Feature-Oriented Business Process and Workflow

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

One of the most important problems that face developing countries is the lack of qualified resources in all the domains of the activities. This is trivially true in the main critical sectors such as manufacturing, health and education areas. Managers in these sectors deal with the problem of efficient management of qualified resources in their possession. Therefore a solid understanding of their activities and their planning for their execution based on the available resources are vital if they wish to meet goals driven by customers. The objective of this paper is to define a generic framework that formally captures the salient features required for efficient planning of human resources within an enterprise, and to give fundammental requirements of a correct execution of an activity based on the resource constraints.

Keywords: *Office Automation, Design Methodology, Requirement Specification, Formal Modeling, Enterprise Human Resource Planning, Business process modeling, Workflow Correctness*

1. Introduction

In the area of information systems, workflow modeling has attracted a lot of researchers. Most of the works [33, 6, 13] on this domain are focused on business proccesses, and do not take into consideration the concrete execution of their resulting workflows [12, 14]. This failure to reason about the execution of a workflow put beside the time and resource concepts. The absence of these two concepts do not allow the management of resources. The reason about this absence is due to the fact that most works carried out in the area of business process and workflow have been done in developed countries. These latter do not suffer of the lack of qualified resources [5], but they are putting in place new processes or re-engineering old one in order to satisfy their customer goals. In developing countries, the problem is different: there is a lack of qualified resources and business processes used here are designed in developed countries. As consequence, business processes are required to be understood, and are required to be implemented based on the available limited qualified resources. The problem that occurs in the implementation of business processes is that resources are most of the time involved in the achievement of work items from different business goals, and even in different organisations. This involvement of resources makes their management very hard to be achieved. In developing countries, the main focus is not only the understanding of the modeling of business processes but also the reasoning about their execution. Reasoning about the execution of a business process allows to deal with its correct execution. The correctness, in most works, is syntactical that is the absence of deadlock and livelock [18].

1.1 Correctness Problem

When a business process is syntactically correct,it does not mean that executions of resulting workflows will always be correct. This is due to the fact that other aspects must be taken into consideration. Those concepts include the time and the resource. Therefore to deal with the correct execution of business goal requires reasoning about the process, the time and the resource having the ability to deal with work items [14]. Thus, the correctness of a workflow is based on the resource constraints such as their availabilities and their abilities. Regarding resource constraints, managers have to avoid resources to be overloaded that is assigned two or more work items whose executions is done at the same time, or to delegate the performance of a work item to a resource that does have the skill to handle it [23, 19].

In the understanding of the business process, managers

should be able to define the ordering among work items, specifically they should be able to express when the execution of some tasks overlaps. To deal with the overlapping of work items, each manager is required to know the action of each work item in its execution's environment. Mostly, customers fix some time constraints regarding the delivery of goods or services. Even business goals require in some cases time constraints for their achievement. Each manager must therefore know the execution time of each work item [23, 19]. Combining all these constraints for the achievement of customer's goals makes the management of business process a very complex task.

To deal with the availability and the ability of resources, each manager must keep track of the schedules and the abilities of each resource [19]. He should also have a way to reason about them. For this end, he may then be able to accept and to plan, or to reject the execution of a business goal. In the case where the execution of a business goal is performed, he can inform the customer involved on the date of the delivery or the date when the execution may start.

In this work, we treate this problem as an engineering one. Based on the domain engineering [7], we first present a formal framework for business process and workflow which take into account not only the description of the business process and workflow but also the modeling of resources. Secondly, we give a fundamental properties of what we mean by a correct execution of a workflow based on the resource constraints. We will not show in this paper how the planning of the execution is carried out. The framework is described as a set of models defined in an incremental manner. Each model is formalised using the Raise Specification language [17].

The rest of the paper is organized as follows. Section 2 presents the salient features in terms of models of business process, workflow and resource. Section 3 defines what it means for an execution of a workflow to be correct, by relating the workflow models and the resource models. Section 4 provides some conclusions.

2 Model-based business process

2.1 Environment

The performance of work items is based on some properties under assumptions whose values are guaranteed by the environment. An environment satisfies some properties [1] that are needed to be checked before the execution of any work items. The term environment has various meanings in the literature. For example, in system design [21], it means a set of variables defined in the system. In the performance of work items, the term environment describes different metrics considered for the achievement of work items within an organization. The design of workflow without taking into consideration the environment concept will be incomplete. This concept may contribute in the operational and information perspective of the workflow design.

The information system of an enterprise changes within time. This change is driven by the execution of work items, the definition of new business goals, or the dynamic behavior of resources. The set of different metrics whose value may change is denoted by Environment. Without any loss of generality, we consider that an environment, denoted by *Env*, is a set of primitive observers (*Observer*). The observer concept is seen as variables defined in Lamport's Temporal Logic of Action [22]. An observer describes a specific metric considered in the environment.

Observers will be assigned values in order to reflect a concrete environment of execution of a business process. From the set of observers and their associated value we define the type *State*. Paraphrasing Leslie Lamport in [22], a *state* is an assignment of values to variables. We then use the same semantic of state defined by Leslie Lamport.

From the observers, we define assumptions which may be satisfied by the environment. We use the concept *Condition* to denote an assumption of the environment. A condition is described as a function from state to the boolean type. Conditions, denoted by the type *Condition*, may be composed using the boolean constructors: *alternative,imply,compose*, or *opposite*. The value of a condition c within a state s is denoted by the boolean term $val(c, s)$. Given two conditions c and c' and a state s, we have the following properties:

$val(alternative(c, c'), s) = val(c, s) \vee val(c', s)$,
$val(imply(c, c'), s) = val(c, s) \Rightarrow val(c', s)$

$val(compose(c, c'), s) = val(c, s) \wedge val(c', s)$,
$val(opposite(c), s) = \neg val(c, s)$

The concept of condition, inspired from the works of *Allen* on actions and events in time temporal logic [4], and the works od Lamport in TLA [22, 2], will allow in specifying the workflow management system behavior in terms of logical formula, including temporal constraints, events and the relation between the two.

Definition 2.1 (Satisfaction of a condition)

Given a state s and a condition c, we say that c is satisfied in s if the value of c in s is $true$ i.e $val(c, s) = true$.

Definition 2.2 (Characteristic of a state)

Given a state s, the characteristic of s denoted by

$charact(s)$ is the set of all conditions satisfied in s.

We are not interested on all type of state, but on states within which some conditions may be satisfied. Therefore, having defined what we mean by the characteristic of a state, we can now introduce properties which restrict the concept of state.

Definition 2.3 (Useful State)

Let s denoted a state, s is useful, i.e $useful(s)$, if its characteristic is not empty.

Definition 2.4 (Equivalent of states)

Two states s and s' are equivalent, denoted by the boolean term $equiv(s, s')$, if $charact(s) = charact(s')$.

Definition 2.5 (Gap between states)

Given two states s and s', the gap between s and s' is the set of conditions whose values are not the same in s and s'. That is $gap(s, s') = \{c | c : Condition \bullet val(c, s) \neq val(c, s')\}$

Definition 2.6 (Sub-state)

Given two states s and s' be two states, we say that s' is a sub state of s and write $is\_in(s', s)$, if s satisfies any condition satisfied by s', i.e $charact(s') \subset charact(s)$. This relation is required to be a partial order, readers can have a look on this concept in [8].

We need to define for non empty set of states, the minimum and the maximun state. To meet this goal, we have to associate to the state concept the structure of complete lattice. Therefore, before defining these two concepts, we first introduce this notion.

2.1.1 State Ordering

Let L denotes an abstract type, and PL a set of L i.e $PL = L - set$. PL is associated with the partial order $\leq$, PL is a lattice if $(PL, \leq)$ is a complete partial order set such that to every non empty subset $SBPL$ of PL are associated a greatest lower and a least upper bound denoted respectively by $sup(pl)$ and $inf()$ where pl is of type PL. More information in this notion can be found in [8].

Let $(PL, \leq)$ be a lattice and T_S a set of states, We define a bijection between PL and T_S in order to transfer the structure PL to T_S.

Therefore, given two different states s and s' if $s \leq s'$ it means that $refine(s) \leq refine(s')$. Moreover, if s is a least upper bound of T_S then $refine(s)$ is a least upper bound in PL.

2.2 Task

The satisfaction of a customer goal is driven by the execution of work items (*Task*). A task concept is guided by the works of *Allen* on time temporal logic in [4], and *Robert Goldblatt* on logic of time and computation in [16].

2.2.1 Modeling action

Given a task t and a state s, the execution of t in s yields the state $compute(t, s)$ where $compute(t, s) \equiv t(s)$. Moreover, for each task t and a state s the action of t in s is the set of conditions $action(t, s)$ whose values change by the execution of t in s.

Formally, $action(t, s) \equiv gap(s, t(s)$

The actions of two different tasks t and t' within a state can overlap that is $action(t, s) \cap action(t', s) \neq \emptyset$.

Actions of tasks can be orthogonal within some states and not orthogonal in anothers. Therefore, tasks may be weakly or strongly different according to their actions.

Definition 2.7 (Weak task Different)

Tasks ts are weakly different, and we write $w\_diff\_task(ts)$ if there exists a state s within which they overlap, that is for all task t and t' of ts we have $overlap(t, t', s) = true$.

2.2.2 Strong Task Different

Tasks ts are strongly different, and we write $s\_diff\_task(ts)$, if they never overlap in any state s i.e for all state s and tasks t and t' in ts, $overlap(t, t', s) = false$.

In the rest of this paper, we use the strong tasks difference in order to model tasks in the business process.

2.2.3 Simultaneous execution of set of orthogonal tasks

Let *ts* denotes a set of orthogonal tasks over the state *s*, then the execution of *ts* yields the state $compute(ts, s)$, which satisfies the relation $compute(ts, s) = sup(shift(ts, s))$. The term $shift(ts, s)$ denotes the set of states that are obtained after the simultaneous execution of ts.

2.3 Ordering

Tasks associated to an activity are performed in a certain order [35, 26]. Some tasks are required to be executed before others. The ordering defined between tasks is not a new concept, it was already defined in other approaches dealing with business process modeling, but does not explicitly capture the different facets of the ordering. We only give in the following a formal syntax of the ordering between tasks of a

business process. In [28, 29], we have defined the semantic of the ordering in terms of follower and successor.

2.3.1 Task Dependency

The dependency among tasks, captured by the type *DEP'* defined as $DEP = Task \quad \overrightarrow{m} \quad Task - set$, defines the relation among them.

The symbol $\overrightarrow{m}$ denotes a map definition, and the term $Task - set$ denotes a set of tasks. This dependency is no cyclic and does not introduce any ghost task.

2.4 Task repetition

To tackel the repetition of tasks, we require the design of each business process to be associated with the number of times each task may be executed. Therefore for eac task, are defined the maximum number of repetition $rep(rp(t))$, the number of time $quorium(rp(t))(t')$ its execution should be done in order to execute its follower or successor t'.

2.5 Assumption

If t and t' are two tasks of a business process such that t depends on t', a partial assumption of t' in relation to t defines an assumption $pt(t)(t')$ whose value is guaranteed by the execution of a task t. Moreover, each task is associated an assumption $trigger(t)$ which triggers the execution of t when it is satisfied. Lets denote by $env(t)$ the constraints that does not depend on the execution of any task. Within an organisation like a medecical center, $env(Xray)$ the availability of a radiologist to perform an X_Ray to a patient. If t is minimum task that is t does not have any predecessor then $trigger(t) \equiv env(t)$, otherwise $trigger(t) \equiv env(t) \wedge env(t)$.

2.6 Time

A time concept is defined by modalities. A modality defines a feature that characterises a concrete representation of a specific point expressing in year, month, day, minute, second and so fourth. The time concept can either be expressed in *seconds*, *minutes*,*hours*, *days*, and *months*, *years* or a combination of some of these concepts. Depending on its use, time can be expressed in different ways. For two time instances tm and tm' such that $tm < tm'$, is defined a time interval ti such that $start(ti) = tm$ and $dus(ti) = tm$.

For a given time interval ti, a duration of ti, specified by $duration(ti)$, denotes the amount of time between $start(ti)$ and $due(ti)$.

2.7 Mobility

Employees (resources) in most enterprises move from the place they are located to their enterprises each day [30]. Most often, due the limited number of resources, enterprises come to share their employees in order to meet their respective business goals. All these enterprises are not located in the same geographic area. Therefore, employees need to move from one geographic area to another. Often, employees face a lot of problems such as poor public transport access and congestion [32, 30]. Based on these difficulties for employees mobility, some business goals may not be accomplished within time constraints, this resulting in customer insatisfation.

2.7.1 Agent Location

To go against customers insatisfactions, the mobility of employees must be considered in the workflow modeling if enterprises are sharing their employees. To deal with the management of mobility, we need to define some key concepts are needed to be taking into consideration. These concepts include, the location of an agent. The location lt is defined by a place $place(lt)$ and the time $time(lt)$. For mobility management, each place pl is associated with a set of places $nbh(pl)$ which are directly accessible.

To move from one place to another will require a certain duration. Thus, let P and P' denote two neighbouring places, the traveling cost of P to P' denotes the time required to move from P to P'. This time is given by the $tc(p)(p')$. We only require that this duration be non nil i.e $tc(p)(p') > 0$. Moreover, to move from a place *p1* to a place *p3* through place *p2*, it may require waiting for sometime in place *p2* before leaving to *p3*. Therefore the time to move from *p1* to *p3* does not only depend on the time to move from *p1* to *p2* and from *p2* to *p3* but also on the waiting time in *p2*. Let P and P' denote two neighbouring places, the travel delay $dl(P)(P')$ from P to P' denotes the time during which the travel from P to P' is delayed.

Moving from one place to another may require passing through an itinerary which is a list of places. The itenerary (travel path) tr, defines the sequence of places such that for two consecutive indexes I and $I+1$, the $(I+1)^{th}$ place is a neighbour of the $(I)^{th}$ place. If tr defines a travel path, dl the delay among places, and tc the travel cost among places then $duration(tr, dl, tc)$ defines the duration of the travel tr.

2.8 Resource

For a job to be executed in an enterprise, some resources are needed. A resource can be either a person, a machine, a service or anything capable of performing tasks or to participate in the execution of a task. Most of the time, resources are mainly human. However, because the execution of work items does not exclusively depend on humans, we use the term $Agent$ to denote any kind of resource which participates in the achievement of a task. There are a lot of confusions between the of the notion of agent [10, 11]. Most definitions given depend on the point of view its autors wish to defend. In our approach, we take a unifying point of view, and consider an agent as an entity that has the ability (skill) to perform work items. This consideration is based on the achievement of work items which is based on the notion of skill and delegation. An important focus will then be made on these three notions.

The main problem posed by agents is the role they play in the achievement of goals defined in an organization. The term *agent* has a large number of meanings in the agent community. Some of these meanings may be found in [24, 25, 20]. In the organisation, we are focused on the achievement of work items, in this way we are interested in agents that are capable of performing work items. We will therefore be limited to this type of agents. In the modeling of agents, we will be considering other aspects such as mobility. The agent concept will allow us to deal with the organization perspective of a workflow design [31]. Based on the important role agents play in the organization, we argue that it will be unfair to model workflows without taking into account the agents involved in the achievement of the associated tasks.

2.8.1 Modeling Skill

Among resource qualifications, is the skill which defines the ability to use one's knowledge efficiently and ready in execution or perfomance. In the achievement of the goal of a business process, the concept of skill will play an important role in performance analysis and resource management in the enterprise. To ensure the performance of a work item in the enterprise, the manager should check not only the available employees but also those who have the skill to perform the work item. Each skill s should then be associated to a set of tasks $task(s)$. This set of tasks is required to be non empty that is $task(s) \neq \emptyset$. If s and *s'* be 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 write $herite(s, s')$.

Agents are managed efficiently according to their role in the organization [20]. The role played by an agent within an organisation may be large than the one initially assigned. We require each agent ag to be associated with a non empty set of skills $dom \quad skills(ag)$. Moreover, we require each skill s of $dom \quad skills(ag)$ to be associated with the set of tasks that ag can perform i.e $tasks(s) \subseteq skills(ag)(s)$. The definition of agents and their associated skills within an organization does not yield their various experiences. The experience of each agent is used to select a specific agent according to the requirements of the goals to be achieved.

2.8.2 Modeling Experience

According to the definition of skills associated to resources, it is interesting to note that resources with the same skills do not always have the same tasks to deal with work items. In some enterprises, like in [15], experienced employees are required to deal with work items. In some other enterprises, managers are embarassed to choose an employee to perform a work item when dealing with many employees having the same skill. This can be done at random. But this does not always work as the less experienced employee will take more time to perform the task [37], as a result if the work item was assigned a deadline to be accomplished, one may not ensure that the deadline will be satisfied. One way to tackle this problem is to consider the experience of the employees in the workflow design.

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 the set of tasks to be performed by *ag'* is included in the set of tasks to be performed by ag in s. This relation is required to be reflexive, and transitive. However, if two agents have the same experience this does not mean that they are equal but that their experiences within the skill are equivalent. The experience does not define the manner in which an agent deals with a specific task within a given skill. The adversity with which an agent can carry out a task t is defined by its ability to perform t.

The experience within a skill does not define the ability with which a resource may perform a given task of the skill [19, 23]. To deal with the concurrency based on the choice of a resource to execute a work item, the ability is one of the concepts to consider in order to deal with the performance analysis of a workflow. Given a resource ag and a task t, the ability of ag regarding t defines the capability of ag to perform t. We require for each task t that an agent is able to perform, that its ability be defined for its performance, this

ability is denoted by $ability(ag)(t)$. In our model, we leave open the way the ability will be expressed in concrete.

However, for efficient management of abilities, we require this concept to be associated with a partial order $\leq$. We also require this relation to be reflexive i.e for all ablility a $a \leq a$, symetric i.e for all a, b two abilities such that $a \leq b$ and $b \leq a$ then $a = b$, and transitive i.e for three given distinct abilities a, b and c such that $a \leq b$ and $b \leq c$ then $a \leq c$. Let *t* be a work item, and *ag* and *ag'* be two agents with the same skill to perform *t*. We say that *ag* is more qualified to deal with *t* than *ag'* if the ability of *ag* is greater than the ability of *ag'*.

2.9 Resources Schedules

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 sickness or holidays. The model of the workflow must allow to handle stochastic events and the load of each employee [12]. Keeping track of the stochastic and deterministic events must allow the focus of the workflow analysis [12] but also to deal with the overload problem faced by employees in most enterprises [23, 15, 19]. One way to tackle these problems is to keep track of all the loads and the time used by each employee.

A schedule E, defines an atomic activity $job(E)$ and a time interval $period(E)$ within which the execution of $job(E)$ should be done.From the entry concept, the diary concept D is defined as an ordered sequence of entries. The diary is required to satisfied the following properties:

- Entries of a diary should be sequentially ordered according to the periods of the achievement of the different jobs, and these time intervals should not ovelap. This means for two different indexes k and k' of the diary D such that $k < k'$ then $due(period(D(k))) < start(period(D(k')))$.

Requirement 2.1

The decision support for the management of a diary D for decision making must allow to check if an entry E belongs to D i.e $in\_in(E, D)$, if a time interval p is idle in D i.e $free(p, D)$, to define a time interval associated to the execution of E of D i.e $search(E, d)$ and to check if a schedule E can be inserted in d i.e $can\_be\_inserted(E, d)$, to insert a new entry in the diary without violating its property, to return a set of idle periods with same given duration.

3 Modeling Execution

Once the business process has been defined, it is neccessary to check weither it can fufill its resulting goal. This can be checked by defining if there can exist an execution path [27] also called procedure [36] which satisfies the associated business goal. A business process may be associated to several execution paths. These execution paths depend on the environment of its execution. When designing a business process, the environment within which it will be executed is unknown. Thus, 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 [34] 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 process.

Definition 3.1 (Implementation)

An implementation, the type $Implementation'$, is list of couples composed each by a state S and a set of tasks TS such that $orthogonal(TS, S)$. Formally, the implementation is captured by the type $Implementation = (State \times Task - set)^{\star}$.

Definition 3.2 (Implementation Impliementation)

Let I notes an implementation, I is said to be valid if S and S' are the k^{th} and the $(k+1)^{th}$ state respectively, then S' is obtained from the action of the orthogonal tasks TS in S, that is $S' = compute(TS, S)$.

3.1 Execution

An execution defines the identity of resources dealing with the achievement of tasks, and the associated time intervals within which each performance is performed in one shift, and the state within which each job is carried out.

We will not deal witth all types of execution, but the one i.e R which guarantees that for each task T of the shift $exec(R)$ the resulting resource $exec(R)(T)$ is free within the period $execperiod(R)(T)$ of execution of T this means that there is not task T' assigned to the resource identifies by $exec(R)(T)$ whose execution period overlaps with $execperiod(R)(T)$ i.e $free(execperiod(R)(T), diary(identifier(exec(R)(T))))$.

Definition 3.3 (Run)

A run Rn defines a well formed implementation $implementation(Rn)$, and a sequence $execution(Rn)$ of executions.

For an execution (run) to be correct, the following requirements are required to be satisfied.

Requirement 3.1

The execution periods associated to different shifts are required not to overlap. This is captured by the following axiom:

axiom
(∀ r:Rn, k : **Nat**, j, j′ : Task •
k ∈ **inds** execution(r) \ {**len** execution(r)} ⇒
let e = execution(r)(k), ex = execperiod(e),
e′ = execution(r)(k + 1), ex′ = execperiod(e′)
in j ∈ **dom** ex ∧ j′ ∈ **dom** ex′ ⇒
due(ex(j))≤ start(ex′(j′))**end**),

Requirement 3.2

Within a shift, each agent should be assigned only one task. This is captured by the following axiom:

axiom
(∀ k : **Nat**,r:Rn • k ∈ **inds** execution(r) ⇒
card dom exec(execution(r)(k)) =
card rng exec(execution(r)(k)))

Requirement 3.3

Tasks ts whose executions have been planned within a shift are orthogonal tasks and are the only tasks assigned to resources. The requirement is verified by the axiom that follows:

axiom
(∀ k : **Nat** •k ∈ **inds** implementation(r) ∧
k ∈ **inds** execution(r)⇒
dom exec(execution(r)(k)) =
dom execperiod(execution(r)(k)) ∧
case implementation(r)(k) **of** (s,ts) →
card ts = **card dom** exec(execution(r)(k)) **end**)

Requirement 3.4

Each agent ag assigned a task t for a given run r should have the skill to deal with the execution of t. That is exists a skill s of ag, i.e $s \in domquadskills(ag)$, such that $t \in skills(ag)(s)$.

Requirement 3.5

Each agent ag to whom a task t has been assigned should have enough time to move from his current location to the place where the execution of a task t he has been assigned should be carried out. That is if $place(t)$, and location(ag) denote respectivelly the execution's place t and $location(ag)$ the current lacation og ag and $mobility(location(ag), place(t))$ the time to arrive in the place $place(t)$, then $mobility(location(ag), place(t)) < start(p)$ if p is the period within which the execution of t should be taken.

Proposition 3.1 (correct execution)

Let Rn denotes a run, Rn is said to be correct, i.e $correct(Rn)$, if and only if the above requirements are satisfied.

4 Conclusion

This paper provides the foundation for the formalization of business process based on the resource constraints. The formalization gives the core features that are suitable to deal with the organisational aspect of business process management. These features are generic as they can be extended to capture the representation of various stages if the resource and business process management. They may be used in the design of the workflow process definition as well as in the definition of the execution of the workflow, the assignment of work items to resources in an enterprise or a virtual enterprise, the mobility of resources. Requirements for a correct execution of a business process based on the resource constraints have been expressed.

The paper does not deal with the concrete assignment of tasks to resources, and does not show how these features can be refined in order to model real world business processes. For future research, it would be interesting to compare this model based approach to other approaches such as Petri nets, to build some case studies. For practical purpose of the models, the definition of case studies is one the future works in order to overcome the limitations of formal methods.

References

[1] M. Abadi and leslie Lamport. Composing Specifications. Technical Report 66, System Research Center, October 10 1990.

[2] M. Abedi. An Axiomatization of Lamport's Temporal Logic of Action. Technical report, Systems Research center of Digital Equipment Corporation, 1990.

[3] J. F. Allen. Maintaining knowledge about Temporal Intervals. *Communication od ACM*, 26(11):832–843, 1983.

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

[5] Z. A. Bakar. The Adaoption of Workflow System in a Developing Country. In *ACS/IEEE International Conference on Computer Systems and Applications*, volume IEEE Catalog Number:03EX722, ISBN:0-7803-7983-7. IEEE, 2003.

[6] J. Becker, M. rosemann, and C. von Uthmann. *Business Process Management*, chapter Guideline of Business Process Modeling. Springer-Verlag Berlin Heidelbert, 2000.

[7] D. Bjorner. Domain Engineering: A Radical Innovation for Software and System Engineering. In *US Army Research Office sponsored Monterery workshop*, 2002.

[8] S. Burris and H. Sankappanavar. *A Course in Universal Algebra*. Springer Verlag, 1981.

[9] E. CA, K. K, and W. J. *Workflow management: Net-based concepts, models, techniques and Tools*, chapter Modeling workflow dynamic change using timed hybrid flow nets, pages 109–128. Number LNCS 1806. Eindhoven University Of tTechnology, 1998.

[10] C. Castelfranchi and R. Falcone. Levels of help, levels of delegation and agent modeling. In M. Tambe and P. Gmytrasiewcz, editors, *Working Notes of the AAAI - 96 Workshop on Agent modeling*, pages 1–8, Portland, OR, 1996.

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

[12] 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.

[13] J. Desel and T. Erwin. *Business Process Management*, chapter Modeling, Simulation and Analysis of Business Processes, pages 129–141. Number LNCS 1806. Springer-Verlag Berlin, 2000.

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

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

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

[17] T. R. L. Group. *The RAISE SPECIFICATION LANGUAGE*. Prentice Hall, 1992.

[18] T. Janowski and O. Adegboyega. Formalising Feasibility and correctness of distributed business processes. In *Lecture Notes in Computer Science*, number 2465, pages 432–443, 2002.

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

[20] E. A. Kendall. Role Modeling for Agent system Analysis, Design, and implementation. *IEEE Concurrency, Agent Systems and Applications*, 2000.

[21] L. Lamport. Specifying concurrent program modules. *ACM Transaction on programming Languages and systems*, 1983.

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

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

[24] D. Milojicic. Agent Systems and Applications. *IEEE Concurrency, Agent Systems and Applications*, 2000.

[25] N. Minar, M. Gray, O. Roup, R. Krikorian, and P. Maes. Hive: Distributed Agents for Networking Things. *IEEE Concurrency, Agent Systems and Applications*, 2000.

[26] S. Onoda, Y. Ikkai, T. Kobayashi, and N. Komoda. Definition of deadlock patterns for business process workflow models. In *32nd Hawai internation conference on system sciences*, 1999.

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

[28] 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.

[29] A. E. Roger and M. F. Ndjodo. Mobile-Based support for Business Processes: Feasibility and Correctness. In *ACS-IEEE International Conference on Computer Systems and Applications*, volume IEEE Catalog Number:03EX722, ISBN:0-7803-7983-7. IEEE, 2003.

[30] T. Rye and C. Paterson. The partnership approach to mobility management: an evaluation of different models for travel planning by groups of employees. In *ECOMM*, 2001.

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

[32] T. R. Stephen Ison. Lesson from travel planning and road user charging for policy-marking: Throug imperfection to implementation. In *third seminar of the IMPRINT-EUROPE, October 2002*, 2002.

[33] W. van der Aalst and al. *Business process Management*, chapter Techniques for Modelling Workflows and Their Support of Reuse, pages 1–15. Springer-Verlag Berlin Heidelberg, 2000.

[34] 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.

[35] 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.

[36] 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.

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