A Scheme for Integrating E-Services in Establishing Virtual Enterprises*

Alan Berfield, *Panos K. Chrysanthis*
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260, USA
{alandale,panos}@cs.pitt.edu

Ioannis Tsamardinos
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260, USA
tsamard@cs.pitt.edu

Martha E. Pollack
Department of EE and Computer Science
University of Michigan
Ann Arbor, MI 48109, USA
pollackm@eecs.umich.edu

Sujata Banerjee‡
Info. Sci. & Telecom. Dept.
University of Pittsburgh
Pittsburgh, PA 15260, USA
sujata@tele.pitt.edu

Abstract

An important aspect of Business to Business E-Commerce is the agile Virtual Enterprise (VE). VEs are established when existing enterprises dynamically form temporary alliances, joining their business in order to share their costs, skills and resources in supporting certain activities. Currently, existing enterprises use workflows to automate their operation and integrate their information systems and human resources. Thus, the establishment of a VE has been viewed as a problem of dynamically expanding and integrating workflows. In this paper, we present an approach to combining workflows from different enterprises, using techniques developed in the Artificial Intelligence literature on planning. Our method takes two workflow views, one representing a service request and the other a service provision (advertisement), with a mix of vital and nonvital steps and a rich set of constraints, and returns a list of possible legal combinations, if any exist. It then uses plan-merging techniques to find potential conflicts between the two workflows, and to suggest additional constraints that can resolve the conflicts. The returned solutions represent terms for the establishment of a new VE, and can be evaluated by each side to determine which is most desirable.

*This material is based upon work partially supported by NSF IIS-9812532 and AFOSR F30602-00-0547 awards.

‡On Leave of Absence at Hewlett-Packard Laboratories, MS 1U-17, Palo Alto, CA 94304, USA.

1. Introduction and Motivation

Electronic Commerce is expanding from the simple notion of E-Store to the notion of *Virtual Enterprises* (VEs) where existing enterprises dynamically form temporary alliances, joining their business in order to share their costs, skills and resources in supporting certain activities. An example of a VE in the context of the travel industry would be the collaboration of different travel agents, airliners, ground transportation services, hotels, restaurants and entertainment services in order to set up and manage a tourism trip.

Many enterprises use workflows to automate their operation and integrate their information systems and human resources [19]. A workflow consists of a set of *activities* (also called *tasks*) that need to be executed according to given temporal constraints over a combination of heterogeneous database systems and legacy systems. A major challenge has been the development of *workflow management* systems (e.g., [9, 5, 13, 1]). Several techniques have been developed for correct and reliable specification, execution, and monitoring of workflows and the involved external support. Many of these techniques are extensions of those in transaction processing in databases combined with general middleware services such as those found in CORBA/DCOM and more recently in Java-based services such as Jini from Sun and E*Speak from HP.

Very recently the idea of the use of workflows to support multi-organizational processes that form a virtual enterprise has attracted some attention [10, 6, 8]. The establishment of a VE can be seen as a problem of dynamically expanding and integrating workflows in decentralized, autonomous and interacting workflow management systems

Proceedings of the 12th Int'l Wrkshp on Research Issues in Data Engineering: Engineering e-Commerce/ e-Business Systems (RIDE'02)
1066-1395/02 $17.00 © 2002 IEEE

[2, 7, 12]. During the establishment of a VE, a distributed, multi-organizational workflow emerges from the dynamic merging and reconfiguration of workflows representing E-Services in the participating enterprises. In our previous work, we looked at using mobile agents as a platform for advertising, negotiating, and exchanging control information about E-Services for the establishment of VE's [6]. In this paper, we focus on a method for verifying that the VE is compatible with workflows in the participating enterprises.

The contribution of this paper is a new method for establishing VEs, involving both the generation of outsourcing requests and the validation of constraints. The scheme incorporates techniques developed in the Artificial Intelligence (AI) literature on planning, specifically algorithms for merging temporal plans. Within the AI literature, a plan is a collection of steps (i.e., tasks), with causal, temporal, and resource constraints. A plan is intended to represent a course of action that will achieve a specified goal when executed beginning in a specified initial state. Critical to the notion of plans is that of *causal structure*: the steps in each plan are specified in terms of their preconditions and effects, and the plan records information about which steps cause (or *establish*) the preconditions of other steps. When merging together two plans, it is necessary not only to check that there are no violations of the temporal and resource constraints of the plans being merged, but also to ensure that the necessary causal relations are maintained in the merged plan. We argue in this paper that similar consistency requirements also hold when a VE is formed.

In the next section, we review the basic structures used in workflows. In particular, we describe a class of workflows that include specifications of preconditions and effects. In Section 3, we describe a VE and its components. Section 4 describes our detailed scheme for establishing such a VE. Section 5 deals with the current state of our implementation. We conclude with a summary in Section 6.

2. Workflow Model

Workflows encode tasks and the relationships among them. Workflow specification formalisms generally provide a small set of basic control flow relationships among tasks. Typically there are four such relationships: OR-split, AND-split, OR-join and AND-join. The first two relationships are used to specify branching decisions in a workflow whereas the remaining two specify points where activities converge to initiate the next activity within a workflow. An OR-join specifies alternatives whereas AND-join specifies required activities.

While the relations just listed provide information about the relative ordering of the tasks in a given workflow, to handle the problem of forming Virtual Enterprises, it is also necessary for the workflow to model a significant amount of information about each task. Thus, we will assume an enriched model of workflows in which each task has the following information associated with it:

- *Pre- and postconditions*, which specify what must be true before a task can be executed, and what will be made true as a result of the task's performance.
- *Causal links*, which relate each task that establishes a condition (listed in its postconditions) to the task that requires it (listed in its preconditions).
- *In- and out-parameters*, which are used in the evaluation of preconditions and postconditions. They carry information and engender data flow during execution. For example, a credit card number could be an in-parameter to a "pay for dinner" task.
- *Temporal Constraints* that specify the earliest and latest start and end times of a task, as well as the minimal and maximal durations of the task. They can be absolute times or relative to the execution of other tasks.
- *Resource Constraints*, which specify the equipment, material, or agent resources required for the task.
- *Significance*, which indicates whether the task is *vital* to the workflow and therefore must be executed, or whether it is *nonvital*, and need only be executed if feasible [6].
- *Cost*, which represents the price of the task.

Other information may also be associated with each task, such as rules for exception handling should the task fail. However, we will not be concerned with these types of information in the current paper.

As shown in Figure 1, a workflow can be graphically depicted with nodes (thick boxes) denoting activities and arrows denoting precedence. The figure represents a business trip from [15]. Shaded nodes indicate vital activities that must be completed to ensure proper execution. Nodes with a pair of dashed lines leading to another workflow are hierarchical activities: those that can be decomposed into workflows themselves. AND-splits and AND-joins are represented implicitly when two or more causal links emanate or arrive at a node respectively. To represent OR-splits we insert a conditional node that creates two new execution contexts (branches), e.g., one for success and one for failure (in Figure 1, these are shown as nodes with edges labeled S and F). Tasks are executed only when their context is true. The OR-join is represented implicitly when the context S or F disappears from the labels of subsequent edges.

We are assuming a typical Workflow Management System (WfMS) architecture with our enriched model of workflows. Specifically, a WfMS consists of the following three basic components:

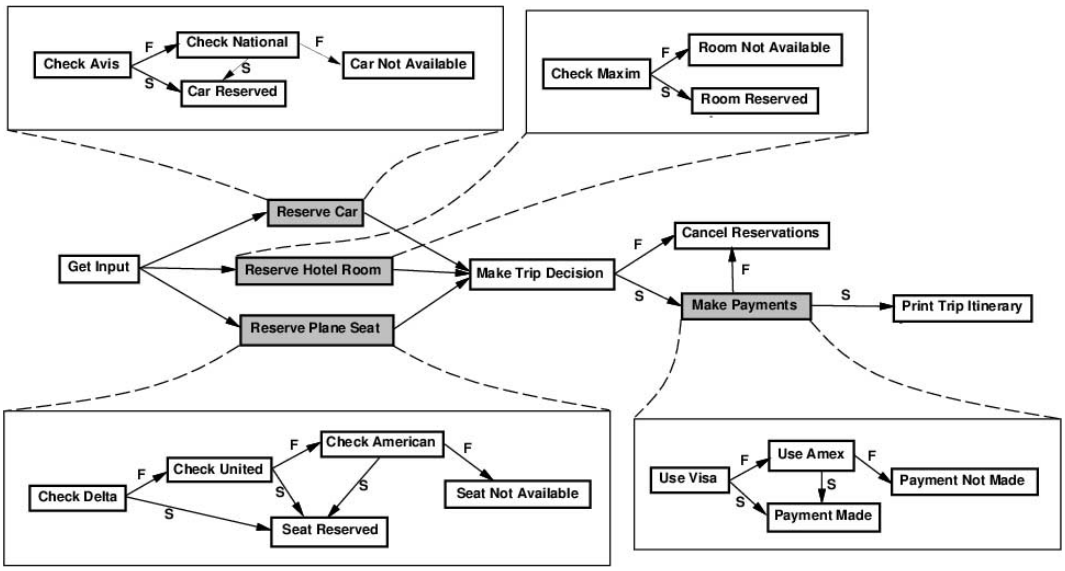

Figure 1. Trip Plan Workflow

- *Workflow Schema Library*, which contains workflow schemas or templates and generic constraints.
- *WfMS Services*, functions provided by the WfMS for managing workflows. These include specifying workflows, verifying their correctness, instantiating and scheduling them, executing them, and monitoring their execution.
- *Workflow Repository*, which contains all instantiated and scheduled workflows, i.e., the workflows the business is committed to performing.

3. Forming Virtual Enterprises

A Virtual Enterprise (VE) is formed when a business decides to commit to a new workflow, while outsourcing some of the work involved in that workflow. Consider the example of Jane Smith, an executive planning a trip to Vienna. She gets in touch with a travel agency to arrange the trip. She decides that while she is there she would like to attend an opera and tour the Art Museum. This adds the nonvital nodes "buy opera ticket" and "buy museum tour ticket" to the trip schema (Figure 1). The travel agency lacks connections with the entertainment/opera industry, so is unable to purchase such tickets. In order to satisfy the customer, they decide to outsource those tasks.

The above example represents a common reason for outsourcing. When a business receives a new request from a client, it takes the form of an instantiated workflow schema from its Workflow Schema Library. The client may have added constraints and/or customized the schema by adding new nodes, which could represent extra or special activities and opportunities. The business may select some of the tasks from the workflow to outsource and/or it may select some of the open conditions from the workflow and outsource their achievement. This outsourcing establishes a VE.

In our VE workflow specification, we use the notion of views to express outsourcing. Any subgraph of a workflow graph defines a *segment* or a *view* of the workflow. Formally, a workflow view can be defined as a *projection* on the graph based on some criteria ($projection(workflow, < criteria >)$). For example, consider the view that includes all and only the vital nodes of the full workflow. The requirement that nodes be vital is the criteria used by the projection.

$$VitalView = projection(workflow, \{a \mid a \in workflow \wedge a.significance = vital\})$$

The nodes in a view retain all information of their originals, including all constraints. However, because all constraints are maintained, a view may have nodes that have temporal constraints referring to other nodes not actually in the view, and may also have broken causal links possibly resulting in unsatisfied preconditions (i.e., a node in the view could have a precondition that was established by some node in the full workflow that is not in the view).

A workflow view can represent any activity performed by a *service provider* on behalf of a *service requester*. Consequently, workflow views can be used to express service requests or service provision (advertisement). In our proposed system, it is these workflow views that are being requested and advertised.

In our scheme, a request has the following structure:

Requests: $Rq = (P, G, RW)$
where P = Service Requester Profile,
G = set of Goals,
RW = Requested Workflow View

The profile can contain various information about the requester, such as name, site identification, credentials, etc.. It may also contain a target price range. The set of goals is a list of all goals (postconditions) that need to be accomplished. The workflow view captures all temporal constraints and resource usage issues involved.

In the example above, the request includes the profile of the travel agency, the goals "opera ticket purchased" and "museum ticket purchased", and a view with two nodes that indicate times by which the tickets must be purchased.

A requested workflow view can be potentially augmented during negotiation to match the service provider's workflow, reflecting opportunities, omitted activities and data. During a negotiation we may decompose the required view into several views and seek other service providers for the other parts of the view. In this way, a single initial request may lead to the establishment of a VE comprising multiple enterprises. A VE comprising multiple enterprises can also result when a service provider's view includes outsourcing. We will elaborate on this in the next section.

The structure of an advertisement is the same as that of a request.

Advertisements: $Ad = (P, G, AW)$
where P = Service Provider Profile,
G = set of Goals,
AW = Advertised Workflow View

It includes a profile, the set of goals accomplished, and a workflow view encompassing constraints. The profile, in addition to other information, may contain cost information for the workflow as a whole, such as minimum cost, maximum cost (cost with all nonvital steps), or both. The list of goals indicates what the advertised workflow actually does, and may also include goals associated with nonvital activities. Such advertisements will typically be stored in the databases of trading servers. Each provider may have a set of advertisements with the same goals but with a different associated workflow view (i.e., different constraints).

To return to our example, an advertisement that would be of interest to the travel agency would be for a business that specializes in Vienna cultural events, including opera. The single goal "opera ticket purchased" is accomplished. Its workflow view includes the tasks "contact opera houses", "read current reviews", and "purchase ticket."

The VE environment is a distributed environment. It consists of multiple businesses, acting as requesters and providers, using services provided by negotiation areas or trading places. The trading places could contain databases of advertisements and could provide services allowing businesses to both place advertisements and to find advertisements that meet their goals. Standardization of representation is clearly required (particularly of preconditions, effects, and goals), and could be enforced by the trading servers. A portion of this environment is shown in Figure 2. Two negotiation areas are depicted, as well as four businesses' WfMS. Shaded nodes again represent vital activities, and an advertised hierarchical Opera activity is shown partially expanded.

3.1. Commitment and Outsourcing Request Generation

In order to commit a new, possibly customized workflow, a WfMS needs to make sure that it is schedulable. A workflow is schedulable if it is *correct*, *complete*, and *compatible* with existing commitments.

Definition 1 Workflow Correctness: *A workflow is* correct *if and only if*

1. *it has no conflicting temporal or resource constraints,*
2. *for each goal/precondition P, there is a task that achieves P (the producer task), and it is ordered before the task that requires it (the consumer task), and*
3. *for each goal/precondition P, no task that may negate P can possibly be ordered in between the producer and the consumer.*

This notion of correctness is important as only correct workflows can possibly be executed. Note that some workflows may contain preconditions that are assumed to be established independently of the workflow itself. We will call such preconditions *open* with respect to the workflow. A simple example of such an open precondition is a workflow for renting a car that assumes the precondition of having a driver's license. Workflows with such open preconditions are incorrect until they have been combined with other workflows that establish all open preconditions.

Definition 2 Workflow Completeness: *A* complete *workflow is a workflow that specifies all tasks needed to achieve its goals and preconditions.*

Definition 3 Workflow Compatibility: *A workflow is* compatible *with another if none of its nodes conflict with any of the other's (and vice versa).*

This means that the temporal constraints, resource usage, and postconditions of its nodes do not prevent the execution of the nodes in the other workflow (though they may place limits on when those nodes can be executed). So for example, a compatibility conflict between workflows arises if

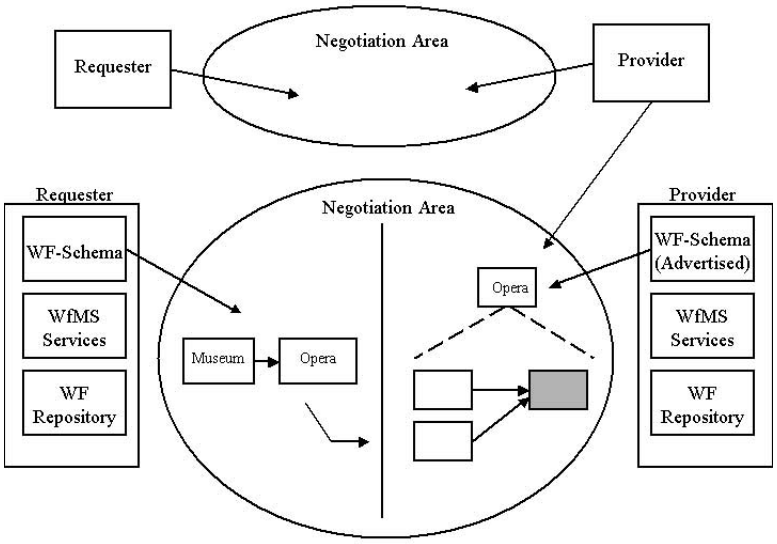

Figure 2. VE Environment

two tasks that use the same resource (e.g., equipment) are set to execute at the same time. Another example is a task that dictates that a robot move to the printing room for the purpose of getting a faxed itinerary, which conflicts with a task that moves the robot to another room that could be executed after going to the printing room but before fetching the fax.

An alternative definition of the compatibility of two workflows is that the workflow resulting from their union is correct. We propose the notion of a *merge* with the Workflow Repository for determining the compatibility of a workflow with the currently scheduled workflows (in the Repository). If the merge is successful, the new workflow can be committed and its execution enabled. If the merge is unsuccessful, the new workflow is not compatible and the business may consider outsourcing.

An effective merging process will check whether the above requirements (correct, complete, and compatible) are met, and will indicate where problems lie: what nodes are conflicting with others, which have unsatisfied (open) preconditions, or which the business lacks the necessary expertise (i.e., roles as resources) to accomplish. It may also suggest additional temporal or resource constraints that are required to ensure that they are met. However, it's desirable to impose a minimal set of extra constraints, i.e., to provide a least-commitment response, as this allows increased flexibility to respond to changes that may arise during execution.

The merge process can also be used to identify and construct outsourcing requests. In the event of an unsuccessful merge, any nodes from the new view that are indicated as problems by the merging process (those having irresolvable resource conflicts with existing commitments) will form part of the requested workflow view *VRq* by extracting them from the full workflow using projection. In addition to these nodes, for each open precondition in the new view not satisfied by the existing commitments (such preconditions will be found by the merge process), a new place-holder node is added to the view. Each of these new nodes represents a task that accomplishes one of the open preconditions, i.e., it has one of the open preconditions set as its postcondition, and any associated temporal and causal links are applied. The complete set of postconditions of every node in *VRq* make up the goal set of the request, G. In the simplest case *VRq* would be a single node, with associated constraints. More complex cases would involve multiple nodes and richer constraints.

Recall that projected nodes maintain all constraints and conditions they had in the parent workflow, and may therefore include unsatisfied preconditions and temporal constraints referring to nodes not in the view. This is not really a problem as they will be satisfied by non-outsourced nodes. The preconditions, along with in-parameters, represent the input to the outsourced view. Goals and out-parameters of the outsourced nodes represent the output.

4. Outsourcing Scheme

In this section, we discuss in detail the steps for outsourcing and establishing a VE.

Let R be a Requester and P be a set of providers $\{P_1, ..., P_n\}$. R has a set of workflows to which it is already committed, and which it stores in the WF Repository;

Proceedings of the 12th Int'l Wrkshp on Research Issues in Data Engineering: Engineering e-Commerce/ e-Business Systems (RIDE'02)
1066-1395/02 $17.00 © 2002 IEEE

let us call them *CR* (commitment workflows at requester). Similarly, each P_i has a set of workflows already committed to; let us call them $CP(i)$.

Let $Rq = (R, G, VRq)$ be a request of R for outsourcing with goals $G = \{G_i, ...G_n\}$ and workflow view *VRq*. Each P_i can provide a set of alternative workflow views $A(P_i)$ for achieving one or more G_j of Rq.

The problem of outsourcing is how to pick a set of workflow views S from the $A(P_i)$ of one or more P_i so that the combined set satisfies Rq and merges with *CR*, and each $A(P_i)$ in it merges with its provider's $CP(i)$. Specifically, such a set achieves all goals of the outsourced workflow, all temporal constraints are satisfiable, there are no resource conflicts, and for every precondition of every workflow activity in *CR* and $CP(i)$ there exists a causal dependency that ensures that the precondition will be met.

Formally, we want a set $S = wf_1 \cup wf_2 \cup wf_3 \cup ... \cup wf_n$, where $wf_j \in \bigcup_i A(P_i),\ j = 1, \ldots, n$ such that

- $postconditions(S) \supseteq G$
  (all outsourcing goals are met)
- $\forall wf_x \in S\ \ postconditions(wf_x) \cap G \neq \phi$
  (each workflow achieves at least one goal)
- $compatible(S, CR)$
  (S is compatible with the requester's commitments)
- $\forall wf_x \in A(Py)\ compatible(wf_x, CP(y))$
  (each alternative workflow is compatible with its provider's commitments)

The above suggests a solution that has three phases:

1. Finding a set of alternative workflows that satisfy Rq (*Terms for the Establishment of a VE*)
2. Checking for the satisfaction of $CP(i)$ (*Providers Validation of Terms and E-Service Bids*)
3. Check for the satisfaction of R (*E-Service Bid Evaluation*)

We elaborate on these phases in the next subsections.

4.1. Phase 1: Terms for the Establishment of a VE

As mentioned previously, we assume in this paper that finding alternative workflow views that satisfy a request Rq is a service provided by trading servers. Each alternative view represents a term for the establishment of a VE. During this first phase the sets $A(P_i)$ of alternative workflow views are generated. These views accomplish the goals G of Rq while not violating any of its constraints. For the sake of simplicity, we will assume in the rest of our discussion that there is only one trading server.

The service searches the database of the trading server, looking for advertisements that meet some or all of the requested goals. Which advertisements are examined first depends on the selection conditions being used. One such condition would include the desirability of first considering those that accomplish all goals, and only considering multiple, partial matches when all such are found. For each advertisement found and selected, the server must finally determine if it or any of its alternatives (involving different combinations of nonvital nodes) can meet the constraints of the request. This process continues until all advertisements that meet any goals have been examined, or some termination criteria are met (such as a deadline for search time).

For the detailed explanation, we will consider one such advertisement found and selected by the trading server that accomplishes all goals in G; let us call it $Ad1$.

As shown in Figure 3, the service must determine if $Ad1$ will satisfy the constraints in the request's workflow view Rq. To do this, $Ad1$ and Rq are first stripped down to only vital nodes using projection. Temporal constraints of the vital nodes may need to be adjusted, as any referring to nonvital nodes will be invalid. For any node that has such a constraint, there are four possible situations:

1. The nonvital node referred to has no constraints on its time[1] : the constraint on the vital node can be dropped.
2. The nonvital node has an absolute time: that time can be used.
3. The nonvital node has a time relative to some other vital node: the reference to that node can be used.
4. The nonvital node has a time relative to some other nonvital node: that nonvital must be searched in the same fashion for a time or vital node reference.

It may be beneficial to instead store such alternative constraints with the vital nodes in order to save computational time, though the number of nonvital nodes is likely to be small.

Next the service attempts to bind the constraints of the vital-only view of Rq (called RqV in the figure) to the stripped view of $Ad1$ (AdV in the figure). Binding adds the constraints of the requested nodes to the corresponding nodes in the advertisement (those that have the same postconditions). If AdV cannot support the added constraints (because they conflict with existing ones), the bind fails and the function must backtrack to find a different advertisement. Otherwise, the new bound advertised view BV is added to $A(P_i)$, where P_i is the provider of BV, and the search continues for its variants that include nonvital nodes.

The search for variants of BV considers combinations of BV and nonvital nodes from the full $Ad1$ and Rq. This can be achieved by the function *addNodes*, that adds a group of nodes to the workflow *BV*, restoring any modified temporal constraints that referred to them. This is basically a merge process. The addNodes function fails and returns null if the

[1] By "time" we mean either start or end time of the task, depending on which the specific temporal constraint refers to.

resulting view is incorrect (i.e., the new nodes cannot be added without violating constraints). If the function does not fail, the resulting view is added to $A(P_i)$.

It is interesting to point out that there is another possible method for this search: working in the other direction, starting by adding back all nonvitals and then removing them to find correct alternatives. It is not clear which approach is better, but we intend to investigate this in future work.

In either case, the search will proceed until all possible combinations have been attempted or some other termination criteria have been reached. As most views are expected to have 3 or fewer nonvital steps, finding all possible combinations is not likely to be impractical. Once the search has finished, $A(P_i)$ contains every alternative workflow solution for the workflow $Ad1$.

The service generates a set $A(P_i)$ for each P_i with at least one selected advertisement. The $A(P_i)$'s created in this fashion are now sent out to their respective provider for validation.

4.2. Phase 2: Providers Validation of Terms and E-Service Bids

The second phase of the outsourcing takes place at the providers of the advertisements. Each P_i receives the $A(P_i)$ generated for it in the previous phase, and must determine whether any of the workflow views in $A(P_i)$ are compatible with its $CP(i)$. Each alternative basically represents a potential new incoming workflow to be scheduled. Recall that such scheduling can be accomplished using the merge process. Thus, the provider attempts to merge each alternative with $CP(i)$ independently. Any that fail are removed from $A(P_i)$. Those that succeed can be kept to form the basis for the service bid. Of course, if $A(P_i)$ is empty at the end of this phase, then none of the views were compatible with the provider's commitments.

To generate the full service bid, each view remaining in $A(P_i)$ could possibly be expanded into multiple views if the provider wishes to add additional nonvital nodes (representing special offers or bonuses). Note that such additions would likely increase cost, but would possibly also increase value. The provider may also rank the solutions in order of preference or cost to help the later decision process.

Each provider sends its service bid to the requester to be evaluated in the next phase.

4.3. Phase 3: E-Service Bid Evaluation

In the third phase, the requester evaluates and selects an E-Service bid. Of the views in the service bids returned by the providers, the requester must determine which are compatible with its *CR*. This is done in exactly the same fashion as with the providers.

Each service bid in the returned list is combined with the rest of the original workflow to form a complete solution. For each solution, a merge is attempted with the committed workflows. Any that fail to merge are discarded. Those that successfully merge are correct views that each accomplish the outsourcing and that are compatible with both the provider's and requester's previous commitments.

The requester may then evaluate these remaining views to make a final decision as to which one will be used, which likely involves cost comparisons.

4.4. Multiple Partial-Solution Views

In the previous discussion, we assumed the simplest case where there exist advertised views that accomplish all the goals of the request. However, in many cases there may be no single advertisements that accomplish them all. This would require views from multiple advertisements to be combined in order to meet the requester's needs. In order to handle these cases, the described first and third phases need to be enhanced.

For example, in Phase 1, the search for alternatives must also search for combinations of advertisements that accomplish all goals. The merge process can be used again to verify that these combinations of advertisements are compatible with each other in addition to meeting the constraints of the request. For combined views belonging to a single provider, the combination (and its alternatives involving nonvital nodes) are grouped together as a single view.

The requester in Phase 3 must be aware that returned views do not necessarily accomplish all goals. Any E-Service bids that only satisfy some of the goals must be combined with other returned views to form complete solutions.

5. Implementation

In our previous work, we proposed to use mobile agents as a platform for establishing VE's [6]. Our goal is to implement our scheme described in the previous section on this platform. The idea is to use mobile agents to perform the phases of the scheme. The requester dispatches an agent with its request. The agent visits trading servers, and spawns copies of itself to deliver alternatives to different providers. It then gathers all returned service bids together and delivers the results back to the requester.

A core concept in our scheme for integrating E-Services is the merge process. It is this process that verifies whether or not different workflow views are compatible with each other. It is also responsible for adding nonvital nodes to views and verifying that a view is compatible with a business' existing workflow repository. The merge process can even be used to generate the outsourcing requests.

Repeat

- $Ad1 = Ad \in DB \mid < selection\ conditions >$
- $AdV = projection(Ad1, \{a \mid a \in Ad1 \wedge a.significance = vital\})$
- $RqV = projection(Rq, \{a \mid a \in Rq \wedge a.significance = vital\})$
- $BV = bind(AdV, RqV)$
- Repeat
  - $Nodes = projection(Ad1, selected\ nonvital \in Ad1) \cup projection(Rq, selected\ nonvital \in Rq)$
  - $Bx = addNodes(BV, Nodes)$
  - $If Bx \neq null \rightarrow A(P_i) = A(P_i) \cup Bx$
- Until all combinations of nonvitals found or termination criteria met

Until all Ads found that meet $< selection\ conditions >$ or termination criteria met

Figure 3. Service For VE Terms

Merging is not a trivial problem. It can be formulated as a *Constraint Satisfaction Problem* or CSP, with temporal features. The process must consider temporal constraints, resource usage, and causal links (preconditions and effects). There has been a great deal of research done on similar problems by the Artificial Intelligence community [14, 17, 20]. A number of formalizations have been developed for variations with more or less expressivity. The two that most closely match our problem are the Disjunctive Temporal Problem (DTP) and the Conditional Disjunctive Temporal Problem (CDTP).

For solving DTPs we have developed and implemented a new algorithm called Epilitis [16], along with algorithms that convert CDTPs to DTPs so that they may be solved by it as well. Epilitis builds on plan merging techniques used in a tool called PMA (Plan Management Agent) [18]. Epilitis integrates a number of techniques for pruning the search space, some of which are Conflict Directed Backjumping, Removal of Subsumed Variables, Semantic Branching, and no-good learning. Epilitis is currently the most efficient algorithm for solving such problems, as experimental results have shown that it is two orders of magnitude faster than the previous state-of-the-art solver, on synthetic benchmark problems.

In the prototype system we are currently developing, we will use Epilitis for the merging process at the WfMS and the trading servers. The representation that Epilitis expects is nearly identical to our enhanced model of workflows; the mapping between the two is trivial. Merging with Epilitis has all the properties discussed in Section 3. Any conflicting tasks are identified with explanation, and a minimal number of constraints are added. Plans with disjunctive temporal constraints are supported (for added flexibility), and duplicate nodes can be identified and pruned/combined.

Epilitis does not support the notion of significance (vital vs. nonvital tasks). However these are implemented in the higher-level layer that performs the phases of our scheme. Only this layer is aware of the vital/nonvital distinction. (This is the cause of some of the complexity in the search for alternatives, as all the different combinations of nonvitals must be attempted separately.) This layer also serves to interface Epilitis with a relational DBMS using Microsoft Access and MySQL that will be used to implement the Workflow Repositories.

The current version of Epilitis is written in LISP, but using JLinker we have interfaced it to the rest of our prototype which is being developed in Java. The new version of Epilitis currently being developed will be in Java as well.

6. Conclusions

We are concerned with integrating E-Services for the establishment of a VE, where such services are represented with workflows. We have therefore created algorithms that make use of existing plan merging and temporal reasoning algorithms from the AI literature. Our scheme is sound, in that the workflows it returns as possible merge candidates are guaranteed to be correct. It is also complete, in that it will find all such candidates, given sufficient time. It further ensures that the merge candidates are compatible with all businesses involved in the VE. It can create the outsourcing requests based on identified conflicts, handle any number of

nodes and workflows to be outsourced, and is flexible in that it can build a VE using multiple providers, each with their own set of constraints. Our scheme also takes into consideration that workflows have both vital and nonvital steps, and appropriately considers them in its search.

In our proposed system, the merging process is built with existing AI algorithms. The specific algorithm, Epilitis, is the best algorithm available at this time. It has been implemented and is a fully functional and working plan merging tool. Currently we are developing our prototype system. Our goal is to evaluate its performance in terms of speed and memory usage. Another area we intend to explore is its use as a plan/workflow repair system that would replace broken or invalidated nodes or views with alternatives, possibly located in different databases on various machines.

Recently, there have been a variety of platforms developed with business to business E-services and Virtual Enterprises in mind. E*speak [3] from HP, VorteXML [4], and CrossFlow [11] are examples. These systems provide various features for managing and monitoring VE's, along with some standards for communication. Such systems could potentially be augmented or used conjunctively with our scheme for automated VE establishment. We will investigate such possibilities as part of our future work.

References

[1] Alonso G., D. Agrawal, A. El Abbadi and C. Mohan. Functionalities and Limitations of Current Workflow Systems. *IEEE Expert*, 12(5), 1997.

[2] Casati F., S. Ceri, B. Pernici and G. Pozzi. Workflow Evolution, *Data & Knowledge Engineering*, 24(3): 211-238, 1998.

[3] Casati F. and M. Shan. Definition, Execution, Analysis, and Optimization of Composite E-Services. Bulletin of the Technical Committee on Data Engineering, 24(1):30–35, 2001.

[4] Christophides V., R. Hull, A. Kumar, and J. Simeon. Workflow Mediation using VorteXML. Bulletin of the Technical Committee on Data Engineering, 24(1):41–46, 2001.

[5] Chrysanthis P. K.. Guest Editor's Introduction to Special Issue on Workflow Systems. *Distributed Systems Engineering*, 3(4):211–212, 1996.

[6] Chrysanthis P., T. Znati, S. Banerjee, S. Chang. Establishing Virtual Enterprises by means of Mobile Agents. *Research Issues in Data Engineering*, 1999.

[7] Cichocki A. and M. Rusinkiewicz. Migrating Workflows. *Workflow Management Systems and Interoperability*, A.Dogac et al. (Eds)Springer Verlag, Series F, Vol 164, pp. 339–355, 1998.

[8] Davulcu H., M. Kifer, L. R. Pokorny, C. R. Ramakrishnan, I. V. Ramakrishnan, and S. Dawson. Modeling and Analysis of Interactions in Virtual Enterprises. *Research Issues in Data Engineering*, 1999.

[9] Georgakopoulos D., M. Hornick and A. Sheth. "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure." *Distributed and Parallel Databases*, 3(2), 1995.

[10] Georgakopoulos D., H. Sinha, K. Huff and B. Hurwitz. "Monitoring Multi-organizational Processes." *Proc. of the 11th Int'l Conf. on Parallel and Distributed Computing Systems*, pp. 75–80, 1998.

[11] Grefen P., K. Aberer, H. Ludwig, and Y. Hoffner. CrossFlow: Cross-Organizational Workflow Management for Service Outsourcing in Dynamic Virtual Enterprises. Bulletin of the Technical Committee on Data Engineering, 24(1):53–58, 2001.

[12] Han Y. and A. Sheth. On Adaptive Workflow Modeling. *Proc. of the 4th Int'l Conf. on Information Systems Analysis and Synthesis*, pp. 108-116, 1998

[13] Jablonski S. et al. External and Internal Support Services in Workflow Management Systems. *Proc. of the 11th Int'l Conf. on Parallel and Distributed Computing Systems*, pp. 81–86, 1998.

[14] V. Kumar. Algorithms for Constraint-Satisfaction Problems: A Survey. *AI Magazine*, 13(1):32–44, 1992.

[15] Ramamritham K. and P. K. Chrysanthis. *Advances in Concurrency Control and Transaction Processing,* IEEE Computer Society Press, 1997.

[16] Tsamardinos I. Constraint-Based Temporal Reasoning Algorithms with Applications to Planning. *Ph.D. Thesis. University of Pittsburgh Intelligent Systems Program*, 2001

[17] Tsamardinos I., M. E. Pollack, et al. Merging Plans with Quantitative Temporal Constraints, Temporally Extended Actions, and Conditional Branches. *Artificial Intelligence Planning and Scheduling (AIPS'00)*, Breckenridge, Colorado, USA, 2000

[18] Tsamardinos I., M.E. Pollack, et al. Adjustable Autonomy for a Plan Management Agent. *AAAI Spring Symposium on Adjustable Agents.*, 1999.

[19] Workflow Management Coalition, Technology & Glossary, Document Number WFMC-TC-1011, June 1996.

[20] Q. Yang. Intelligent Planning: A Decomposition and Abstraction Based Approach. Springer, 1997.

Proceedings of the 12th Int'l Wrkshp on Research Issues in Data Engineering: Engineering e-Commerce/ e-Business Systems (RIDE'02)
1066-1395/02 $17.00 © 2002 IEEE