. The previous work on workflow correctness and verification was based on its structure and did not tackel the human resource constraints.

This paper discusses tha analysis on the human resource resource constraint of a workflow specification and defines different abstractions of workflow correctness, and human resource constraints. One first defines the correctness of a workflow within an enterprise. Secondly, this correctness has been extended to the mobility constraints of the human resources, and finally by their sharing in a virtual organisation. This paper does not tackel the management of the loads among human resource, as some person may be overloaded than others. So, a future work is to distribute load among resources in such a way that an overloaded agent cannot be found.

References

[1] J. F. Allen and G. Ferguson. Actions and Events in Interval Temporal Logic. *Journal of logic and computation*, 4(5):531–579, 1994.

[2] C. Bussler. Enterprise-Wide Workflow Management. *IEEE Coucurrency*, page 32, July-September 1999.

[3] C. Castelfranchi and R. Falcone. Towards a theory of delegation for agent based-systems. *Robotic and autonomous systems*, 24:141 157, 1998.

[4] J. Dehnert, J. Freiheit, and A. Zimmermann. Modelling and evaluation of time aspects in business processes. *Journal of the Operational Research Society*, 53:1038–1047, 2002.

[5] A. Dogac, K. Leonid, M. T. Ozsu, and A. P. Sheth. Workflow Management Systems and Interoperability. *NATO Asi Series Series F, Computer and Systems Sciences,Springer Verlag*, 164), October 1998.

[6] P. L. ed. *WFMC Workflow Handbook*. John Wiley and Son, New York, 1997.

[7] J. Eder and M. Rabinovich. Time Constraints in Workflow Systems. *Lecture Notes in Computer Science*, 1999.

[8] J. Eder, M. Rabinovich, H. Pozewaunig, and E. Panagos. Time Management in Workflow Systems. *Lecture Notes in Computer Science*, 1999.

[9] E. Egger and I. Wagner. TIME MANAGEMENT: A case for CSCW. In *CSCW'92*, pages 249 – 25. ACM, November 1999.

[10] R. Goldblatt. *Logics of Time and Computation*. Number 7 in CSLI Lecture notes. CSLI - Center for study of language and information, Stanford University, 1986.

[11] C. Jeff Davidson, MBA. *Managing Your Time*. alpha books, 1995.

[12] L. Lamport. TLA in Pictures. Technical report, System Research Center - SRC, 1994.

[13] H. Li, Y. Yang, and T. Y. Chen. Resource constraints analysis of worflow specification. *The journal of Systems and Software*, (73):271–285, 2004.

[14] A. Mackenzie. *The Time Trap*. AMACOM, 1997.

[15] M. R-Moreno, P. Kearney, and D. Meziat. A case study: Using Workflow and AI Planners.

[16] A. E. Roger and M. F. Ndjodo. A Generic Abstract Model for Business Processes and Workflows Management. In B. Gerald and K. Thomas, editors, *4th International Workshop on Mobile Computing*, pages 62–72. IRB Verlag, Stuttgart Germany, 2003.

[17] A. P. Sheth, W. van der Aalst, and I. B. Arpinar. Processes driving the networked economy. *IEEE Concurrency*, page 18, July-September 1999.

[18] C. P. Tom Rye. The paertnership aproach to mobility management: an evaluation of different models for travel planning by group of employees. In *ECOMM 2001* .

[19] W. van der Aalst and A. ter Hofstede. Verification of workflow task structures: A petri-net-based approach. *Information Systems*, 25(1):43–69, 2000.

[20] W. van der Aalst and K. van Hee. Business Process Redesign: A Petri-net-based approach. *Computer in Industry*, 29(1-2):15–26, 1996.

[21] W. van der Aalst, K. van Hee, and G. Houben. Modelling and analysing workflow using a petri-net based approach. In *Second workshop on CSCW*, pages 31–50, 1994.

[22] D. D. E. Wetmore. The Blocks to Employees Productivity. http://www.balancetime.com, August 23 1999.

[23] E. Younker. Management Update:The new need for better Mobile Technology Management by Enterprises. *Gartner Review of the Telecommunications Industry*, pages 9–11, august 2001.