Regulating Work in Digital Enterprises: a Flexible Managerial Framework

Takahiro Murata and Naftaly H. Minsky
Department of Computer Science
Rutgers University
Piscataway, NJ, 08854 USA
Phone: (732) 445-3999
Fax: (732) 445-0537
E-mail: {murata|minsky}@cs.rutgers.edu

ABSTRACT

This paper demonstrates that work in digital enterprises—like work in conventional enterprises—can be carried out effectively by *autonomous agents*, subject to a regulatory regime that combines standing enterprise-wide policies with flexible managerial controls. The proposed regulatory mechanism, which is based on the concept of Law Governed Interaction (LGI), can support a wide range of enterprise policies, and a wide spectrum of managerial styles—including the procedural style underlying the so called Workflow Management System (WfMS).

INTRODUCTION

As information technology is embraced by the business world, a new mode of work is emerging. The presence of goods, funds, and services is represented electronically, and their handling (moving, storing, selling, buying, etc.) is carried out by acting upon such electronic representation. In addition, it is becoming increasingly prevalent, in such *digital enterprises*, for electronic actors, e.g., stock trading agents, auction agents, etc., to initiate such actions automatically. One consequence of these trends is that work in digital enterprises tends to be quite invisible, and be carried out in enormous speed, making such work difficult to control and to audit. There clearly is a need for new approaches for the management of this kind of work.

The currently leading approach for the management of work within digital enterprises is the, so called, Workflow Management System (WfMS). Although this concept has many variations, they are all reasonably close to the following definition in the "Workflow Reference Model" [13]:

> "A Workflow Management System is one which provides *procedural automation* [emphasis is ours] of a business process by management of the sequence of work activities and the invocation of appropriate human and/or IT resources associated with the various activity steps."

In spite of its undeniable popularity, this procedural approach to the management of business processes has its critics. One apt criticism of WfMS has been raised in [14], as follows:

> "Business processes are highly dynamic and unpredictable—it is difficult to give a complete *a priori* specification of all the activities that need to be performed and how they should be ordered. Any detailed time plans which are produced are often disrupted by unavoidable delays or unanticipated events."

As an alternative for procedural specification, these authors propose the following:

> "...the most natural way to view the business process is as a collection of *autonomous* [emphasis is ours] problem solving agents, which interact when they have interdependencies."

We too believe in the importance of autonomy for the participants in business processes, for the above mentioned reason, and for others. In particular, an autonomous agent can take "opportunity-based initiatives" [17], based on his/her[1] intimate familiarity with the operating environment, which may not be available to the manager. An attempt to communicate such information to one's manager, or to a workflow mechanism,

[1] We will henceforth use "he" when referring to a human agent, for simplicity, and we will use "it" when an agent is, or is likely to be, a software component.

could be impractical when fast response is required—particularly in a distributed setup, when the agent in question is geographically separated from his manager.

However, we maintain that such autonomy cannot be unlimited. Indeed, the autonomy of agents operating within a conventional enterprises is generally limited by the rules and regulations imposed on them by their environment. Let us examine the nature of such limitations.

Broadly speaking, one can distinguish between two types of regulations that govern agents operating within an enterprise: (a) *enterprise policies* and (b) *managerial controls*. By enterprise policies we mean the standing rules of the enterprise, regarding the behavior of its agents when involved in certain activities, and regarding the use of its resources. For example, an enterprise may have a policy stipulating that issuance of purchase orders must be subject to the availability of funds, and that it must be audited. Such a policy may also require that every purchase order is cosigned by somebody other than its originator (a case of *separation of duties*).

Of course, a typical enterprise is bound to be governed by a multitude of policies. They might have diverse reasons for their existence, such as: (a) the internal business practices of the enterprise; (b) product design and manufacturing constraints; (c) the auditability of the enterprise; (d) security considerations; (e) software engineering principles; and (f) government regulations. Such policies are likely to be interrelated in complex ways, forming an *ensemble* that is to govern the enterprise as a whole.

By *managerial controls* we mean more timely, and usually more detailed, specifications of what is to be done, when it should be done, and by whom. Such controls can be exercised at various levels of specificity. In particular, a manager can give each of its subordinates some broadly specified tasks, providing them with appropriate resources, such as budget, space, etc.; and then let them do their work autonomously. A more hand-on manager might monitor the progress of its subordinates, and, if necessary, steer them towards a desired goal. Finally, a manager might micro-manage the task he is in charge of, by dynamically assigning a specific agent to each specific task, when it is to be carried out. This extreme mode of management, which severely curtails the autonomy of the agents, is what the workflow management systems generally employs.

Note that these two types of regulation, i.e., enterprise policies and managerial controls, are interrelated. First, because the ability of managers to manage must be grounded on some enterprise policies, which provide managers with a degree of control over their subordinates. And, second, because agents must generally conform to the enterprise policies governing them, even when following the dictates of their managers.

The purpose of this paper is to demonstrate that work in digital enterprises—like work in conventional enterprises—can be carried out effectively by *autonomous agents*, subject to a regulatory regime that combines standing enterprise-wide policies with flexible managerial controls. The proposed regulatory mechanism can support a wide range of enterprise policies, and a wide spectrum of managerial styles—including the procedural style underlying workflow management systems.

The computational means for our regulatory mechanism are based on a generic coordination and control mechanism for distributed systems called Law Governed Interaction (LGI) [19]. This mechanism provides for the explicit specification, and for the scalable enforcement, of policies governing the interaction between distributed agents.

We start in the next section with an example designed to motivate the need for flexible regulation of work in digital enterprises, and to demonstrate the nature of the required regulatory mechanism. We then provide an overview of LGI, which is the basis for the regulatory mechanism being proposed here. This is followed by the demonstration of how the example policy introduced earlier can be formulated and enforced under LGI. Finally we discuss related research, including workflows, and we conclude.

MANAGEMENT BY REGULATION: A CASE STUDY

Consider a department store that deploys teams of agents whose purpose is to supply its various departments with the merchandise they need. Each such team consists of a set of *buyers*, and a *manager*; and there is a distinguished agent called the *auditor*, whose job is to maintain the audit trail of the various purchasing activities. Let all such teams be governed by the following policy (which we call BT, for "buying-team"):

1. *A manager can give each buyer in his team a collection of assignments, each consisting of the identification of the merchandise to be purchased, and the deadline for the purchase to be carried out. The manager can also provide each buyer with a budget for all his purchases.*
2. *Every buyer is allowed to issue purchase orders (POs) as set forth by his current assignments, while remaining within his budget. Every PO issued by a buyer is to be monitored by the auditor.*
3. *A buyer can transfer any of his assignments to another buyer in his team, unless the assignment is marked as "exclusive."*

4. *The manager can check the status of each buyer, in terms of the progress of his assignments, and his remaining budget. He also monitors any transfer of assignments between buyers.*

5. *The manager can change the assignments and the budget of every buyer in his team. He can also mark an assignment as "exclusive," thus preventing it from being transferred to another buyer as described in item 3 above.*

6. *The role played by all participants in this activity—both managers and buyers—needs to be certified by a specific certification authority (CA), called here* admin.

This is a reasonable policy for a traditional department store to employ (with the possible exception of item 6 above). Later we will show how this policy can be formalized as a law under LGI, and enforced on a more digital department store.

On the Nature of This Policy

One can distinguish between two aspects of this policy: (a) its broad regulation of the activities of the otherwise autonomous buyers, and (b) the control it provides a manager over the buyers working for him.

First, buyers are provided by this policy with a carefully circumscribed autonomy. According to items 2, each buyer can satisfy his assignments by issuing purchase orders to vendors of their choice, at arbitrary prices—within his total budget—and in arbitrary order. Moreover, according to item 3 of this policy, a buyer can transfer to other members of his team any of his non-exclusive assignments. Such an autonomy is very important, particularly if the buyers travel around the country, or around the world, and have to make their decisions on the basis of what they find in the field. Note that such autonomy of action does not imply any loss of accountability, because this policy requires all issuance of POs to be monitored by the auditor, and all exchanges of assignments to be monitored by the manager. So, buyers cannot conduct their deals in the dark.

Second, the above mentioned autonomy of buyers can be restricted in various ways by their manager, who is given a great deal of control over them by this policy. First, by item 1 of policy BT, it is the manager who can define the initial assignments of his/her buyers, and who can provide them with their budgets. Second, by item 4, the manager can monitor the progress of each of his buyers, and by item 5, he can steer the activities of his team of buyers dynamically, by changing their assignments and their budgets, at will.

A good manager is likely to use his ability to control his buyers sparingly. But a manager does have it in his power, under policy BT, to force his team of buyers to operate as a kind of workflow. That is, he can give a specific assignment, and a suitable budget, to a specific buyer, when he wants this particular buyer to carry out this assignment. And then, when this assignment is carried out, he can give the next assignment to the next buyer, and so on. This would be a very inefficient way for managing teams of buyers, in most circumstances. But it might be appropriate in some rare situations.

We would like to point out—what may already be obvious to the reader—that there is no primitive concept of a *manager* in our framework. An agent playing the role of a manager under policy BT has specific powers over members of his team, which allows him to manage his team's work as has been described above. But these powers have been defined specifically by this policy. It is quite possible to provide managers with different types of power; or to provide a single team with several managers, each with its own powers; or to have a team operate as a set of peers, with no manager to supervise them. In short, it is the policy that is fundamental in this framework, not the manager.

Finally, we point out that all but the last items of this policy can be reasonably formulated for a traditional enterprise. Item 6, however, is meaningful only for a digital enterprise. This item prescribe the manner in which all participants in the purchasing activities need to authenticate their roles—via certificates issued by a specified certification authority.

Broader Perspectives

So far we have treated BT as an isolated policy, governing purchasing activities without any regard to whatever else is going on in the enterprise in question. This is an oversimplification. As we have already pointed out, a typical enterprise is bound to be governed by a whole ensemble of interrelated policies. As a simple example, when a buyer, operating under policy BT, makes a purchase he may need to reserve transportation for it. But the agents that deal with transportation are likely to be operating under their own policy, which may require these two policies to interoperate. We will also describe later how such interoperation between policies is carried out under the proposed regulatory mechanism.

THE CONCEPT OF LAW GOVERNED INTERACTION (LGI)–AN OVERVIEW

As mentioned earlier, we turn to Law Governed Interaction (LGI) [19] as an underlying communication and computational mechanism of our framework. Broadly speaking, LGI is a message-exchange mechanism that allows a group of distributed agents to engage in a mode of interaction *governed* by an explicitly specified policy, called the *law* of the group. The messages thus exchanged under a given law $\mathcal{L}$ are called $\mathcal{L}$-*messages*,

and the group of agents interacting via $\mathcal{L}$-messages is called an $\mathcal{L}$-*community* $\mathcal{C}_\mathcal{L}$ (or simply community). No assumptions are made about the structure and behavior of the community member agents[2], which might be software processes, or human beings.

For each agent `x` in a given community $\mathcal{C}_\mathcal{L}$, LGI maintains the *control-state* $\mathcal{CS}_x$ of this agent, in the form of a bag of terms. These control-states, which can change dynamically, subject to law $\mathcal{L}$, enable the law to make distinctions between agents, and to be sensitive to dynamic changes in their state. The semantics of control-states for a given community is defined by its law, and could represent such things as the role of an agent in this community, along with privileges and tokens it carries. For example, under the law introduced later to implement the regulation discussed earlier for the buyers' teams, the term `manager` in the control-state of an agent denotes that this agent plays the role of the manager in the buyers' mission.

We elaborate on several aspects of LGI below, focusing on (a) its concept of law, (b) its mechanism for law enforcement, and (c) its interoperability between communities, while additional features of LGI essential to enforcing the buyers' team policy will be explained later when the actual law is presented. For the aspects of LGI not covered here, including the treatment of *exceptions*, the expressiveness, and the deployment of LGI communities, the reader is referred to [19, 24, 2].

The Concept of Law: Law $\mathcal{L}$ of community $\mathcal{C}_\mathcal{L}$ is defined over a certain types of events, occurring at members of $\mathcal{C}$. These events subject to laws, called *regulated events*, include (among others): the *sending* and the *arrival* of an $\mathcal{L}$-message; the *coming due of an obligation* previously imposed on a given agent; and the *submission of a digital certificate*. Given a regulated event, law $\mathcal{L}$ produces its mandate called the *ruling* for that event. A ruling is made of a list of *primitive operations*, which include operations on the control-state of the agent where the event occurred (called, the "home agent"), such as insertion (`+t`), removal (`-t`), and replacement (`t<-s`) of terms; operations on messages, such as `forward` and `deliver`; and the imposition of an obligation on the home agent.

Law $\mathcal{L}$ is enforceable strictly *locally* at each member of $\mathcal{C}$, leading to scalability. This is due to the following: (1) $\mathcal{L}$ only regulates local events at individual agents; (2) the ruling of $\mathcal{L}$ for an event `e` at agent `x` depends only on `e` and the local control-state $\mathcal{CS}_x$ of `x`; (3) the ruling of $\mathcal{L}$ at `x` can mandate only local operations to be carried out at `x`. Note that we lose no expressiveness in stating policies to be enforced, by limiting ourselves to such local laws, as shown in [19].

[2]Given the popular usages of the term "agent," it is important to point out that we do not imply by it either "intelligence" nor mobility, although neither of these is being ruled out by this model.

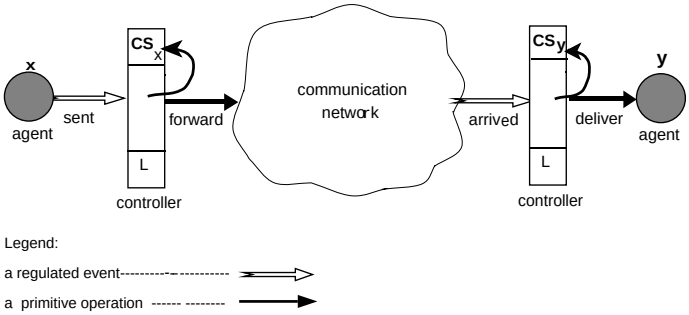

Figure 1: Enforcement of the law

Law-Enforcement Mechanism: The law $\mathcal{L}$ of community $\mathcal{C}$ is enforced by a *middleware* consisting of a set of trusted agents called *controllers* that mediate the exchange of $\mathcal{L}$-messages between members of $\mathcal{C}$. Every member `x` of $\mathcal{C}$ has a controller $\mathcal{T}_x$ assigned to it ($\mathcal{T}$ here stands for "trusted agent") which maintains the control-state $\mathcal{CS}_x$ of its client `x`. Controllers are *generic* in that it can interpret and enforce any well-formed law; they operate as independent processes anywhere in the network, and a set of active controllers is maintained by *controller-service*. (Concerning the basis for trust in this mechanism, the reader is referred to [3].)

All these controllers, which are logically placed between the members of $\mathcal{C}$ and the communications medium (as illustrated in Figure 1) carry the *same law*[3] $\mathcal{L}$. Every exchange between a pair of agents `x` and `y` is thus mediated by *their* controllers $\mathcal{T}_x$ and $\mathcal{T}_y$, so that this enforcement is inherently decentralized. Several agents can share a single controller, if such sharing is desired. (The efficiency of this mechanism, and its scalability, are discussed in [19].)

The fact that the same law is enforced at *all* agents of a community gives LGI its necessary global scope, establishing a *common* set of ground rules for all members of $\mathcal{C}$ and providing them with the ability to trust each other, in spite of the heterogeneity of the community.

Interoperability Between Communities: Finally, we point out that LGI supports the ability of agents `x` and `y` operating under distinct laws $\mathcal{L}$ and $\mathcal{L}'$, respectively, to exchange messages. Such interoperation is regulated under the authorization by each law, and may cause an effect only on the recipient of a message as stipulated by the recipient's law. Consequently, interoperating parties need not be aware of the details of each other law.

[3]A one-way hash [21] of the law is used for the controllers to establish the identity of the laws they carry.

To achieve such interoperability, primitive operation `export` and regulated event `imported` are provided as follows:

- `export(x,m,[y,L'])`: invoked by agent `x` under law $\mathcal{L}$; initiates the transmission of a message `m` from `x` to agent `y` operating under law $\mathcal{L}'$.
- `imported([x,L],m,y)`: occurs when a message `m` exported by `x` under law $\mathcal{L}$ arrives at `y` under $\mathcal{L}'$.

ESTABLISHING THE BT POLICY IN A DIGITAL CONTEXT

We will formulate here our example policy BT as a law $\mathcal{L}_{\mathcal{BT}}$ under LGI, and then explain how this law is enforced. Then, we illustrate, via a very simplified example, how agents operating under this law can interoperate with agents operating under different laws in the enterprise.

Note that law $\mathcal{L}_{\mathcal{BT}}$ is formulated under the following simplifying assumptions: (1) every purchase is made electronically; (2) any PO issued by an authorized buyer will not be rejected by the vendor chosen for that purchase; (3) the payment and the delivery for any purchase will be carried out accordingly once the PO has been sent; and (4) there is only one team of buyers (this assumption, which can be easily removed, is to avoid the need to identify the team, when buyers and their manager are authenticated under item 6 of policy BT).

Law $\mathcal{L}_{\mathcal{BT}}$ of Buyers' Team

Shown in Figures 2 and 3, law $\mathcal{L}_{\mathcal{BT}}$ implements the policy stipulated earlier, which regulates the entire interaction of the buyers' mission. In general an LGI law consists of two parts: the preamble and the rule section. The preamble of $\mathcal{L}_{\mathcal{BT}}$ specifies the following: (1) `admin` as a trusted CA, identified by its public key; (2) the certification requirement on the controllers that interpret this law, with the public key of the CA to certify them (and optionally the attributes that the controllers need to be certified about), whose compliance is verified by a controller-server at the time of the agent's joining this $\mathcal{L}$-group; (3) `auditor`, with its identifier, for auditing; and (4) the initial control state given to every agent that adopts the law, in this case empty.

The rest of the law stipulates a set of rules, in a Prolog-like syntax. Most rules are followed by a comment (in italic), which, together with our discussion, should be understandable even for a reader not well versed in the LGI language for writing laws. Each rule has a *head*, up to symbol `:-`, and a *body*, the rest. Recall that the same law is interpreted though individually by the controller assigned to each agent in the community. A regulated event occurring at this agent triggers a rule that has a matching head, if any (in the order in which the rules are written when more than one match). The triggered rule proceeds to check if all the goals in its body are attained, given the context of the control-state of this agent.

In addition to the standard types of Prolog goals, the body of a rule may contain two distinguished types of goals. These are the *sensor-goals*, which allow the law to "sense" the control-state of the home agent, and the *do-goals* that contribute to the ruling of the law. A *sensor-goal* has the form `t@CS`, where `t` is any Prolog term. It attempts to unify `t` with each term in the control-state of the home agent. A *do-goal*, which always succeeds, has the form `do(p)`, where `p` is one of the primitive-operations, mentioned in the LGI overview. It appends the term `p` to the ruling of the law. Thus, a successful rule body leads to non-empty ruling if it contains do-goals. (By default, an empty ruling implies that the event in question has no consequences—such an event is effectively ignored.)

Rule $\mathcal{R}1$ implements policy item 6, which regulates the status of buyers, and that of the manager. The `certified` event that triggers this rule is generated when the controller is presented with a valid certificate, i.e., duly signed by an authority declared in an `authority` clause, in this case `admin`.[4] The `certified` event has as its argument the following representation of the submitted certificate: `[issuer(admin), subject(Self), attributes([role(R)])]`. Term `issuer(admin)` tells about the issuer of the certificate, while `subject(Self)` is used to signify the subject of the certification. `Self` is an LGI built-in variable bound to the identifier (id) of the home-agent[5], which means that this rule requires a self-certificate be presented. Term `attributes` in the above argument describes what is certified about the subject, and in this case asserts that the agent should be allowed to play role `R`, either `buyer` or `manager`[6], which is recorded in the control-state by this rule.

Rules $\mathcal{R}2$ and $\mathcal{R}3$ allow the manager to impose an (additional) assignment to a buyer. $\mathcal{R}2$ checks if the sender has the manager status before forwarding the message, either of `asmt` or of `budget` (`Msg` is another LGI built-in variable bound to the regulated message).[7], $\mathcal{R}3$, after ensuring that the recipient is a buyer, adds a `asmt` term to the control-state, representing this assignment item, including the information regarding its deadline and whether it is exclusive (`excl`) or transferable (`trans`), bound to `Due` and `Excl`, respectively. The second argu-

[4] If the certificate is found invalid, then an `exception` event is triggered.

[5] An agent id is of the form: `local-name@domain-name` [2].

[6] Syntax `(P;Q)` in the laws should read `P` **or** `Q`; similarly `(P->Q;R)` means **if** `P` **then** `Q` **else** `R`.

[7] When the addressed recipient is absent from the group, an LGI exception is raised; again, the handling of this exception and other similar ones are omitted in this paper for brevity.

Preamble:

```
authority(admin, publicKeyOfAdmin).
portal(thisLaw, publicKeyOfAdmin, []).
alias(auditor, "auditor@someDepartment.com").
initialCS([]).
```

$\mathcal{R}1$.
```
certified([issuer(admin),subject(Self),
      attributes([role(R)])])
  :- (R=buyer ; R=manager), do(+R).
```

Every participant needs to be certified via admin *for the respective role.*

$\mathcal{R}2$.
```
sent(M, Msg, B)
  :- (Msg=addAsmt(Spec,Due,Excl);
      Msg=delAsmt(Spec); Msg=budget(Op,A)),
     manager@CS, do(forward).
```

$\mathcal{R}3$.
```
arrived(M, addAsmt(Spec,Due,Excl), B) :- buyer@CS,
      (not(mgr(_)@CS)->do(+mgr(M)); true),
      do(+asmt(Spec,act,Due,Excl)),
      do(imposeObligation(asmtDue(Spec),Due)),
      do(deliver).
```

$\mathcal{R}4$.
```
arrived(M, delAsmt(Spec), B) :- buyer@CS,
      do(-asmt(Spec,act,_,_)), do(deliver).
```

The manager can impose some (additional) assignment on a buyer or take some off.

$\mathcal{R}5$.
```
obligationDue(asmt(Spec))
  :- asmt(Spec,act,Due,Excl)@CS,
     do(asmt(Spec,act,Due,Excl)
       <-asmt(Spec,unf,Due,Excl)).
```

When an assignment turns due without the merchandise purchased, it is classified as unfulfilled.

$\mathcal{R}6$.
```
arrived(M, budget(Op,A), B) :- buyer@CS,
      (Op=add->(budget(A1)@CS,A2 is A1+A,
          do(budget(A1)<-budget(A2));
          do(+budget(A))) ;
       Op=del->budget(A1)@CS,A2 is A1-A,
          do(budget(A1)<-budget(A2))),
      do(deliver).
```

Similarly, the manager can add/delete some amount to/from a buyer's budget.

Figure 2: Law $\mathcal{L}_{\mathcal{BT}}$ of buyers' team

$\mathcal{R}7$.
```
sent(B, po(Spec,P), S) :- buyer@CS, budget(A)@CS,
      asmt(Spec,act,D,E)@CS,
      R is A-P, R>=0, do(budget(A)<-budget(R)),
      do(asmt(Spec,act,D,E)<-asmt(Spec,P,D,E)),
      do(deliver(B,Msg,S)),
      do(deliver(B,po(Spec,P,S),auditor)).
```

A buyer can send a PO, while not exceeding his budget.

$\mathcal{R}8$.
```
sent(B, transfer(Spec), B1)
  :- buyer@CS, asmt(Spec,act,D,trans)@CS,
     do(-asmt(Spec,act,D,trans)),
     do(forward(B,transfer(Spec,D),B1)),
     mgr(Mgr)@CS,
     do(deliver(B,monitored(Msg,to(B1)),Mgr)).
```

$\mathcal{R}9$.
```
arrived(B, transfer(Spec,D), B1)
  :- (buyer@CS -> do(+asmt(Spec,act,D,trans)),
        imposeObligation(asmtDue(Spec)),
        do(deliver);
      do(forward(B1,
        notApply(transfer(Spec,D)),B))).
```

$\mathcal{R}10$.
```
arrived(B1, notApply(transfer(Spec,D)), B)
  :- do(+asmt(Spec,act,D,trans)), do(deliver).
```

A buyer can transfer some assignment to another buyer unless marked "exclusive."

$\mathcal{R}11$.
```
sent(M, checkStatus, B)
  :- manager@CS, do(forward).
```

$\mathcal{R}12$.
```
arrived(M, checkStatus, B)
  :- buyer@CS, budget(A)@CS,
     findall(asmt(S1,P),
        (asmt(S1,P,D1,E1)@CS,number(P)), Fs),
     findall(asmt(S2,act),
        asmt(S2,act,D2,E2)@CS, As),
     findall(asmt(S3,unf),
        asmt(S3,unf,D3,E3)@CS, Us),
     do(deliver(B,status(B,[active(As),
        filled(Fs),unfilled(Us),budget(A)]),M)).
```

The manager can check the status of a buyer.

Figure 3: Law $\mathcal{L}_{\mathcal{BT}}$ of buyers' team (cont'd)

ment `act` of this `asmt` term stands for this assignment item being active. In parallel, $\mathcal{R}4$, in reducing the assignment, removes the corresponding `asmt` term from the recipient's control-state. Note, when an assignment is added, LGI obligation named `asmtDue(Spec)` is imposed to manage the deadline of this assignment (whose becoming due will be explained shortly). Thus these rules implement a part of policy items 1 and 5 that concerns assigning and adjusting the buyers' tasks. Similarly, the other part of these policy items for setting and adjusting the buyers' budget is implemented by rules $\mathcal{R}2$ and $\mathcal{R}6$.

By rule $\mathcal{R}7$, a buyer can issue a PO for a merchandise, assigned, but not obtained yet, within the remaining budget. This rule changes the status of this assignment from active to fulfilled, by replacing the second argument of the `asmt` term from `act` to the price of the merchandise (`P`). Then, the PO is delivered to the intended supplier.[8] Notice that the message content, together with the PO recipient's id, is also delivered by $\mathcal{R}7$ to `auditor` for auditing. The deadline expiry of an assignment is handled by rule $\mathcal{R}5$, which revises the status of this assignment to be unfulfilled if it is still active at this point. Thus, these rules implement policy item 2.

Rules $\mathcal{R}8$ and $\mathcal{R}9$ implement policy item 3 that permits buyers to coordinate by exchanging their assignment. A buyer can send a `transfer` message specifying an item to be transferred ($\mathcal{R}8$). However, notice that the speci-

[8] Sending a message via `deliver` to a non-home-agent triggers no regulated event on the recipient side (just like `deliver` to a home-agent, but unlike `forward`).

fied item must be transferable; i.e., the fourth argument of the corresponding `asmt` term should be `trans` (not `excl`), which is removed from the control-state. The arrival of the transfered item adds the corresponding `asmt` term to the control-state ($\mathcal{R}9$). Note that if the recipient is not a buyer, the transfered item is returned in a `notApply` message, whose arrival causes $\mathcal{R}10$ to add the `asmt` term back to the control-state of the original sender.

Rules $\mathcal{R}11$ and $\mathcal{R}12$ implement a part of policy item 4 that permits the manager to check the status of each buyer. $\mathcal{R}12$ uses (higher-order) built-in operation `findall` to pick up all assignments whose corresponding `asmt` terms qualify the specification given as its second parameter, and to accumulate them in a list bound to its third parameter; the first, the second, and the third invocation of `findall` are to collect the filled assignments, the active ones, and the unfilled (expired) ones, respectively. Combined with the remaining budget, these are delivered in a `status` message back to the manager.

The rest of item 4 that allows the manager to monitor assignment exchange among buyers is implemented by rule $\mathcal{R}8$, which delivers a notice of the occurrence of this monitored event to the manager. The manager's id in term `mgr` of the control-state should have been inserted by $\mathcal{R}3$ when the buyer first received an assignment from the manager.

Interoperability Between Policies

So far we have concentrated on a single policy; namely, one that governs the purchasing of merchandises. However it is highly likely that such an activity is conducted in association with other activities inside (or outside) the pertinent enterprise, each of which regulated by its own policy. Thus, there ought to be a means to establish interoperability between those policies, which we illustrate by building on our previous example.

Suppose, having purchased a merchandise, buyers acting under law $\mathcal{L}_{\mathcal{BT}}$ should reserve the means of its transportation through some authorized agent. Such an agent is engaged in finding a means of transportation upon receiving a specific request from a buyer, which is regulated by law $\mathcal{L}_{\mathcal{T}}$. Having located and booked a means to qualify the given request, the transportation agent notifies the requester of the booking information; failing to book any means within the deadline specified by the requester, the agent must notify the requester of the failure.

Using interoperability of laws in LGI, we achieve this interaction between two policies, one for purchasing, and the other for reserving transportation means, as follows.

In Figure 4, we show an additional portion of law $\mathcal{L}_{\mathcal{BT}}$ to enable this interoperation. First, it has an additional portal clause in the preamble, which identifies law $\mathcal{L}_{\mathcal{T}}$, shown below, as a law to interoperate with. Note the name `transportationLaw` given to $\mathcal{L}_{\mathcal{T}}$ is only local to this law, $\mathcal{L}_{\mathcal{BT}}$, and the hash of $\mathcal{L}_{\mathcal{T}}$ is used to identify it. Then, rule $\mathcal{R}13$ allows a buyer, who has purchased an assigned merchandise, to send a request message to a transportation agent, specifying the merchandise, the origin, the required arrival date, and the deadline for finding an appropriate means. Note that the rule prohibits the buyer to send more than one request while an agent is searching for a means. The message is conveyed to law $\mathcal{L}_{\mathcal{T}}$ via `export` operation.

Preamble:

```
    portal(transportationLaw, hashOfTransportationLaw).
```

$\mathcal{R}13$.

```
    sent(B, reqTrans(Spec,Fm,By,Due), T)
      :- buyer@CS, asmt(Spec,P,_,_)@CS, number(P),
         not(requested(Spec)), do(+requested(Spec)),
         do(export(B,Msg,[T,transportationLaw])).
```

$\mathcal{R}14$.

```
    imported([T,transportationLaw],
         booked(Spec,Tspec), B)
      :- do(+trans(Spec,Tspec)), do(deliver).
```

$\mathcal{R}15$.

```
    imported([T,transportationLaw], notFound(Spec),
         B) :- do(-requested(Spec)), do(deliver).
```

Figure 4: Law $\mathcal{L}_{\mathcal{BT}}$ of buyers' team (interoperability)

Rules $\mathcal{R}14$ and $\mathcal{R}15$ are triggered when an agent acting under $\mathcal{L}_{\mathcal{T}}$ exports a `booked` message and a `notFound` message, respectively, as described below.

Figure 5 is an excerpt of law $\mathcal{L}_{\mathcal{T}}$ that governs the activity of locating and reserving transportation means. (It does not show how an agent goes about making a reservation with a carrier.) Its preamble has a `portal` clause to declare reciprocally $\mathcal{L}_{\mathcal{BT}}$ for interoperation, under local name `buyersLaw`. The entry of role `transportation` allowed by $\mathcal{R}1$, via an appropriate certificate issued by admin, is similar to $\mathcal{R}1$ of $\mathcal{L}_{\mathcal{BT}}$.

The `imported` event generated on the receipt of a request message conveyed from $\mathcal{L}_{\mathcal{BT}}$ to $\mathcal{L}_{\mathcal{T}}$ triggers rule $\mathcal{R}2$. After making sure that the recipient is a transportation agent, and storing the request information, this rule imposes an obligation named `replyDue` to issue a failure message back to the requester ($\mathcal{R}4$), if no such means is found. Finally, rule $\mathcal{R}3$ allows a transportation agent to send the booking information for a requested means, which checks the presence of the corresponding `request` term, and verifies the destination of such a message.

As seen above, in administrating the transportation reservation, the effect of receiving a request is deter-

```
Preamble:
    authority(admin, publicKeyOfAdmin).
    portal(thisLaw, publicKeyOfAdmin, []).
    portal(buyersLaw, hashOfBuyersLaw).
    initialCS([]).

R1. certified([issuer(admin),subject(Self),
        attributes([role(transportation)])])
      :- do(+transportation).

R2. imported([B,buyersLaw], reqTrans(Spec,Fm,By,Due),
        T) :- transportation@CS,
        do(request(Spec,B)@CS),
        do(imposeObligation(replyDue(Spec),Due)),
        do(deliver).

R3. sent(T,booked(Spec,Tspec),B)
      :- request(Spec,B)@CS, do(-request(Spec,B)),
        do(export(T,Msg,[B,buyersLaw])).

R4. obligationDue(replyDue(Spec))
      :- request(Spec,B)@CS, do(-request(Spec,B)),
        do(export(Self,notFound(Spec),
          [B,buyersLaw])).
```

Figure 5: Law $\mathcal{L}_{\mathcal{T}}$ for transportation

mined by law $\mathcal{L}_{\mathcal{T}}$ alone, while, regarding the legitimacy of such a request, the buyers' purchasing activity, regulated by law $\mathcal{L}_{\mathcal{BT}}$, is trusted. In particular, $\mathcal{L}_{\mathcal{T}}$ does not reflect under what condition a buyer can send such a request, and leaves it to the regulation under $\mathcal{L}_{\mathcal{BT}}$. (Similarly, the legitimacy of booking information received by a buyer under $\mathcal{L}_{\mathcal{BT}}$ derives from the fact that it originates an agent acting under $\mathcal{L}_{\mathcal{T}}$, regardless of the detail of its regulation.) Such transparency is an important factor in achieving interoperability among multiple policies without incurring unnecessary administrative overhead.

COMPARISON WITH RELATED WORK

We will discuss here briefly related work concerning three aspects of this paper: workflows management systems, the autonomy of workers, and enterprise-wide policies.

Workflows: Broadly speaking, a workflow [13] is a procedural specification of a sequence of tasks to be performed by participating agents. For enacting such specification, a WfMS is deployed to initiate planned tasks one by one. This general concept has many variations, generated by researchers attempting to enhance its flexibility, by such means as: handling of workflow exceptions [12, 20, 8]; run-time evolution of the process models [10, 7]; accommodating different degrees in the models' specificity [5]; and generating the process model at run-time [15]. All these variations can be incorporated into a community governed by a law such as $\mathcal{L}_{\mathcal{BT}}$, by having the manager instruct each agent what to do, when; or even by adopting a WfMS as an LGI agent playing the role of the manager.

So, workflows represent just one management style, among many. In particular, they do not give any support to styles such as: (1) having no explicit managerial roles, allowing the peers to completely coordinate by themselves, under a policy that imposes certain rules of engagement; and (2) having more than one manager monitor and steer the activity, while interacting between themselves (to adjust the "territory" from time to time, say).

Moreover, there is a problem with WfMS, even for one who likes their management style. The problem is that workflows do not have the means to ensure that all the managerial role players and participating agents conform to the given enterprise-wide policies. Granted that some workflow systems, e.g., [6], provide a language to specify constraints on assigning tasks to agents, along with an algorithm to enforce them. However, such an approach cannot ensure a policy that ranges over multiple, possibly heterogeneous WfMSs; e.g., requiring that the issuing of POs from all buyers' teams, each supported by a distinct WfMS of its department, be monitored by a designated auditor.

The Autonomy of Agents: Several researchers [23, 17, 14] consider the the autonomy of participating agents to be crucial for conducting work effectively. Each of these proposal has its own approach for such agents to collaborate harmoniously towards a common goal. Action Workflow [17], in particular, is conceptualized as a counter-theme to the workflow model, which relies on "better educated workers who combine structured work with opportunity-based initiative and individual responsibility for quality and customer satisfaction." Jennings et al. [14] view a business process as collection of autonomous problem solving agents, who negotiate the terms and conditions of the service to be provided, in order to cope with a highly dynamic, unpredictable aspect of business processes.

However, all these researchers seem to miss the need for limiting the autonomy of agents by enterprise policies and managerial controls.

Enterprise-Wide Policies: Such policies are widely considered fundamental to enterprise modeling, and their specification were the subject of several recent investigations, such as [9]. But this modeling work is interested mostly in the specification of policies, and not in their enforcement. Enforcement of enterprise-wide policies has been addressed mostly in the context of access control. Several policy enforcement mechanism have been proposed [11, 4, 16]. All of them rely on a centralized enforcement mechanism, however. We believe that such centralization constitute a dangerous single-point of failure and performance bottleneck, and is thus not scalable. Attempts to solve these problems through

replication leads to difficulty in enforcing *stateful* policies, such as those we have described here, which are necessary for proper regulation of work in digital enterprises.

CONCLUSION

We have demonstrated in this paper that work in digital enterprises can be carried out effectively by *autonomous agents*, subject to a regulatory regime that combines standing enterprise-wide policies with flexible managerial controls. The proposed regulatory mechanism can support a wide range of enterprise policies, and a wide spectrum of managerial styles—including the procedural style underlying workflow management systems.

The regulatory mechanism described in this paper has been implemented. And it has been tested on several applications relevant to this paper. These include establishing a policy that governs the work of distributed committees [22]; establishing a policy that enforces certain broad accounting principles over the work done in a financial system [18]; and establishing an access control policy that manages dissemination of patient records in a medical center [3]. But much work remains to be done for making this technique usable in industrial context. In particular, it is necessary to support the complex ensemble of interrelated policies that governs the various activities within a typical enterprise, following preliminary work in this direction that has been published in [1].

REFERENCES

1. X. Ao, Minsky N. H., and T. Nguyen. A hierarchical policy specification language, and enforcement mechanism, for governing digital enterprises. In *Proc. of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks Monterey California*, June 2002. (available from `http://www.cs.rutgers.edu/~minsky/pubs.html`).

2. X. Ao, N. Minsky, T. Nguyen, and V. Ungureanu. Law-governed communities over the internet. In *Proc. of Fourth International Conference on Coordination Models and Languages; Limassol, Cyprus; LNCS 1906*, pages 133–147, September 2000. (available from `http://www.cs.rutgers.edu/~minsky/pubs.html`).

3. X. Ao, N. Minsky, and V. Ungureanu. Formal treatment of certificate revocation under communal access control. In *Proc. of the 2001 IEEE Symposium on Security and Privacy, May 2001, Oakland California*, May 2001. (available from `http://www.cs.rutgers.edu/~minsky/pubs.html`).

4. J. Barkley, K. Beznosov, and J. Uppal. Supporting relationships in access control using role based access control. In *Proceedings of the Fourth ACM Workshop on Role-Based Access Control*, pages 55–65, October 1999.

5. A. Bernstein. How can cooperative work tools support dynamic group processes? Bridging the specificity frontier. In *Proceedings of ACM 2000 Conference on Computer Supported Cooperative Work*, December 2000.

6. E. Bertino, E. Ferrari, and V. Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security*, 2(1):65–104, February 1999.

7. F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow evolution. *Data & Knowledge Engineering*, 24(3):211–238, January 1998.

8. D.K.W. Chiu, Q. Li, and K. Karlapalem. A meta modeling approach to workflow management systems supporting exception handling. *Information Systems*, 24(2):159–184, 1999.

9. J. Cole, Derrick J., Z. Milosevic, and K. Raymond. Policies in an enterprise specification. In Morris Sloman, editor, *Proc. of Policy Worshop, 2001, Bristol UK*, January 2001.

10. C.A. Ellis and K. Keddara. ML-DEWS: Modeling language to support dynamic evolution with workflow systems. *Computer Supported Cooperative Work*, 9(3/4):293–333, November 2000.

11. D. Ferraiolo, J. Barkley, and R. Kuhn. A role based access control model and refernce implementation within a corporate intranets. *ACM Transactions on Information and System Security*, 2(1), February 1999.

12. C. Hagen and G. Alonso. Exception handling in workflow management systems. *IEEE Transactions on Software Engineering*, 26(10):943–958, October 2000.

13. D. Hollingsworth. The workflow reference model. Technical Report WFMC-TC-1003 Issue 1.1, WfMC, January 1995.

14. N.R. Jennings, P. Faratin, M.J. Johnson, P. O'Brien, and M.E. Wiegand. Using intelligent agents to manage business processes. In *Proc. of First Intn'l Conf. on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM96)*, pages 345–360, 1996.

15. H. Jørgensen. Interaction as a framework for flexible workflow modelling. In *Proceedings of ACM 2001 Intn'l Conf. on Supporting Group Work (GROUP 2001)*, October 2001.

16. G. Karjoth. The authorization service of tivoli policy director. In *Proc. of the 17th Annual Computer Security Applications Conference (ACSAC 2001)*, December 2001. (to appear).

17. R. Medina-Mora, T. Winograd, R. Flores, and F. Flores. The action workflow approach to workflow management technology. In *Proceedings of ACM 1992 Conference on Computer Supported Cooperative Work*, pages 281–288, November 1992.

18. N.H. Minsky. Establishing accounting principles as invariants of financial systems. In *Proc. of the Fourth International IFIP TC-11 WG 11.5 Conference on Integrity and Internal Control in Information Systems, Brussels, Belgium*, November 2001. (available from `http://www.cs.rutgers.edu/~minsky/pubs.html`).

19. N.H. Minsky and V. Ungureanu. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *TOSEM, ACM Transactions on Software Engineering and Methodology*, 9(3):273–305, July 2000. (available from `http://www.cs.rutgers.edu/~minsky/pubs.html`).

20. T. Murata and A. Borgida. Handling of irregularities in human centered systems: A unified framework for data and processes. *IEEE Transactions on Software Engineering*, 26(10):959–977, October 2000.

21. B. Schneier. *Applied Cryptography.* John Wiley and Sons, 1996.

22. C. Serban, X. Ao, and N.H. Minsky. Establishing enterprise communities. In *Proc. of the 5th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2001), Seattle, Washington*, September 2001. (available from `http://www.cs.rutgers.edu/~minsky/pubs.html`).

23. L.A. Suchman. *Plans and Situated Actions: The Problem of Human-Machine Communication.* Cambridge University Press, 1987.

24. V. Ungureanu and N.H. Minsky. Establishing business rules for inter-enterprise electronic commerce. In *Proc. of the 14th International Symposium on DISTributed Computing (DISC 2000); Toledo, Spain; LNCS 1914*, pages 179–193, October 2000. (available from `http://www.cs.rutgers.edu/~minsky/pubs.html`).