Journal of the Association for Information Systems

Design Considerations for Incorporating Flexible Workflow and Multi-Agent Interactions in Agent Societies

N. C. Narendra
Technology Cell, E-Solutions Center
Hewlett-Packard India Software Operations Ltd.
29 Cunningham Road
Banglore 560 052 INDIA
ncnaren@india.hp.com

ABSTRACT

In this paper, we present our conception of a *Flexible Agent Society (FAS)*, an extension of the Contractual Agent Society (CAS) idea. Essentially, a FAS is a distributed information system modeling an agent society, providing agents with the ability to collaborate in order to meet certain common goals. In a FAS, unlike the CAS, the agents themselves have control over the workflow processes and multi-agent conversations that they need to execute in order to meet their common goals.

Keywords: Agents, adaptive workflow, agent-oriented workflow, agent societies

I. INTRODUCTION

Recent work on agents (an excellent survey of the latest work is provided by Griss [2000]) has evoked much interest in multi-agent systems. The logical extension of multi-agent systems is "agent societies"—groupings of agents that come together to collaborate to meet certain common goals [Dellarocas 2000; Dellarocas and Klein 1999; Minar et al. 2001]. Another type of agent society is an "e-services ecosystem," exemplified by the development and deployment of e-Speak [**http://www.e-speak.net/**].

In this paper, we investigate the architectural underpinnings of agent societies. Inspired by the Contractual Agent Society (CAS) approach [Dellarocas 2000; Dellarocas and Klein 1999], we have cast this approach in our work on agent-oriented adaptive workflow [Narendra 1999, 2000, 2001a, 2001b, 2001c, 2001d]. Our work provides a sound architectural foundation on which to build a *Flexible Agent Society (FAS)*.

In particular, we discuss how security, specifically Role-Based Access Control (RBAC), is implemented in our FAS; our unique contribution is in specifying how RBAC can be implemented for adaptive and distributed workflows. The other major contribution of this paper is in specifying how the transactional aspects of distributed workflow adaptivity are managed in our FAS.

This paper is arranged as follows. In section II, definitions of the basic concepts used are presented. The concept of agent societies and the FAS concept are described in section III. In Section IV, we discuss how RBAC is implemented in our FAS. Section V describes how workflow adaptivity and transactionality are managed in the FAS. The conclusions and suggestions for future work are presented in Section VI.

II. PRELIMINARIES

WORKFLOW

Our basic workflow model is based on the graph-theoretic notions in the OpenPM process model [Shan et al. 1997]. We assume that each workflow instance is a directed graph W = <N,E>, where N is the set of nodes and E is the set of edges. Each edge e in E is a tuple of the form $<n_{begin}, n_{end}>$, where the edge is directed from n_{begin} to n_{end}.

We first define two unique nodes: START and END nodes. For each workflow, there will be only one of each kind. A START node has no predecessor, and an END node has no successor.

The workflow graph is assumed to obey two simple conditions:

- Every node in the graph is reachable from the START node (i.e., there is a path from the START node to every other node in the graph).

- The END node is reachable from every node in the graph (i.e., there is a path from the node to the END node).

Nodes are of two kinds: – *work nodes* and *route nodes*. Work nodes are nodes that actually perform the activities in the workflow. Route nodes are nodes whose only function is to evaluate rules (i.e., boolean conditions) and, depending on the result of the evaluation, direct the flow of control to specific work nodes or route nodes. The rules are represented in Event-Condition-Action format (also known as ECA) [Kappel et al. 1998]; these rules prescribe certain actions to be performed upon receiving an event that obeys certain conditions.

For each work node, we assume the existence of specific data structures that capture the data flow information with the node. In other words, each node is assumed to perform certain operations on data and artifacts (e.g., complete a form, collate and summarize data, etc.). These operations, together with the data flow, define the flow of control throughout the workflow; this is a consequence of the workflow design.

Edges are of four kinds: *forward edge, loop edge, soft-sync edge, and strict-sync edge*. Forward edges are needed to depict the normal workflow execution, which is in a forward direction. Loop edges are backward pointing edges that are used to depict the repeated execution of loops.

The sync edges are quite different. These edges are used to support synchronizations of tasks from different parallel branches of a loop. There are two types:

- A soft-sync edge is used to signify a delay dependency between two nodes n_1 and n_2 (i.e., n_2 can only be executed if n_1 is either completed or cannot be triggered any more). This type of synchronization does not require the successful completion of n_1.

- A strict-sync edge between n_1 and n_2 requires that n_1 successfully complete before n_2 executes.

  Clearly, the use of such edges must satisfy some conditions:

- Redundant control flow dependencies between nodes and loops should be avoided.

- A sync edge may not connect a node from inside a loop body with a node not contained within that loop.

An example workflow graph illustrating all of these definitions is given in Figure 1. The arcs depict the flow of the data and artifacts from the START to the END node.

We can also define states for each node, depending on where it is during the workflow execution. Hence we define the following states for nodes: NOT-ACTIVATED, ACTIVATED, DONE, FAILED, SUSPENDED.

The definition of these states naturally leads to the conclusion that we can define the state transition diagram that each node has to obey. Indeed, this means that the state of every node in the workflow instance is governed by the transition rules in the state transition diagram. This diagram is given in Figure 2.

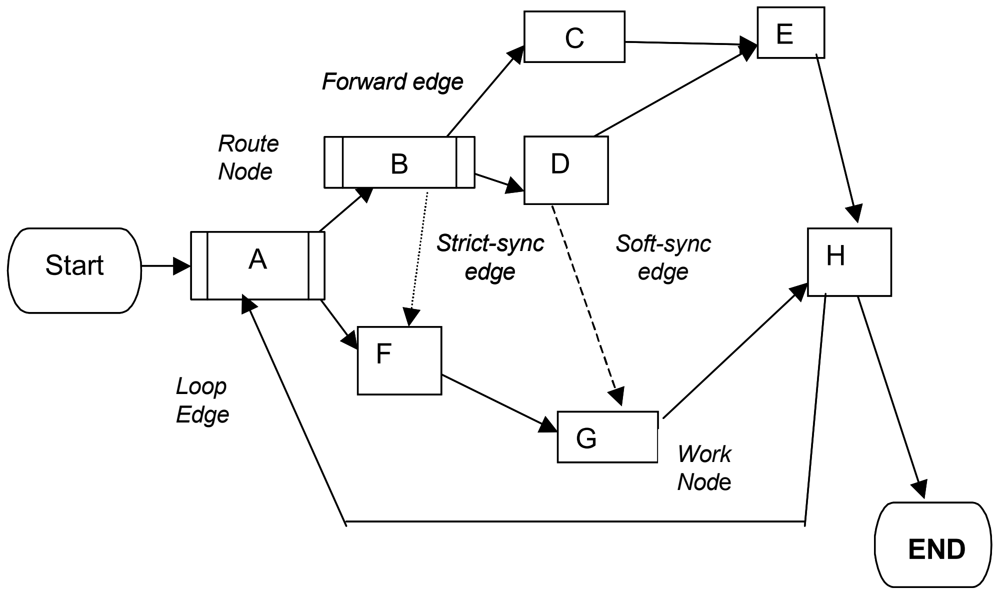

Figure 1. Workflow Graph

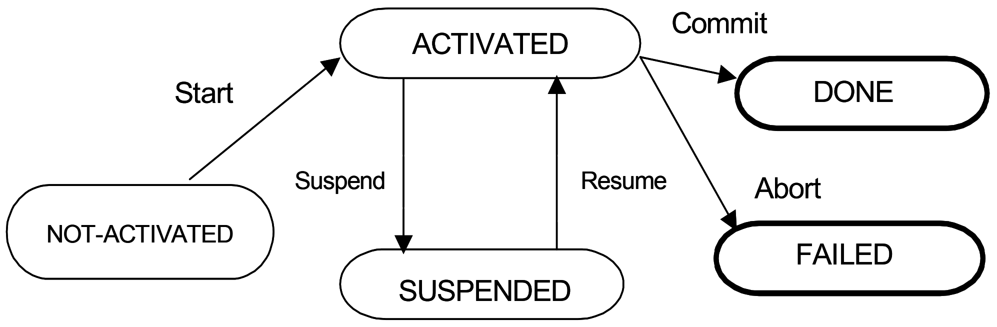

Figure 2. State Transition Diagram

ADAPTIVE WORKFLOW

A three-tier approach to adaptive workflow management was developed in earlier work [see Narendra 2000]. Essentially, we recognized that a workflow needs to have a schema (i.e., a basic definition of its structure) before it can be instantiated and executed. Moreover, since workflow is typically executed in business organizations, it is essential that it meet certain business goals. Therefore, we see that business goals (essentially arising out of **planning**) lead to workflow **schema** being defined, out of which a workflow **instance** is generated for execution. (Later in this section, in the "Adaptive and Flexible Workflow Example" sub-section, an example illustrating all of these ideas is presented.)

Workflow adaptivity has been identified as basically one of three types:

- **Adaptivity at instance level**: here, only the workflow instances need to be modified, perhaps to make them more efficient, or to make them easier to execute. The modules that implement this are:
  - ➢ *Basic Workflow Model*: stores the instances of our workflow model
  - ➢ *Workflow Change Model*: stores the constructs for specifying dynamic workflow changes
  - ➢ *Workflow Change Verifier*: performs the syntactic and semantic checking for workflow adaptation
  - ➢ *Workflow-Schema Interface Manager*: interface module that interacts with the Schema Layer

- **Adaptivity at schema level:** here, the workflow schema will need to be modified. This would arise if certain "ways of doing things" (e.g., developing a product in-house instead of outsourcing the development) change, and will necessitate radical modifications to all the workflow instances of the schema in question. The modules that implement this are:
  - ➢ *Schema Handler*: creates, versions and stores the workflow schema
  - ➢ *Schema Migration Manag*er: handles migration of workflow instances between schemas and interfaces with the Schema Handler
  - ➢ *Schema-Planning Interface Manager*: interface module that interacts with the Planning Layer

- **Adaptivity at planning/goal level:** here, the *goals* may have to be modified in response to changing environmental conditions. This could cause radical changes in the schema (e.g., a change in the outsourcing goals of an organization could result in “going back to the drawing board” in order to redefine all its workflow schema), which could cause disruptive changes in the workflow instances. The modules that implement this are:

- ➢ *Organizational Workflow Repository*: organization-wide repository of all workflow processes stored in the system; stores the workflow schemas and their instances, and also stores the goals that led to the creation of the workflow in the first place.
- ➢ *Goal Specification Module*: through this module, the workflow administrator first enters the goals that the workflow has to satisfy. He/she can then look into the *Organizational Workflow Repository* for reusing any workflows from the past that can satisfy—either fully or partially—the goals that he/she has specified.
- ➢ *Workflow Design Module*: with this module, and with the goal and workflow information from the *Goal Specification Module* and the *Organizational Workflow Repository*, the workflow administrator can design the workflow schema. He/she will then enter it into the *Organizational Workflow Repository*. The schema will then be transferred into the *Schema Handler* via the *Schema-Planning Interface Manager*, where it will be appropriately versioned and stored. The appropriate version number is then communicated to the user.

  Essentially, the Workflow Design Module is the main module, where the workflow is defined and adapted by the workflow administrator. This module presents a user interface through which the workflow administrator can adapt the workflow by specifying the adaptation as a graph transformation (explained next).

The architecture is depicted in Figure 3.

ADAPTIVE WORKFLOW MECHANISM

Any change to be made on-the-fly to an already executing workflow has to proceed as specified in the following steps:

- The proposed change has to be specified in terms of a graph transformation (see below for details).
- The graph transformation is verified using graph queries in the system; if the transformation is not feasible, the user is notified, and the unsatisfied portions of the query are also transmitted to the user.
- If the transformation is feasible, the user is informed and the query is executed.

Our algorithms follow.

Graph Transformations

If a change is needed to the workflow definition on the fly, this will have to be specified by the user to the system via graph transformations. Each graph transformation obeys the following sequence:

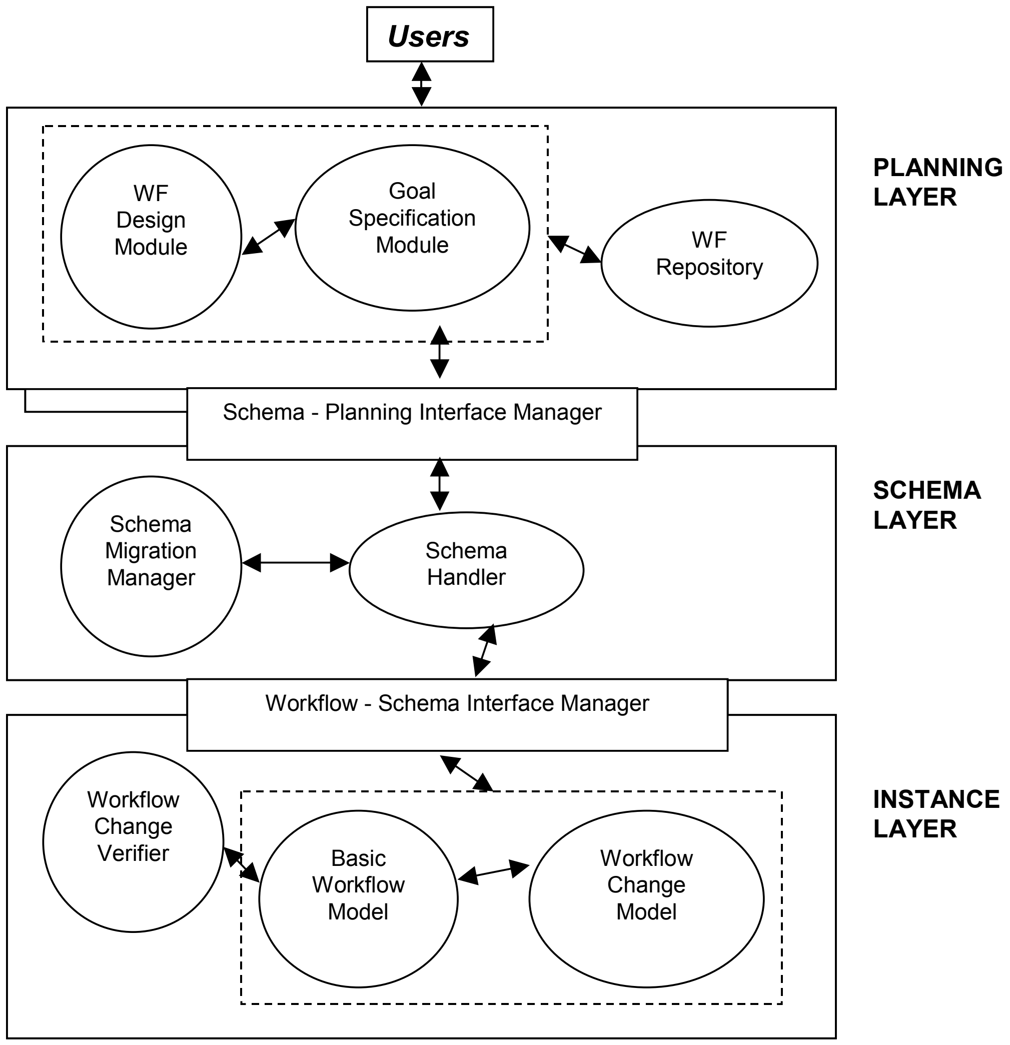

Figure 3. Three-Tier Adaptive Workflow Architecture

- Specify the sub-graph which needs to be transformed: this will involve isolating that part of the workflow, by specifying the beginning and ending nodes of the sub-graph

- Specify what the sub-graph will be after the transformation: here, the user will have to specify the new workflow sub-graph that will substitute for the old sub-graph

- Specify the basic operations needed in terms of node addition/deletion and edge addition/deletion that define the transformation: in order to get from the old sub-graph to the new sub-graph, the user will have to specify the order in which the transformation, which is essentially a combination of node and edge additions/deletions, will have to take place

Graph Queries

The graph transformation provided by the user triggers certain queries within the graph-oriented internal representation of the workflow model. The queries that are executed by the system are:

- Checking syntactic correctness and consistency of the transformation (using the graph-theoretic properties of the workflow model), including:
  - ➢ All input parameters of a node are supplied before the node is executed
  - ➢ Nodes from different parallel branches of a loop are not allowed write access to the same data element unless they are synchronized by a sync edge

- Checking semantic correctness and consistency of the transformation using the ECA-type rules provided in the route nodes

If the queries are executed successfully, the user is notified and the appropriate changes are made to the workflow definition. If not, the user is informed as to which queries failed.

Changing the Workflow Schema

Changing the workflow instance on the fly naturally raises the question of whether—and if so, when—to migrate the new instance onto the definition, so that the workflow schema itself changes. There are two options here:

- Modify the workflow schema as soon as the new workflow instance is created.

- Mark the new workflow instance as a new version of the old workflow schema, thereby creating *version trees* of workflow schemas and their associated instances (as described in Kradolfer and Geppert [1999]).

For example, if a workflow instance I of a schema S is adapted to I', then one of two things can happen:

- I' is "rolled into" S, thereby changing S; that is, the same changes that change I to I' are also effected in S, so that S becomes the schema for I'

' I' is stored as an instance of S-version2, where S is stored as S-version1

TYPICAL USAGE SCENARIO

In order to illustrate how our architecture is used, we sketch a typical usage scenario. We do this by enumerating the possible reasons for a workflow administrator to adapt the workflow.

(a) *Changes in goals*: the goals underlying the currently existing workflow processes may have changed, necessitating a major change in the workflow schema (for example, the business has to be substantially reengineered and repositioned, or the product line itself may undergo a major change). Perhaps the current workflow schema may have to be discarded and a new one created, along with its associated workflow instances. This will have to be done at the Planning Layer.

(b) *No changes in goals, but changes in business relationships and environment*: this will necessitate a change in the workflow schema, since the business models governing the workflow may have to change, although the overall business goals may remain unchanged (for example, the workflow may need to be implemented in a distributed fashion, by using sub-contractors, necessitating a change in the workflow schema). This will have to be done at the Schema Layer.

(c) *No changes in either goals or business environment, but the need to improve delivery efficiency or the need to introduce alternative processes (one such example is sometimes referred to in the literature as "exception handling")*: this will necessitate a change in the workflow instances only. This can be restricted to the Instance Layer itself.

Hence we can look at the following typical scenario:

- The workflow administrator perceives a situation in the organization where there is a need to change some business processes. He/she then determines which of the situations described above apply.

- *If (c) applies:* In this case, only an instance of the workflow schema is being modified. This may essentially be a one-time exception in order to deal with a special case that was not thought of earlier. After the workflow process has completed executing, the workflow administrator can then consider migrating the workflow instance to the new workflow schema version and discard the old version, should he/she so choose, as described. This can be done following the rules as described by Kradolfer and Geppert.

- *If (b) applies:* This could have a serious impact on the business, since business relationships may themselves have to be rewritten. This will require evolving a new workflow schema, and ensuring that all future workflow instances are derived from this schema and not the older one. Again, as described above, migration of the existing workflow instances will have to be decided as described by Kradolfer and Geppert.

- *If (a) applies:* Such a situation would arise if the business itself is being reengineered or repositioned. In such a case, the most likely response would be to complete all of the currently running workflow instances (depending on whether they can be completed; - this will have to be a business decision) and starting the workflow design from scratch for future activities to define new goals, workflow schema, and instances.

FLEXIBLE WORKFLOW

Traditionally, workflow management has been regarded as being centralized, monolithic, rigid, and not easily amenable to adaptation. Once once a workflow process was defined, it was regarded as more or less "cast in stone," to be executed with little change. The advent of adaptive workflow [Narendra 2001c; Reichert and Dadam 1999] has eased this situation somewhat, but even adaptive workflow systems are designed with centralized control in mind. However, this does not meet the needs of most information systems today, where workflow participants need more flexibility in defining and executing workflows. To that end, we extended the OpenWater idea from Whittingham [1999] and developed an architecture for what we called "flexible workflow support" [Narendra 2001c], containing the following ideas:

- Any workflow can be classified to belong to one of the following three "control levels": loose, medium or tight.
  - ➢ Loose workflows can be defined and started by any user (not necessarily the Workflow Administrator) in collaboration with the other participants in the workflow. Such a workflow can adapt itself as much as possible, depending on the need and on the participants.
  - ➢ Medium workflows are also started by any user, but need approval from the Workflow Administrator in order to ensure that they adhere to a particular workflow schema. This approval is also needed if either the workflow instance or schema is adapted.
  - ➢ Tight workflows are the traditional workflows, which are defined and modified centrally by the Workflow Administrator.

- Our extensions to Figure 3 to accommodate flexible workflow support are the following (see Figure 4 for the modified architecture of the Planning Layer, which supports flexible workflow):
  - ➢ Logically separate the Workflow Repository for representing and storing these three types of workflows.
  - ➢ Have a Schema Discovery Module as part of the Goal Specification Module, which will assist the Workflow Administrator in observing the execution of several instances of a loose workflow and then upgrade it to medium level by designing a schema for it; all future executions of this process will then have to be at medium level, adhering to the schema.
  - ➢ Similarly, the Workflow Administrator can, using the Goal Specification Module, also observe the execution of several instances of medium processes and upgrade them to the tight level; the assumption being that processes that are executed a sufficient number of times can be better controlled centrally. Hence the Schema Discovery Module can assist the Workflow Administrator in developing schemas for loose workflows that are deemed to be critical to the organization, and hence need to be controlled centrally.
  - ➢ Synchronization and interaction between different workflow processes can also be at the three control levels, and would depend on the control levels of the individual workflows (i.e., they can be either peer-to-peer or centralized interaction).

- Extending this to the distributed workflow case means that there are two categories of workflows, which can be at different control levels—the overall workflow and its constituent workflows (which will be executed at different workflow servers)—resulting in nine different combinations for the three control levels. Essentially they are the following:
  - ➢ If the overall workflow process is at the loose level, this means that it is defined among the workflow servers in a peer-to-peer fashion. However, each constituent sub-workflow can be at any of the three levels, and managing the constituent workflows becomes an internal matter for the individual workflow servers (which could belong to different organizations).

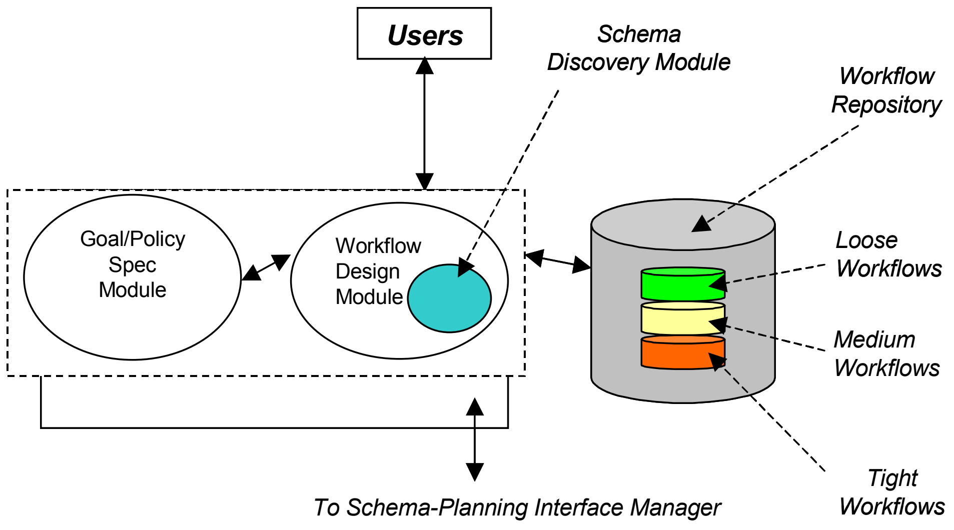

Figure 4. Flexible Workflow Architecture—Planning Layer

➢ If the overall workflow process is at the medium level, this will need a Central Coordinator (CC) (similar to that of a Workflow Administrator in a single organization; distributed workflows are discussed in more detail in section III) that will need to approve the workflow before it is executed. The individual workflow processes can be at any control level; however, the CC will again ensure—from the viewpoint of the overall workflow process—that the individual workflow processes are represented as black boxes with predefined inputs and outputs, which the individual workflow servers will need to satisfy. The individual workflow process can be at any of the three control levels.

➢ If the overall workflow process is at the tight level, then the CC itself is in charge of defining and imposing the overall workflow process on the individual workflow servers. Here also, the individual workflow processes are specified as black boxes with predefined inputs and outputs, and it will be the responsibility of the individual workflow servers to design workflow processes to satisfy the inputs and outputs. The individual workflow process can be at any of the three control levels.

  ❖ Please note that the black box approach towards defining sub-workflows serves to preserve the autonomy of the respective individual organizations so that they can define their sub-workflows as convenient; however, the sub-workflows need to satisfy the appropriate "process interfaces," i.e., the aforementioned inputs and outputs.

There are two other types of distributed workflows [see Hollingsworth 1995]:

- Chained: this type allows a task anywhere in workflow A to connect to another task anywhere in workflow B. This allows only the transfer of a single item of work between the two workflow environments, which then operates independently in the second environment with no further synchronization.

- Parallel synchronized: this allows two workflows in two different workflow servers to operate independently, but requires that synchronization points exist between them (typically at route nodes) at prespecified intervals. For example, workflow processes for components of a product being developed at different organizations may need to be synchronized at regular intervals, so that the development and delivery of the components can be "in sync" with each other.

Hence the workflows A and B can be treated as two distributed workflows, and the same logic for distributed workflows (i.e., the nine cases) will apply here.

Hence the CC in this case is also an adaptive workflow system just as the one described in section II and depicted in Figure 3, with the important difference that it supports and manages the overall distributed workflow process, with the constituent workflows being managed by (perhaps separate) individual workflow servers.

Administration of Flexible Workflows

From the description of flexible workflow given above, it is clear that the workflow administrator (or CC, as the case may be) needs a process lifecycle to manage all of the workflow processes effectively. This lifecycle needs to take into account workflow processes at the three different control levels. Hence our process administration lifecycle has the following steps for any workflow process:

(a) *Process creation*: as described above, depending on the control level:
- Creation of loose workflows—by the user directly, who can define the process, and start executing it.
- Creation of medium workflows—here, the created process is derived from an already existing schema and executed. As long as the process conforms to the schema, which needs to be checked by the administrator—, t can be started and executed by the users.
- Creation of tight workflows—the traditional workflow creation process, performed directly by the administrator.

(b) *Process Synchronization and Interaction*: again, as already described, this would depend on the control levels of the workflows. This is essentially the nine different cases of inter-workflow interaction already described (see "Flexible Workflow"); chained and parallel synchronized workflows will also be synchronized here.

(c) *Process maintenance*: this is an ongoing activity that involves the following:
- For loose processes, this would mean workflow instance monitoring, schema discovery, and history management
- For medium and tight processes, this would mean schema monitoring, history management, and adaptivity management as per the three-tier approach

(d) *Process completion and archiving*: here also, history management is needed for storing the results of process execution.

(e) *Process upgrade*: if a loose workflow has been implemented a sufficient number of times (how much is sufficient is a business decision for the Workflow Administrator), then there may be a need to upgrade it to medium level by determining its schema from the various process instances. The Workflow Administrator may decide that this workflow has "matured" sufficiently enough so that it can be better controlled centrally. For medium level processes, this can also involve upgrading to tight level by linking the schemas to the process goals. The idea behind this upgrade is that the workflow is now sufficiently well understood and can be more tightly controlled by the Workflow Administrator or CC.

ADAPTIVE AND FLEXIBLE WORKFLOW EXAMPLE

In this section, we reproduce the simple yet non-trivial example for flexible workflow described in Narendra [2001c]. This example will not only illustrate flexible workflow, it will also illustrate how instance, schema, and (to a limited extent) planning level workflow adaptation can take place.

This example is derived from the personal experiences of the author and his relatives at an insurance company (specifically in the case of processing motor vehicle accident insurance claims). Let us assume this insurance company has to process several insurance claims. If the insurance claims workflow is regarded as a loose workflow, there could be several ways in which it is defined and executed. Let us assume that four different people in the insurance company define it in the following four ways, starting with **Instance #1**:

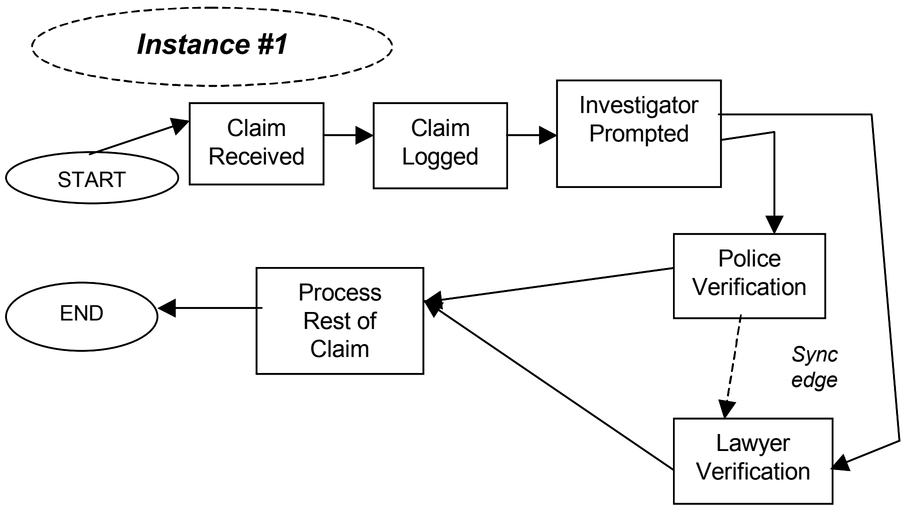

Figure 5. Instance #1

Instance #1 is, therefore, a simplistic way in which the insurance claim can be processed. This brings us to **Instance #2**:

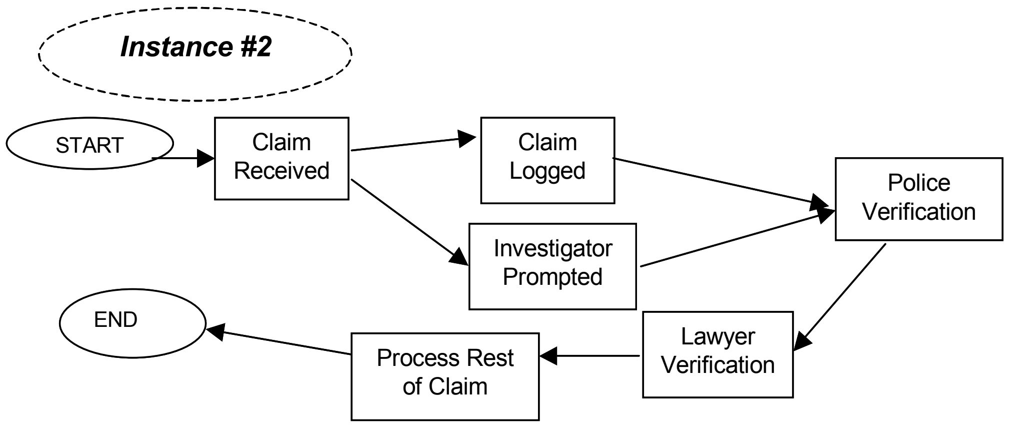

Figure 6. Instance #2

In **Instance #2**, on the other hand, the logging of the claim and the prompting of the insurance investigator are done in parallel, presumably to save time, whereas Police and Lawyer verification are done sequentially. We now come to **Instance #3**:

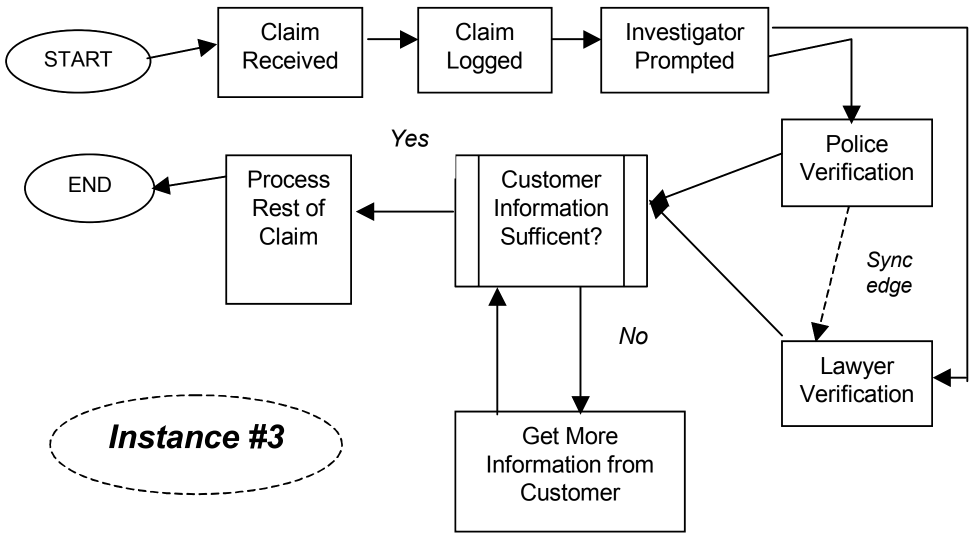

Figure 7. Instance #3

In **Instance #3**, the investigator feels the need to check whether the information provided by the customer is sufficient and accurate, introducing extra processing steps in the workflow.

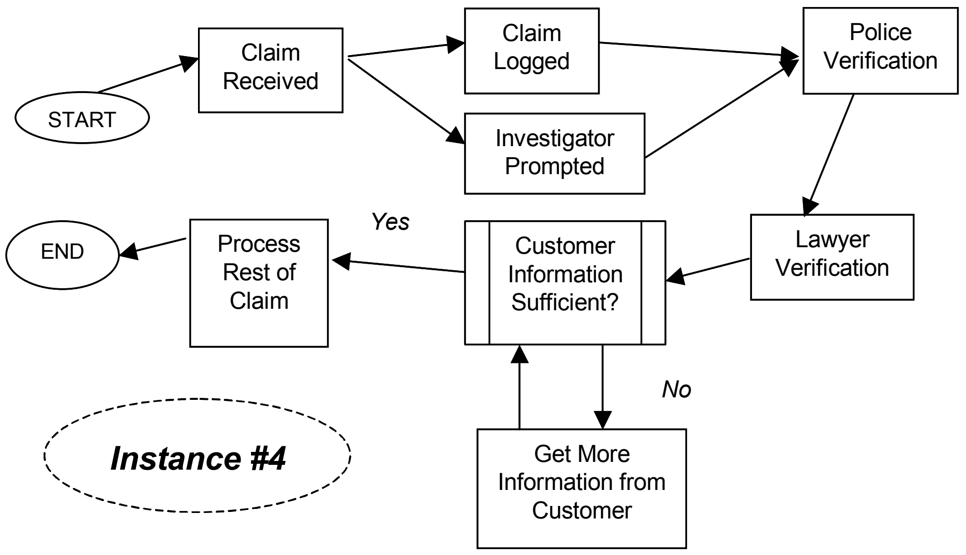

Figure 8. Instance #4

Instance #4 is an improvisation of Instance #3, in that some amount of parallelism is achieved (as in **Instance #2**).

Hence, depending on the number and business utility of these disparate workflow executions, the workflow administrator may decide to upgrade the workflow to medium level by developing the following **schema** for it (here, it happens to be the same as **Instance #3**; in the general case, it could be a combination of one or more instances):

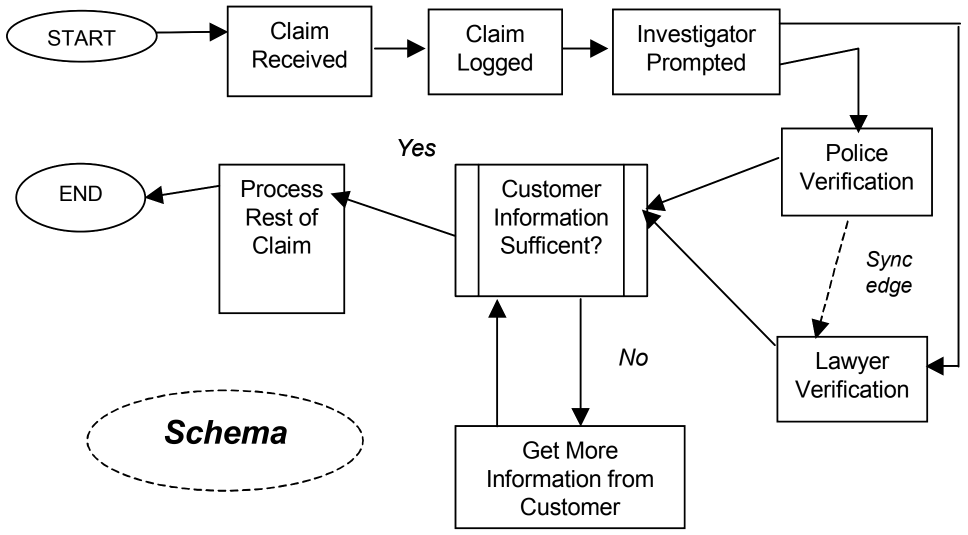

Figure 9. Schema

The overall structure of this schema could be described as follows:

- The claim should first be logged into the computer before it is sent for police and lawyer verification.
- The information should be checked for sufficiency before further processing.
- The "Get More Information from Customer" task could be a (probably complex) sub-workflow in itself, and may need to be outsourced to another organization. If that happens, then our overall workflow would become a distributed workflow, and the "Get More Information from Customer" sub-workflow would again have to be defined and executed in a (probably) separate workflow server in another organization.
- The "Police Verification" and "Lawyer Verification" tasks could also be "bundled" into a separate sub-workflow, and outsourced to another organization.

Hence, once the workflow is upgraded to medium level, any employee of the insurance company can define a new instance of this workflow, as long as it adheres to the schema by obeying its overall structure. The workflow administrator can check whether this adherence is achieved before approving the workflow instance.

Later on, the workflow administrator can upgrade the workflow to the tight level by imposing the following constraints on the workflow participants:

- All insurance claims to be completely processed within two weeks.
- Complete information to be provided to the customer about the decision (accept or reject) regarding the insurance claim.

Sync Edges

This example would not be complete without a short note about the sync edges shown in **Instance #1**, **Instance #3**, and the Schema. The sync edge, which connects the Police Verification task to the Lawyer Verification task, could be either soft-sync or strict-sync:

- If it is soft-sync, then the Lawyer Verification task is supposed to start only after Police Verification starts.
- If it is strict-sync, then the Lawyer Verification task is supposed to start only after Police Verification ends successfully.

Hence the appropriate sync edge would be used, depending on the business need.

SECURITY AND ROLE-BASED ACCESS CONTROL (RBAC) IN WORKFLOW

The most well-known approach to security management has been the Role-Based Access Control (RBAC) model [Sandhu and Samarati, 1994]. This model allocates access rights to users based on the roles that they perform in the organization. Roles can also be organized in a hierarchy, typically a partial order. In this hierarchy, roles at a higher level are assumed to "inherit" the access rights of roles at lower levels in the hierarchy, in addition to the access rights they already possess.

In addition to this hierarchy, constraints can also be defined on how these access rights are granted to particular roles. Some of the most common constraints are separation of duty constraints, which are of two types:

- Static Separation of Duty (SSOD): these constraints impose static restrictions such as "the role R cannot have access rights A1 and A2 simultaneously."

- Dynamic Separation of Duty (DSOD): these constraints impose dynamic restrictions such as "if role R1 is executing a task T1, then R1 cannot execute task T2."

In other words, SSOD constraints can be evaluated without executing the workflow, whereas DSOD constraints evaluation can only be performed at run time. Hence, in Sandhu and Samarati, the RBAC model has itself been modeled as a partial order lattice as depicted in Figure 10.

- RBAC0—the basic RBAC model
- RBAC1—RBAC0 with role hierarchies
- RBAC2—RBAC0 with constraints
- RBAC3—combination of all of the above

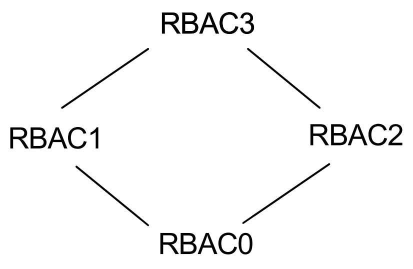

Figure 10. RBAC Lattice

An RBAC3-compliant version of RBAC is presented in Nyanchama and Osborn [1999], which also presents a three-tier security model, consisting of user-group assignments, user-role assignments, and role-privilege assignments. The unique features of this model are the following:

- The notion of "direct privileges" (privileges that are owned solely by the role) and "inherited privileges" (privileges owned by roles that are junior to the role in question).

- Algorithms for role management (i.e., role addition and role deletion) and addition/deletion/ modification of privileges associated with a role.

Privileges, which are also synonymous with the term “access rights” in the security literature, are essentially authorizations or powers given to individuals in organizations based on their role and position in the organization. For example, a Vice President (VP1 and VP2 in Figure 11) role could be authorized to approve purchase requests up to $200,000 at a time, whereas a Project Leader (L1 through L4 in Figure 11) could be authorized to approve only up to $10,000 at a time.

Several conflicts can arise in this model:

- *Role-role conflicts*: assigning two conflicting roles to a user; for example, a person cannot simultaneously perform the roles of approving and implementing a payment request process, since this could potentially cause a “conflict of interest” situation.

- *Privilege-privilege conflicts :* assigning two conflicting privileges to a role.

- *User-role assignment conflicts :* assigning a role to a user who is not allowed to play that role.

- *Role-privilege assignment conflicts :* assigning a privilege to a role which is not allowed to access that privilege.

A sample Role Graph is presented in Figure 11. Every Role Graph has two "dummy" roles, MaxRole and MinRole, that are the senior and junior, respectively, of all roles. The numbers next to each role are the direct privileges of that role. For example, for VP1, the direct privileges are {9,10}, whereas the inherited privileges are {1,2,3,4,5,6}, which consists of the direct privileges of all roles reporting (directly or indirectly) to VP1.

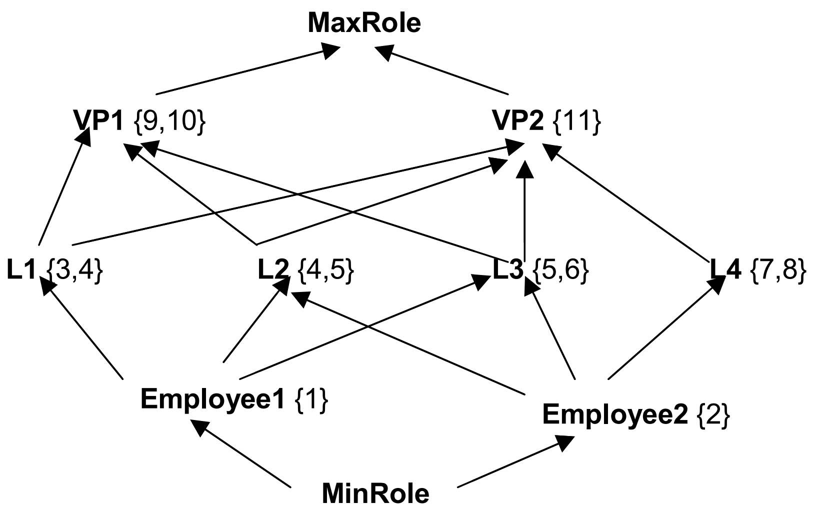

Figure 11. Role Graph

The following algorithms for Role management are provided in Nyanchama and Osborn:

- *Role Addition with only direct privileges*: a role is added to the Role Graph along with only its direct privileges.
- *Role Addition with inherited privileges*: a role is added to the Role Graph along with its direct and inherited privileges.
- *Role Deletion*: a role is deleted from the Role Graph; this would also involve either deletion or re-allocation of the direct privileges associated with the Role.
- *Privilege Addition*: a privilege is added to a Role.
- *Privilege Deletion*: a privilege is deleted from a Role.
- *Edge Insertion*: a new reporting relationship between a Role and one of its Senior Roles is added (i.e., a person playing the Role is made to report to a person playing the Senior Role in the organization). Hence the Senior person will inherit all the privileges available to the Junior person.
- *Edge Deletion*: a reporting relationship between a Role and one of its Senior Roles is deleted (i.e., a person playing the Role no longer reports to any person playing the Senior Role in the organization).

All Addition algorithms also contain steps for checking for the four types of conflicts described above, and abort with an error message when a conflict is detected. The algorithms also incorporate mechanisms for detecting cycles in the Role Graph, in which case they abort with an error message.

Thomas and Sanders [1997] present a security model for workflow called Task Based Authorization Control (TBAC). This model specifies, in a manner similar to that of RBAC, the tasks that each role is authorized to perform. A lattice structure similar to that of RBAC (i.e., TBAC0 through TBAC3) is also presented. This is taken further by Cholewka et al. [2000], who present a mapping between RBAC and a similar TBAC-like model. In essence, TBAC works by setting users' access rights either at workflow definition time, or at workflow execution time before the workflow task is executed.

SECURITY IN ADAPTIVE WORKFLOW

As we have seen above, most of the existing secure workflow models do not take into account the fact that workflow can be adaptive. Since all workflows are assumed to be highly adaptive, we have developed a three-tier RBAC-based security infrastructure [Narendra 2001d] that mirrors our three-tier adaptive workflow architecture. Thus:

- At instance level, role-task assignments will be modified.
- At schema level, user-role assignments will be modified.
- At planning level, global SSOD and DSOD constraints will be modified.

The assumption behind this infrastructure is that, in most organizations, user-role allocation is considered to have more impact on the organization than role-task allocation. For example, making a person a Vice President could significantly enhance his/her payment approval privilege and subsequently impact on the organization, whereas adding a task to a role would only enhance the significance of that one role. Of course, this may be an over-simplification in many cases, but this model provides one way to effectively manage security-related inconsistencies that could arise due to workflow changes.

A brief sketch of the algorithm follows [the algorithm is given in detail in Narendra 2001d]:

- Assume an initial role-task and user-role allocation in a workflow, and the existence of a Role Graph for the workflow as depicted in Figure 11.

- At the most elementary level, any workflow adaptation is a combination of either a task deletion or a task addition:
  - ➢ If a task is deleted, and if other tasks are "brought forward" in order to take advantage of the task deletion, then all of these tasks belong to the Modification Region (MR) (i.e., those tasks that are affected by the workflow change).
  - ➢ If a task is added, this could affect the tasks following the added task. Also, any other tasks that have to be performed in parallel with the added task may be affected if these tasks have to be executed by the same user(s) who are authorized (as per the Role Graph) to execute the added task. Hence these tasks also become part of the MR.

- For each task in the MR, the following steps are executed:
  - ➢ If the task can be done by the currently assigned user, then the search ends, and we move on to the next task in the MR.
  - ➢ If **not**, then we move up the Role Graph, and check whether any senior of the assigned user can execute the task. If so, the search ends and we move on to the next task in the MR. If **not**, then we check among the juniors of each of the already checked seniors of the user in question. This continues recursively upward in the Role Graph, starting at each of the seniors of the user in question, until the MaxRole role is reached. Of course, any time an available user is found, the search ends.
  - ➢ If still **not**, then we check among the juniors of the user in the Role Graph. This is basically a "mirror image" of the search among the seniors, thus: at each junior, the seniors of the junior are also checked, and this also proceeds recursively down the Role Graph, until the MinRole role is reached.
  - ➢ Also note the following:
    1. At every step of both the senior and junior searches, we pay attention to the SSOD and DSOD constraints. If a suitable candidate is found such that no global SSOD or DSOD constraint is violated, then that candidate is assigned to the task. If, after the entire search has been completed and no suitable candidate is found, then the algorithm aborts with an error message to the Workflow Administrator.
    2. Whenever a potential user is checked and rejected, that user is automatically flagged. Later in the search, when searching among either juniors of seniors, or seniors of juniors, this flag helps in eliminating the need to search the same user again.

If the workflow change is supposed to be at schema level, then the modified allocation is made permanent. For a workflow change at instance level, this modification is temporary and applies only to that particular instance.

AGENTS

Currently, there is not much consensus on what an agent is, and many definitions abound. For our purposes, we will combine the following definitions:

- MASIF [1998]: "An agent is a computer program that acts autonomously on behalf of a person or organization."

- Griss [2000]: "an autonomous software component that interacts with its environment and with other agents."

Hence we define an agent as "a software component that acts autonomously on behalf of a person or organization, and is also able to interact with its environment and with other agents."

Agents can be characterized as "weak" or "strong" agents [Wooldridge and Jennings 1995]. Weak agents possess the following characteristics: *autonomous* (having goals and plans for achieving them), *social* (can interact with other agents and their environment), *reactive* (can perceive their environment and respond to changes that occur), and *pro-active* (affect their environment rather than passively allowing their environment to affect them). In addition to the above characteristics, strong agents also possess the following characteristics: *mentalistic notions* (have beliefs, desires, and intentions), *rationality* (can reason about their actions and perform actions which further their goals in line with their beliefs, desires and intentions), *learning* (have the ability to learn from their actions and their environment and other agents). In this paper, we restrict our attention to weak agents, since it is sufficient for our FAS that the agents be autonomous, social, pro-active, and reactive.

A seven-axis characterization of agents can be found in Griss and has the following dimensions: adaptability, autonomy, collaboration, intelligence, mobility, persistence, and personality/sociability. For our purposes, we require our agents to have high adaptability, autonomy, collaboration and persistence, since these are essential requirements for agents in our FAS. The other three axes, i.e., intelligence, mobility, and personality/sociability, are not applicable for us, since we are concerned with weak agents.

The Foundation for Intelligent Physical Agents (FIPA) has been developing standards for agents and multi-agent systems. Their reference architecture for agent platforms is accessible from **http://www.fipa.org/specifications/index.html**.

There are many interesting linkages between workflow and agents, which we will be exploiting in this paper [Griss 2000]:

- *Agents can collaborate to perform a workflow,* e.g., telecom provisioning, service provisioning, scheduling.

- *Agents can be used to make workflow more intelligent,* e.g., by adding negotiation, reasoning, or decision points.

- *Workflow can be used to coordinate a set of agents,* e.g., application management.

- *Workflow can be used to coordinate interaction between people and agents, having agents delegate to people or other agents*, e.g., a telecom management system alerting a human operator, or assigning a repair or provisioning engineer.

For our purposes, we recognize that the first two linkages represent *agent-enhanced workflow* (using agents to enhance workflow systems, i.e., agents representing workflow systems) and the last two represent *agent conversations* (i.e., using workflow concepts to model agent interactions in multi-agent systems). Agent-enhanced workflow will be used to define what we will call "macro-workflow" and agent conversations will be used to model what will be termed micro-workflow. Hence macro-workflow will model adaptive workflows, whereas micro-workflow will model multi-agent interactions.

MULTI-AGENT INTERACTIONS

Distributed Flexible Workflow and Multi-Agent Interactions were integrated into our three-tier in earlier work [Narendra 2001a], called *AdaptAgent*. Its its salient features are:

- Distributed flexible workflow is used for executing the distributed business processes.
- Agent technology is used to provide the necessary automation of decision-making tasks that influence the execution of the business processes.
- The multi-agent interactions/conversations (i.e., micro-workflows) are modeled as single tasks within the flexible workflows (i.e., macro-workflows) with predefined entry and exit criteria. Within a micro-workflow, its related macro-workflows are represented within these entry and exit criteria. Hence *AdaptAgent* supports two types of distributed workflows, with the following characteristics:
  - ➢ A micro-workflow is defined after its respective macro-workflow has been created. As explained above, the micro-workflow is a conversation conducted by the agent executing the macro-workflow jointly with the other agents involved in the conversation. The conversation is started after the predecessor of the task representing the micro-workflow has successfully completed. After the conversation is completed, the agent will then resume its macro-workflow execution from the successor of the task representing the micro-workflow. This will be the same for every other agent participating in the conversation. Naturally, the outcome of the conversation could determine the course of the future macro-workflow executions of the agents, as per the macro-workflow definitions.
  - ➢ Hence, macro- and micro-workflow processes can be adapted relatively independently of each other; however, when a macro- (resp. micro-) workflow is adapted, the effect of the adaptation on its related micro- (resp. macro-) workflows will need to be considered before going ahead with the change. This is done as follows:
    - ❖ If the change is to a macro-workflow instance or schema, then the affected micro-workflow instance or schema will be syntactically and semantically checked. If, as a result of this, a change in the micro-workflow schema is needed, then it must also be adapted as per the procedures described earlier.

      For example, a change in the manufacturing workflow (which is a macro-workflow) of an automobile component among a prime contractor and its sub-contractors could trigger changes in the multi-agent negotiations (i.e., micro-workflow) needed for the procurement process.
    - ❖ If the change is to the micro-workflow instance or schema, then if there is a corresponding change in its input or output to any of its related macro-workflows, the macro-workflow may also need to be modified as per the procedures described earlier.

      For example, a change in the multi-agent negotiations (micro-workflow) among the aforementioned prime contractor and its sub-contractors could alter the schedules of the associated manufacturing workflow (macro-workflow), resulting in the manufacturing workflow itself having to be adapted in order to meet stringent delivery deadlines.

- Macro- and micro-workflows can be at different control levels; however, this should not cause any issues, due to the highly modular way in which the micro-workflows have been modeled as black boxes within macro-workflows. Hence, both the macro-workflow and micro-workflow are created, and their adaptivity managed, just like the overall distributed workflow process and its constituent workflow processes as described earlier. This is because both types of workflows will have components that are executed by different agents representing individual workflow servers. The only major difference is that the micro-workflow is created **after** the macro-workflow is created.
- Agent Communication Languages (ACLs) such as KQML [Finin et al. 1997] can be used for implementing the multi-agent conversations. Each edge in the multi-graph represents a performative sent by an agent to its recipients, which would either be a broadcast message to the recipients, request for information, reply to an earlier message, etc.

Figure 12 presents a graphical depiction of macro- and micro-workflow using a (highly simplified) example. As per our definitions, the multi-agent negotiation (depicted in italics) with the sub-contractors will determine which sub-contractor will develop which component. The outcome of this negotiation will have an impact on the subsequent macro-workflow processes “Obtain components from sub-contractors” and “Final Assembly.”

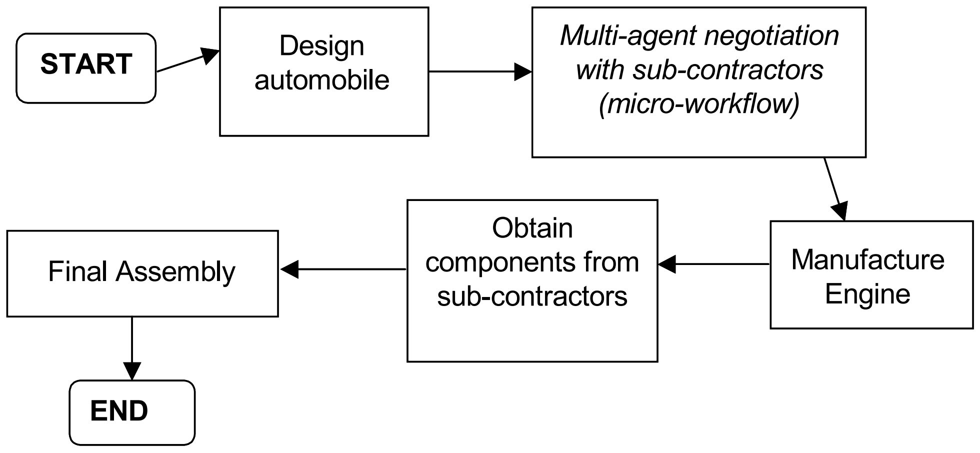
START
Design automobile
Multi-agent negotiation with sub-contractors (micro-workflow)
Manufacture Engine
Obtain components from sub-contractors
Final Assembly
END

Figure 12. Micro- and Macro-Workflow Example

III. FLEXIBLE AGENT SOCIETIES

In this section, we describe our understanding of a Flexible Agent Society and how it would function using the basic concepts described in section II.

INTRODUCTION

A Flexible Agent Society (FAS) is envisioned as a collection of agents, collaborating with each other in order to execute common business processes that meet certain business goals. It could also be described as a virtual organization or an e-services ecosystem. The FAS is an extension of the Contractual Agent Society idea [Dellarocas 2000; Dellarocas and Klein 1999]. The main difference is that, in an FAS, agents are free to form collaborations/associations with each other and define and execute common workflows.

Hence the FAS provides the following functions (the appropriate services provided by the FAS Administrator, who plays a role similar to that of the Central Coordinator described in section II, is given in italics):

- There is a facility for admitting members into the society in an orderly fashion, i.e., there are rules and procedures for this (*registration, reputation* [Dellarocas 2000; Dellarocas and Klein 1999]) so that only the "right" members are admitted and in the "proper" manner.

- Members of a society can "discover" each other in case they want to transact business with each other (*matchmaker* [Dellarocas 2000]).

- Once discovered, the FAS Administrator facilitates negotiations among the members in order to reach an agreement about how to interact, and at what cost to each society member (*negotiation, loose workflows, and multi-agent conversations*). Sometimes, the society itself may recommend/mandate certain procedures to be used (*medium and tight agent-oriented adaptive workflow*).

  For example, the FAS Administrator can mandate a certain shipping workflow to be used for transportation of products between members who decide to form a collaboration under the auspices of the FAS. The FAS Administrator can also mandate only a certain type of auction (say, English Auction) as the micro-workflow to be used by all FAS members.

- Collaborations between the society members can also be defined in terms of a common goal that needs to be met, and also in terms of "contracts" [Dellarocas 2000], which are essentially commitments that each agent makes toward the Society (*planning layer of Figure 3, goal and risk-based planning* [Narendra 1999]).

- The FAS Administrator also monitors interactions between the members, measures their effectiveness based on certain predetermined parameters, and uses this information as historical data for simulation and planning and for scheduling of future collaborations (*Admin Module, Simulation and Modeling* [Minar et al. 2001]; planning layer of Figure 4).

The conceptual architecture of our FAS is presented in Figure 13 (the red colored arrows depict the usage scenario described below):

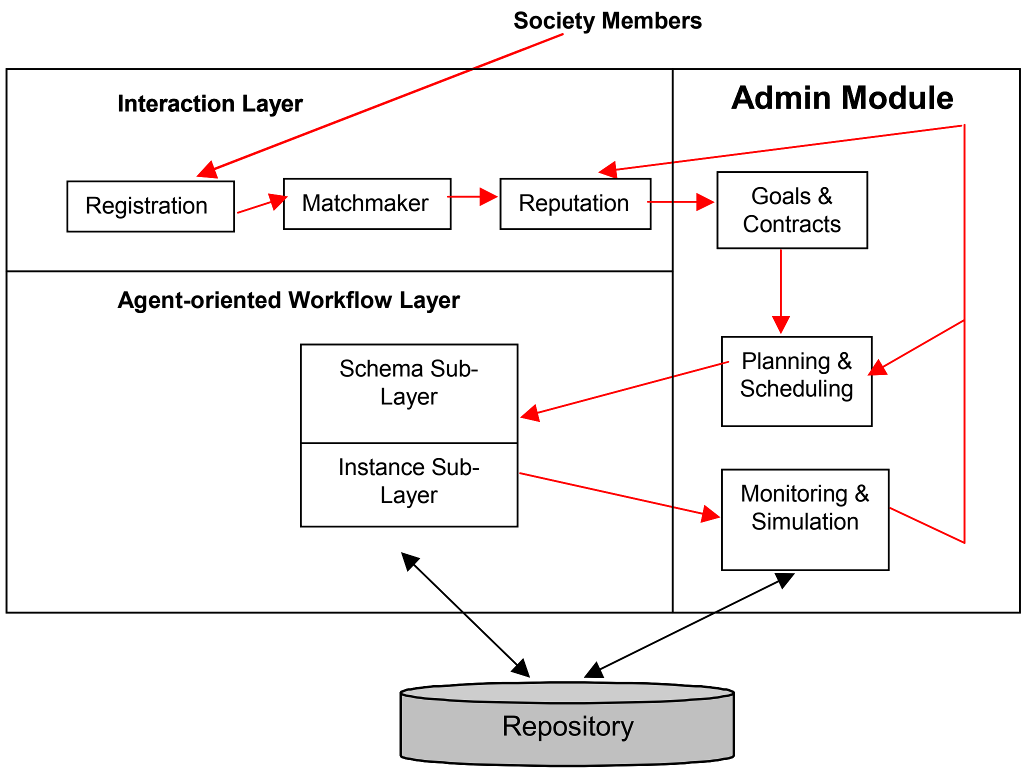

Figure 13. Flexible Agent Society: Conceptual Architecture

The registration (not needed for members already registered), matchmaker, and reputation services as described in Dellarocas are useful for our FAS, and we will incorporate them into the Interaction Layer. The matchmaker service will, upon request from a client agent, find the appropriate agents meeting the client agent's requirements. The reputation service basically monitors the extent to which the member agents have fulfilled their goals and contracts, and assigns "trustworthiness values" (based on Quality of Service parameters, security constraints, etc.) to the member agents. This service can also implement Role-Based Access Control (RBAC) mechanisms, such as ensuring that participating agents meet certain necessary security constraints for the needed collaboration. In other words, the Interaction layer will be used by the society members (i.e., participating agents) to get admission into the society, discover each other, and form collaborations. As part of the collaboration process and based on the goals, the agents create the following:

- Contracts, which are commitments made by the agent toward the society and the collaboration.
- The workflows—both macro and micro—that they will follow as part of the collaboration. This can be at either loose (peer-to-peer defined), medium (peer-to-peer but with approval from the FAS Administrator) or tight (defined by the FAS Administrator). The inputs for this will come from the Reputation service, via the Planning & Scheduling and the Goals & Contracts modules.

Once this is done, the agents start executing their workflows and conversations, with the help of the Agent-oriented Workflow Layer. The Schema and Instance modules are functionally the same as the Schema and Instance layers of our three-tier architecture described in section II (the functionality of the Planning layer will be subsumed by the Admin module in Figure 13), hence adaptivity can also be managed in the same way. The results of the workflow executions are sent to the Monitoring & Simulation module, which uses it for monitoring the workflow execution and alerting the agents and the FAS Administrator in case of problems. This data is stored in the Repository, and the Monitoring & Simulation module also uses this data for building simulation models of the workflows and conversations. These simulation models can be used by the Planning & Scheduling module for planning and scheduling similar workflows and agent conversations in the future.

The flexibility in our FAS architecture arises from the fact that sufficient social control (as described in section 1 of Dellarocas) is addressed by imposing basic requirements via the Reputation service, such as Quality of Service (QoS), security constraints, etc. However, the participating agents are free to choose with whom they will collaborate, and also the manner of their collaboration. Even for those workflows at tight level, the choice of handing over control to the FAS Administrator can be made freely by the agents themselves. Hence our approach provides a sound architectural foundation for building an FAS and realizing the CAS vision of Dellarocas.

In the remainder of this paper, we focus on two of the essential aspects of an FAS: RBAC and Transactionality of Distributed Adaptive Workflows. In section IV, we describe our distributed RBAC mechanism, and in section V we describe how transactional aspects of distributed adaptive workflows are managed in our FAS.

DISTRIBUTED WORKFLOW MANAGEMENT IN FAS

Distributed workflow management in our FAS is performed at the Agent-oriented Workflow layer, thus:

- The mechanism for managing distributed workflows among the different hierarchical levels is essentially the same as described in section II. Typically, the overall workflow would be defined by a global FAS administrator (or any workflow starter, if the workflow is at loose or medium control level—this could be one of the participating organizations, which is one of the FAS members), who would then "delegate" sub-workflows to other participating organizations for execution. (Henceforth, we refer to the individual defining the workflow—either the workflow starter or workflow administrator—as "workflow definer".) This way, we could have a delegation hierarchy of organizations, where a delegating organization O_d is defined to be superior (i.e., at a higher level in the hierarchy) of the organizations to which O_d does the delegation.

- As described in section II, we restrict the role of the overall workflow definer to defining only the overall workflow in detail; individual sub-workflows are merely specified as black boxes with prespecified inputs and outputs, and it is left to the respective organization to define the sub-workflow satisfying the inputs and outputs.

- However, care should be taken to ensure that an organization at a lower hierarchical level does not delegate a sub-workflow to an organization that is either at the same level or is its superior in the hierarchy. This will prevent "circularity" in distributed workflow definition.

- We accommodate chained and parallel synchronized workflows in the following manner:
  - ➢ A workflow can be "chained" or "parallel synchronized" only with a workflow that is **not** at a higher level in its hierarchy
  - ➢ The workflow definers of the respective organizations that are executing the chained or parallel synchronized workflows, will need to define and manage them together

IV. DISTRIBUTED SECURITY AND RBAC IN FAS

RBAC FOR FLEXIBLE WORKFLOW

The RBAC and Role Graph models introduced in section II need to be enhanced for Flexible Workflow (i.e., where the person who creates and starts a workflow is not the workflow administrator). Hence, for a loose level workflow, a different Role Graph may need to be maintained. The MaxRole for this Role Graph will be the workflow definer. For a medium level workflow, the MaxRole will be shared by the workflow administrator and workflow definer, with the workflow administrator being able to override the decisions of the workflow definer. For a tight level workflow, the model will be the same as in section II.

Hence for loose level workflows, our RBAC model will operate more like a discretionary access control (DAC) model, where the workflow definer is given discretionary administrative privileges for his/her workflow [Osborn et al. 2000].

RBAC FOR FAS

Here, we will essentially have several agents (belonging to different organizations) collaborating with each other to execute a distributed workflow. Hence we will need to extend the RBAC and Role Graph models thus:

- At the global level (i.e., the overall workflow), there will be an Overall Role Graph, with each organization possessing a role (with its associated direct and inherited privileges). The workflow definer will, by definition, possess the MaxRole role.

- Each role in the overall Role Graph will map to a role in an Organizational Role Graph, where the equivalent organizational role (for the sub-workflow assigned to the organization) will be stored in the Organizational Role Graph. The Organizational Role Graph will be maintained by the respective organization.

- In the case of distributed workflows at several hierarchical levels, i.e., if the organization itself subcontracts its assigned workflow tasks to other organizations, the Organizational Role Graphs are recursively created and maintained down to the “leaf” level in the hierarchy.

- If the overall workflow is at the loose level, then the workflow definer can assign specific organizations to the tasks or sub-workflows that make up the overall workflow (i.e., role-task assignment at the global organizational level). If the overall workflow is at medium control level, then these assignments need to be approved by the FAS Administrator before they become valid.

If the overall workflow is at the tight level, then the assignments can only be done by the FAS Administrator. Of course, within an organization, the authority for user/role and role/task assignment will rest on the workflow definer of the sub-workflow (the workflow definer in cases of loose or medium workflow, and the organization's workflow administrator in cases of tight workflow).

- For chained workflows, the respective participating organizations would need to communicate to each other the tasks at which the chaining will take place, along with names of the appropriate users executing the tasks, so that they can be managed together.

- For parallel synchronized workflows, the synchronization points have to be defined by the respective workflow definers of each organization while defining their respective sub-workflows, so that they can be managed together.

- Separation of duty constraints, whether SSOD or DSOD, can be specified as global policies and constraints at the overall workflow level; at any hierarchical level, each organization would also have its own SSOD & DSOD constraints

We are, therefore, logically extending the ideas from section II for implementing distributed RBAC. One point to note is that our approach, as recommended in Kang et al. [2001], achieves the important separation of the workflow-specific and organization-specific security infrastructures.

In the next section, we will describe how distributed RBAC is managed in the presence of adaptivity.

V. DISTRIBUTED WORKFLOW ADAPTIVITY AND TRANSACTIONALITY IN FAS

In the previous section, we defined RBAC implementation for flexible workflows in our FAS. However, since flexible workflows are expected to be highly adaptive, we need to consider how best to incorporate adaptivity—and by extension, transactionality—into our model.

At the most elementary level, any workflow change—either at schema or instance level—involves a combination of the following atomic operations:

- Deletion of a task
- Addition of a task

For either schema or instance level adaptivity, there are essentially two ways in which a workflow instance can be modified:

- On-the-fly, i.e., while the workflow is running

- At entry time, i.e., when the new workflow instance begins executing

If the workflow change is at entry time, then transactionality considerations will not arise, since the new workflow instance is simply executed. Hence we only consider the case of on-the-fly workflow change.

PROTOCOL FOR WORKFLOW ADAPTATION

Since a typical workflow in our FAS involves multiple agents/organizations interacting among each other, adapting the workflow—either on-the-fly or at entry time—requires a well-defined protocol. This will itself become a kind of "meta-workflow" that will be executed and monitored by the workflow definer. Therefore our generic protocol is as follows:

- One of the participating agents feels the need for the workflow to be modified and informs the workflow definer. Alternatively, this need could emerge directly from the workflow definer.

- The workflow definer initiates a multi-party negotiation among the participating agents. The outcome of the negotiation is the new workflow, along with the modifications—task deletions and task additions—needed for adapting the workflow.
  - ➢ If the overall workflow is at the loose level, then the workflow definer will him/herself modify the workflow via appropriate task aborts and roll-backs (discussed in detail below).
  - ➢ If the overall workflow is at the medium level, then the FAS Administrator will have to approve the modification before it can be implemented.
  - ➢ For a sub-workflow at the loose level, the same task abort-and-rollback procedure will be followed by the respective sub-workflow definer.
  - ➢ Similarly, if any sub-workflow is at the medium level, then the respective sub-workflow administrator will have to approve the modification. It is assumed that this will be part of the multi-party negotiation process.
  - ➢ This process is repeated for the sub-workflows of the sub-workflows and so on, recursively.

- The actual workflow adaptation is now ready to be implemented.

TASK ABORTS AND ROLL-BACKS

When a workflow has to be adapted mid-stream, several existing tasks may need to be aborted or rolled back so that “replacement” tasks can be introduced. Given the distributed nature of workflows in our FAS, this will need to be implemented in a cross-organizational manner. Hence we extend the work of Vonk et al. [2000a, 2000b] and implement roll-backs in the following way:

- Only tasks that are in the ACTIVATED, SUSPENDED, DONE, or FAILED state (see section II) can be rolled back. It is assumed that the rolling back activity is carried out by a compensating task, which is also assumed to be defined along with every task. This compensating task can be either fully or partially executed. (For example, a compensating task for a payment fulfillment task, could be to reverse the payment by returning the money.)
  - ➢ Hence we introduce two additional sub-states of the ACTIVATED state to denote a task that is being rolled back:
    - ✓ PARTIAL-ROLL, to denote a task that is partially rolling back (i.e., compensating)
    - ✓ FULL-ROLL, to denote a task that is fully rolling back

- A task that has finished rolling back fully is deemed to be in the NOT-ACTIVATED state; i.e., it is as if the task has not been executed at all.

- Some tasks cannot be rolled back. For example, items that have been downloaded via the Internet (such as music, documents in electronic format, software deliverables, etc.) cannot be returned. In the physical world, industrial components that have been imperfectly manufactured cannot be rolled back. In such situations, since compensating tasks cannot be defined, the respective workflows will have to be adapted taking into account the change of state of the repositories of the FAS and the participating agents in the workflow. This is, of course, a business decision, and out of the scope of this paper, since it requires detailed research into workflow semantics [Singh 1996].

  Likewise, we implement aborts in the following way:

- Only tasks that are in the NOT-ACTIVATED, ACTIVATED, or SUSPENDED state can be aborted, provided this abortion does not permanently alter the states of the repositories of the FAS and the participating agents in the workflow. For example, a payment fulfillment task cannot be aborted if it has been successfully completed, since the repositories would already have recorded the payment information; hence such a task needs to be rolled back by a compensating task, which will reverse the payment and restore the original state of the Repository. As in the case of task roll-backs, this is also a business decision, and out of the scope of this paper, requiring detailed research into workflow semantics [Singh 1996]. Hence if a task abort does change the states of the aforementioned repositories, then the respective workflows will need to be adapted taking into account the changed repository states. How this will be done, however, is a business decision depending on the individual workflows, and is out of the scope of this paper.

- An important point to note is that a task that is in either PARTIAL-ROLL or FULL-ROLL state **cannot** be aborted.

- Let a task T need to be aborted (rolled back) in a workflow W. If, as a result of T aborting (resp. rolling back), a sub-workflow of W is affected, then all tasks in the sub-workflow that are in NOT-ACTIVATED, ACTIVATED, or SUSPENDED (resp. ACTIVATED, SUSPENDED, DONE, or FAILED) states will also need to be aborted (resp. rolled back).

  For example, in the Adaptive & Flexible Workflow Example in section II, let us assume that the Police Verification and Lawyer Verification tasks are performed as a separate sub-workflow and that they are represented as a single task (let us call it "Overall Verification") in the main workflow. If this task is aborted, then, depending on the state of the Police Verification and Lawyer Verification tasks, these tasks will have to be either aborted or rolled back.

  ➢ If a task T' in the sub-workflow of W is in the DONE or FAILED state, and W is affected by T aborting, then T' will need to be rolled back.

    In our example, if the Police Verification task is in the DONE or FAILED state, and the parent workflow is affected by the Overall Verification task aborting, then the Police Verification task will need to be rolled back.

  ➢ If T' is in the NOT-ACTIVATED, ACTIVATED, or SUSPENDED state, and W is affected by T rolling back, then T' will need to be aborted.

    That is, if the Police Verification task is in the NOT-ACTIVATED, ACTIVATED, or SUSPENDED state, then it will need to be aborted.

  ➢ If W is itself a sub-workflow of a workflow Y, then the tasks in Y that map to the sub-workflow W will need to be aborted (resp. rolled back), subject to the same caveats described above for the task T'.

For chained workflows, we will have a task in one workflow A accessing another task in a workflow B. For the purposes of aborts and roll-backs, we will consider B as the sub-workflow of A, and the same arguments described above will apply here.

For a pair of parallel synchronized workflows A and B, we will have the case where A and B will "meet" at a synchronization point (typically, a route node). In this case, either workflow can play the part of the sub-workflow. Typically, the workflow where the change is occurring first will be considered the "parent" workflow; this change will trigger task aborts and roll-backs in the other sub-workflow.

ADMINISTRATION OF WORKFLOWS IN FAS

For the FAS Administrator, workflow administration (both macro and micro) is done in a manner similar to that described in section II. The additional task in the administration lifecycle, is that of maintenance (essentially, Role Management) of the Overall Role Graph of the FAS. This task is carried out during Process Creation and Process Upgrade.

For each individual administrator, the administration lifecycle is the same as that of the FAS Administrator. The only difference is that the individual administrator maintains his/her individual Organizational Role Graph. This maintenance will need to be done in sync with the Role Graphs of organizations above in the hierarchy, and also with the Overall Role Graph.

Since our distributed workflow infrastructure has been designed to preserve the autonomy of each organization, the individual administrator can upgrade his/her respective sub-workflows independent of the other sub-workflows, as long as the upgrade information is propagated to the FAS Administrator.

SECURITY CONSIDERATIONS

In this section, we extend the ideas presented in section II for adaptivity of workflows in our FAS, i.e., workflows that are distributed and flexible, in the following way:

- At any level, the respective workflow definer is authorized to create/modify user/role and role/task assignments, since he will always occupy the MaxRole role. However, if the (sub-)workflow is at the medium level, this will require the approval of the respective workflow administrator.

- Since the workflow is distributed, we could have organizations as users executing certain tasks and sub-workflows, instead of individual users (since the individual users will be directly assigned by the organizations themselves).

- If we are to extend the algorithm of section II for distributed workflow, then the following steps should be executed in order:
    - ➢ Modifications in role/task assignments should be made starting at the global level (i.e., at the Overall Role Graph), as per the algorithm of section II, and working downward to the individual Organizational Role Graphs. Note that since some users could be organizations instead of individuals, this would involve reallocating organizations to tasks and sub-workflows.

- ➢ This reallocation may necessitate reallocation of users within the individual organizations executing their respective sub-workflows. Hence, based on this reallocation, the algorithm of section II is again applied to all of the Organizational Role Graphs down the hierarchy, until all of the Role Graphs up to and including all the leaf levels are covered.
- ➢ If there is a SSOD or DSOD constraint violation at any level in this hierarchy, all reallocations up to and including this level are aborted, and an error message is signaled to the following:
  - ❖ The workflow definer and workflow administrator at that level, and
  - ❖ All workflow definers and workflow administrators at parent levels, all the way up to the global workflow definer and the FAS Administrator.

VI. CONCLUSIONS AND FUTURE WORK

We have described our idea of a Flexible Agent Society (FAS), an extension of the Contractual Agent Society (CAS) work [Dellarocas 2000; Dellarocas and Klein 1999] and our earlier work on agent-oriented adaptive workflow [Narendra 1999, 2000, 2001a, 2001b, 2001c, 2001d]. We have also shown how adaptive workflows and multi-agent interactions can be seamlessly integrated and modeled in the FAS. The major contributions are incorporation of security (specifically, role-based access control) and transactional aspects of adaptive and distributed workflows into the FAS.

There are several opportunities for future work:

- *Implementation and experimentation*: The FAS needs to be implemented and tested on various usage scenarios. An evaluation should be conducted of the need for enhancements to the workflow model depicted in Figures 1 and 2 to include additional states for handling errors/exceptions and other types of conditional branches as described in van der Aalst et al. [2000] (e.g., XOR branches, M-out-of-N branches, etc.).

  The other important research issue is task aborts and roll-backs. As explained in section V, certain tasks cannot be rolled back (resp. aborted) and their execution (resp. abortion) could permanently alter the state of the repositories of the FAS and the individual agents. For such a problem, research needs to be done on how to "recover" the repository states so that the workflow can be adapted successfully. This is a problem that requires more investigation into workflow semantics [Singh 1996].

- *Policies and Trust Management*: The functionality of the Reputation service should be extended to more fully implement trust management [Herzberg et al. 2000]. A related and very important research area is that of Policies. Our architecture needs to be enhanced to include support for specifying, executing and checking the RBAC policies for flexible workflows, borrowing ideas from [Damianou et al. 2001].

- *Distributed Service Management*: The functionality of the Monitoring & Simulation module needs to be extended, especially developing techniques for distributed service management [Sahai et al. 2000].

- *Distributed Workflow Administration*: We have briefly described the administration lifecycle for the individual workflow administrators and the FAS Administrator. This functionality needs to be specified in more detail. This will also need to be integrated with Distributed Service Management.

- *Multi-Agent Conversations*: We touched upon multi-agent conversations in an FAS, preferring instead to investigate the architectural underpinnings of workflow definition and execution, while classifying multi-agent conversations as micro-workflows. However, multi-agent conversations being much more dynamic than workflows, they need to be modeled in a different way. We are currently working on developing an appropriate modeling mechanism for multi-agent conversations—with special emphasis on adaptivity and transactionality (similar to that described in section V for macro-workflows)—that is consistent with our FAS approach [Narendra 2002]. We also intend to use this approach in defining appropriate protocols for automating Distributed Workflow Administration, with emphasis on process upgrade.

- *Contract Specification Languages*: As described in Dellarocas [2000], there is a need to develop appropriate contract specification languages that can be used in matchmaking and to link them to the goals to be met by the collaboration.

VII. ACKNOWLEDGMENTS

The author wishes to thank his manager, Srivatsa Krishnaswamy, and the ESC Center Manager, Padma Ravichander, for supporting his work. The author also thanks Bernard Burg of HP Labs, for his useful review comments on an earlier paper [Narendra 2001b], on which this paper is based. Special thanks are also due to the Editor of JAIS and the anonymous reviewers, whose feedback significantly improved the quality of the paper.

Editor's Note: This paper was first received on August 30, 2001, and was with the author one month for two revisions. Phillip Ein-Dor was the editor.

VIII. REFERENCES[1]

Cholewka, D. G., R. A., Botha, and J. H. P. Eloff. "A Context-Sensitive Access Control Model and Prototype Implementation," in *Proceedings of the 15th International Information Security Conference (IFIP/SEC 2000)*, Beijing, China, August 2000 (available from **http://www.petech.ac.za/secwflow/images/papers/SEC2000Cholewka.pdf**).

Cook, J. M. "Discovering Models of Software Processes from Event-Based Data," *ACM Transactions on Software Engineering and Methodology* (7:3), July 1998, pp. 215-249.

[1]Editor's Note: The following reference list contains the addresses of World Wide Web pages. Readers who have the ability to access the Web directly from their computer or are reading the paper on the Web can gain direct access to these references. Readers are warned, however, that

1. These links existed as of the date of publication but are not guaranteed to be working thereafter.
2. The contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions references.
3. The authors of the Web pages, not JAIS, are responsible for the accuracy of their content.
4. The author of this article, not JAIS, is responsible for the accuracy of the URL and version information.

Damianou, N., N. Dulay, E. Lupu, and M. Sloman. "The Ponder Specification Language," Workshop on Policies for Distributed Systems and Networks (Policy2001), Hewlett-Packard Labs Bristol, January 29-31, 2001 (available from **http://www.doc.ic.ac.uk/~mss/Papers/Ponder-Policy01V5.pdf**).

Dellarocas, C. "Contractual Agent Societies: Negotiated Shared Context and Social Control in Open Multi-Agent Systems," *Workshop on Norms and Institutions in Multi-Agent Systems, Fourth International Conference on Multi-Agent Systems (Agents-2000)*, Barcelona, Spain, June 2000 (available from **http://ccs.mit.edu/dell/aa2000/paper13.pdf**).

Dellarocas, C., and M. Klein. "Civil Agent Societies: Tools for Inventing Open Agent-Mediated Electronic Marketplaces," *Proceedings of the Workshop in Agent-Mediated Electronic Commerce* (co-located with IJCAI'99), Stockholm, Sweden, July 199 (available from **http://ccs.mit.edu/dell/civilagentsocieties.pdf**).

Finin, T., Y. Labrou, and J. Mayfield. "KQML as an Agent Communication Language," in *Software Agents*, Jeffrey Bradshaw (ed.), Cambridge, MA: AAAI/MIT Press, 1997 (available from **http://www.csee.umbc.edu/~jklabrou/publications/mitpress96.pdf**).

Griss, M. L. ""My Agent Will Call Your Agent … But Will It Respond?," *Software Development Magazine,* 2000.

Herzberg, A., Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid. "Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers," December 2001, **http://www.haifa.il.ibm.com/projects/software/e-Business/papers/Paper_Trust.pdf**.

Hollingsworth, D. *Workflow Management Coalition: The Reference Model*, Workflow Management Coalition, January 1995, available from **http://www.wfmc.org/standards/docs/tc003v11.pdf**.

Kang, M. N., J. S. Park, and J. N. Froscher. "Access Control Mechanisms for Inter-organizational Workflow," *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies* (SACMAT), New York: ACM Press, May 2001, pp. 66-74.

Kappel, G., S. Rausch-Schott, and W. Retschitzegger. *Coordination in Workflow Management Systems: A Rule-Based Approach,* Heidelberg, Germany: Springer, 1998.

Kradolfer, M., and A. Geppert. "Dynamic Workflow Schema Evolution based on Workflow Type Versioning and Workflow Migration," in *Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems*, Edinburgh, Scotland, September 1999, pp. 104-114 (available through **http://www.ifi.unizh.ch/dbtg/Projects/TRAMs/trams.html**).

MASIF (Mobile Agent System Interoperability Facilities) specification, 1998 (available from **ftp://ftp.omg.org/pub/docs/orbos/98-03-09.pdf**).

Minar, N., R. Burkhardt, C. Langton, and M. Askenazi. "The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations," 2001, available from **http://www.swarm.org/index.html**.

Narendra, N. C. "*AdaptAgent*: Integrating Adaptive Workflows and Multi-Agent Conversations for B2B E-Commerce," manuscript in progress, 2002.

Narendra, N. C. "*AdaptAgent*: Integrated Architecture for Adaptive Workflow and Agents," *Proceedings of International Conference on Artificial Intelligence, Special Session on Agent-oriented Software Architectures for B2B,* H. Arabnia (ed.), Las Vegas, NV, June 2001a.

Narendra, N. C. "Adaptive Workflow Management: An Integrated Approach and System Architecture," *ACM Symposium on Applied Computing*, New York: ACM Press, 2000, pp. 858-866.

Narendra, N. C. "Flexible Agent Societies: Flexible Workflow Support for Agent Societies," *Proceedings of International Conference on Computational Intelligence for Modeling, Control, and Automation - CIMA 2001,* Las Vegas, July 2001b.

Narendra, N. C. "Flexible Support and Management of Adaptive Workflow Processes," under review, 2001c.

Narendra, N. C. “Goal-based and Risk-based Creation of Adaptive Workflow Processes,” *American Association for Artificial Intelligence (AAAI) Spring Symposium*, Stanford University, Palo Alto, CA, 1999; also available from **http://aifbhermes.aifb.uni-karlsruhe.de/AAAI2000/CameraReady/NNarendra00.pdf**.

Narendra, N. C. “An Integrated Security Infrastructure for Adaptive Workflow,” under review, 2001d.

Nyanchama, M., and S. Osborn. "The Role Graph Model and Conflict of Interest," *ACM Transactions on Information and Systems Security* (2:1), February 1999, pp. 3-33 (available from **http://www.csd.uwo.ca/faculty/sylvia/conflict.ps**).

Osborn, S., R. S. Sandhu, and Q. Munawer. "Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies," *ACM Transactions on Information and Systems Security* (3:2), 2000 (available from **http://www.csd.uwo.ca/faculty/sylvia/models.ps**).

Reichert, M., and P. Dadam. “ADEPTflex: Supporting Dynamic Changes of Workflows Without Loosing Control,” *Journal of Intelligent Information Systems*, Special Issue on Workflow Management Systems (10:2), 1999, pp. 93-129. (available from **http://www.informatik.uni-ulm.de/dbis/papers/1997/ReDa97c.ps**).

Sahai, A., J. Ouyang, V. Machiraju, and K. Wurster. “End-to-End E-service Transaction and Conversation Management through Distributed Correlation,” *Hewlett-Packard Labs Technical Report HPL-2000-145*, 2000 (available from **http://hpl.hp.com/techreports/2000/HPL-2000-145.pdf**).

Sandhu, R. S., and P. Samarati. "Access Control: Principles and Practice," *IEEE Communications* (32:9), September 1994 (available from **http://www.list.gmu.edu/journals/commun/pdf_ver/i94ac.pdf**).

Shan, M-C., J. Davis, W. Du, and Y. Huang. “HP Workflow Research: Past, Present, and Future,” *Hewlett-Packard Labs Technical Report HPL-97-105*, August 1997 (available from **http://www.hpl.hp.com/techreports/97/HPL-97-105.html**).

Singh, M. P. "Formal Semantics of Workflow Computations," Technical Report TR-96-08, Department of Computer Science, North Carolina State University, January 1996 (available from **http://www.csc.ncsu.edu/faculty/mpsingh/papers/databases/wf_semantics.pdf**).

Thomas, R., and R. S. Sandhu. " Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management," in *Proceedings of the IFIP WG 11.3 Workshop on Database Security*, Lake Tahoe, CA, August 11-13, 1997, London: Chapman & Hall (available from **http://www.list.gmu.edu/confrnc/ifip/pdf_ver/i97tbac.pdf**).

van der Aalst, W. M. P, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. "Workflow Patterns," BETA Working Paper Series, WP 47, Eindhoven University of Technology, Eindhoven, 2000 (available from **http://tmitwww.tm.tue.nl/staff/wvdaalst/Publications/p108.pdf**).

Vonk, J., W. Derks, P. Grefen, and M. Koetsier. “Model, Architecture and System for Cross-Organizational Transaction Support in Virtual Enterprises,” 2001a (available from **http://www.ub.utwente.nl/webdocs/ctit/1/00000031.pdf**).

Vonk, J., P. Grefen, E. Boertjes, and P. and Apers. “Distributed Global Transaction Support for Workflow Management Applications,” 2001b (available from **http://www.ub.utwente.nl/webdocs/ctit/1/0000000e.pdf**).

Whittingham, K. "OpenWater - White Paper", IBM Research Division, Zurich Research Laboratory, 1999.

Wooldridge, M., and N. R. Jennings. "Intelligent Agents: Theory and Practice," *Knowledge Engineering Review*(10:2), 1995 (available from **http://www.csc.liv.ac.uk/~mjw/pubs/ker95.ps.gz**).

IX. ABOUT THE AUTHOR

N. C. Narendra is a Software Architect at Hewlett-Packard India Software Operations Ltd., in Bangalore, India. He is the author of over 20 papers and technical articles. His research interests are in the areas of information systems, workflow, agent technology, security, and e-service management. He is a reviewer for *IEEE Internet Computing* and has been a reviewer for *ACM Symposium on Applied Computing*.

Journal of the Association for Information Systems

ISSN: 1536-9323