Modeling and Analysis of Interactions in Virtual Enterprises [*]

Hasan Davulcu, Michael Kifer, L. Robert Pokorny
C.R. Ramakrishnan, I.V. Ramakrishnan
Dept. of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794.

Steven Dawson
Computer Science Laboratory
SRI International
333 Ravenswood Avenue,
Menlo Park, CA 94025.

Abstract

Advances in computer networking technology and open system standards are making the creation and management of virtual enterprises feasible. A virtual enterprise is a temporary consortium of autonomous, diverse, and possibly geographically dispersed organizations that pool their resources to meet short-term objectives and exploit fast-changing market trends. For a virtual enterprise to succeed, its business processes must be automated, and its startup costs must be minimized.

In this paper we describe a formal framework for modeling and reasoning about interactions in a virtual enterprise. Such a framework will form the basis for tools that provide automated support for creation and operation of virtual enterprises.

1. Introduction

Advances in computer networking technology and open system standards have made it practically feasible to create and manage *virtual enterprises*. A virtual enterprise [6, 7] is a temporary consortium of autonomous, diverse, and possibly geographically dispersed organizations that pool their resources to meet short-term objectives and exploit fast-changing market trends. Upon realizing the objective, the enterprise can possibly disband. For a virtual enterprise to succeed, it must coordinate many varied tasks and facilitate data sharing among heterogeneous information systems without compromising the proprietary information assets of any individual organization. Network-based virtual enterprise is a powerful paradigm that has the potential to profoundly impact a wide range of business practices.

[*]This work is partially supported by a DLA/DARPA contract and by the NSF grants IRI-9404629, CCR-9705998, 9711386, 9510072 9404921, CDA-9504275, 9303181, INT-9600598

Because virtual enterprises are composed of autonomous entities and created for short-term objectives, their business processes must be automated, and their startup costs must be minimized. Therefore, the very process of creating a virtual enterprise must be automated as much as possible.

In this paper we propose a formal framework based on Concurrent Transaction Logic ($\mathcal{CTR}$) for modeling and reasoning about interactions in a virtual enterprise. This framework will form the basis for a *Virtual Enterprise Management Systems* (VEMS). A VEMS is envisioned as a tool that will enable declarative modeling and automatic enactment of virtual enterprises. Moreover, since the proposed framework is rooted in logic, it will permit reasoning about the intended behavior of virtual enterprises and support verification to ensure that virtual enterprises function as specified. For illustration purposes we focus on task coordination and information interchange in a virtual enterprise.

2. Example of a Virtual Enterprise

The following scenario, drawn from our CASP project[1] experience illustrates a typical situation that would benefit from the creation of a virtual enterprise. A maintenance crew is doing a routine inspection of a rescue helicopter. A defect is discovered in the strut assembly of the landing gear. This component is an assembly of parts that was designed to last for the life of the aircraft and is not available from the maintenance parts depot. The original supplier of the assembly is no longer available to provide the assembly, so an alternate source must be found. Currently this process can easily take more than a year. To expedite the repair,

[1]CASP, the Center for Agile Sources of Parts, manages a consortium of more than 100 manufacturers with experience building military parts. The Defense Logistics Agency uses CASP to identify and contract with these manufacturers for parts that are no longer available from their original sources. At SUNY Stony Brook, we have developed tools to support this identification process [12].

a request for the assembly is made to the Defense Logistics Agency's (DLA) On-Demand Manufacturing Program (ODM). To rapidly supply the needed assembly, the ODM management team assembles a virtual enterprise made up of several collaborating manufacturers, DLA engineers, auditors, and assorted information providers.

The ODM management team first identifies potential manufacturers by matching their capabilities, previously inferred and stored in DLA's knowledge bases, with the technical characteristics of the needed part. Bids are then solicited from the selected manufacturers. Some bids may propose changes to part design to reflect current technology trends. Evaluation of such bids requires assembling an engineering team with competence in the appropriate technology. The engineering evaluation may itself require searching heterogeneous data sources for information on parts built previously using similar technology. Each bid is analyzed and a contract is awarded to the group that best meets the goals of price and timeliness. After the contract is awarded, the ODM management team interacts continually with the manufacturing group to identify and work around problems that may arise as the assembly is being produced.

Observe from the scenario above that in a virtual enterprise, the interactions between entities are inherently complex, since the autonomy of the entities precludes any simplification of the interrelationships. Given the complexity, diversity and short-term nature of virtual enterprises, automation of coordination and information interchange is essential. Without automation, creation and operation of a virtual enterprise is extremely labor intensive and on a large scale is well nigh impossible. To make automation possible we need:

- A formal specification language to specify the structure of entities, processes and their interactions at a high-level.
- Verification methods to ascertain that the virtual enterprise possesses certain key properties that are essential for it to function correctly. For example, in the ODM virtual enterprise above, a key property is: *Any part produced using new technology must always be approved by the engineering team.*
- Techniques for automatically deriving coordination and information interchange mechanisms from the formal description.

Analogous to a Data Base Management System (DBMS) that provides tools for modeling and manipulating large collections of structured data, we envision a Virtual Enterprise Management System (VEMS), providing a comprehensive set of tools for modeling, analysis, and operation of a virtual enterprise. A VEMS, based on a formal framework as outlined above, enables the creation of virtual enterprises that meet their design specifications.

2.1. Enabling Technologies

Several core technologies are needed for the creation and operation of a virtual enterprise. Internet technology and the evolving standards for interoperability are important technologies that support the communication fabric of a virtual enterprise. *Workflow Management* technologies serve the coordination needs, and *Mediation* technologies address the information needs of a virtual enterprise.

We will first examine the issues underlying coordination in a virtual enterprise through the following example drawn from our CASP project.

Example 2.1 (Bid Evaluation). Consider the workflow represented in Figure 1, which represents a simplified process of evaluating a single bid received in response to DLA's request for bids on the landing gear strut assembly.

The workflow represented by the graph consists of two major parts: market evaluation of the bid (nodes B and C) and technical evaluation of the bid (nodes D, E, F, G, H). When both evaluations are completed (as indicated by the AND-node A), a decision is made (node I).

The technical evaluation of a bid can be done either by an internal team of engineers (nodes E, F), by a consultant firm (nodes G, H), or both the internal team and the consultant may need to be involved. These alternatives are indicated by the OR-node D.

Coordinating such a workflow would have been quite straightforward if not for the dependencies that cannot be easily captured by graphs. Specifically, the following constraints, expressed informally, might be applicable in the bid evaluation process:

1. **if** $B.cost > \$1000$ **then** prefer E over G
   This constraint says that if the cost of the part is above \$1000, then task E or task G must be executed (*i.e.*, technical analysis should be performed), but E (internal evaluation) is preferred over G (hiring a consultant).

2. **if** $B.cost < \$1000$ **then not** G
   In other words, do not hire a consultant if the cost of a part is not very high.

3. if $occurs(E) \wedge occurs(G)$ **then** G **before** F
   If it turns out to be necessary to do both the internal evaluation and hire a consultant, then the consultant should finish work before risk analysis gets into full swing.

4. **if** $E.recommendation = consultant$ **then** G
   If our internal investigation concludes that a consultant is needed, then hire one.

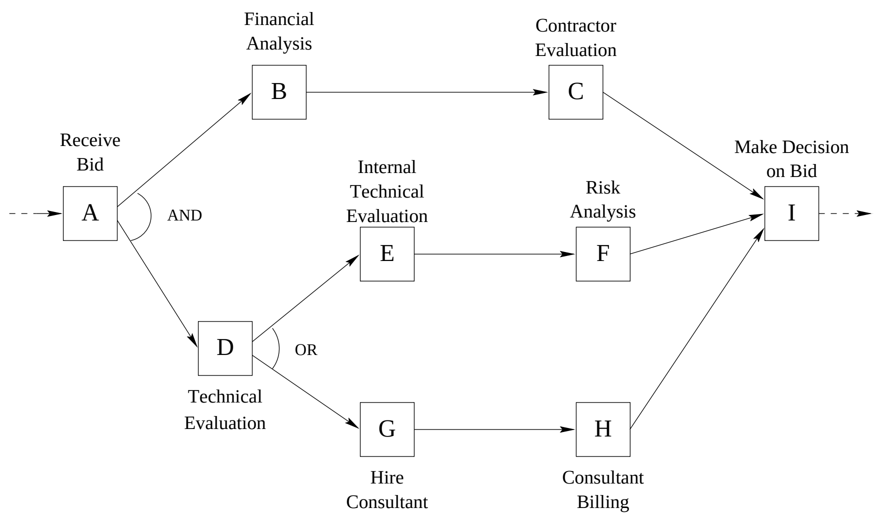

Figure 1. Bid Evaluation Workflow

5. **if** $C.contractorRating = low$ **then not** G
   If the contractor's overall rating was determined to be low, then spending additional funds on a consultant is not justified.

6. **if** $occurs(E)$ **then** E **before** C
   If internal technical evaluation must take place, then do it before evaluating the contractor.

7. **if** *occurs(G)* **then** C **before** G
   Do not hire a consultant before the contractor is fully evaluated.

□

3. Logic-based Framework for Virtual Enterprises

As can be seen from Example 2.1, even in small workflows it might be difficult to fully comprehend all the consequences of the specifications. For instance, are the above constraints consistent? If they are, are they consistent with the task precedence order implied by the graph in Figure 1? Can it happen that certain activities can never be executed (which probably indicates a bug)? Apart from the *consistency* issue, it is important to be able to *reason* about the properties of the workflows. For instance, to assess the correctness of our specifications, we might need to verify properties, such as "is it true that E (the internal technical evaluation of the bid) is always done?" Yet another issue is the efficiency of the coordination process. Of course, we can always select a possible execution and then check if it satisfies the constraints. If it does not, then we can try to find another execution. Unfortunately, this process might force the workflow scheduler to do exponential amount of work and thus is inefficient.

Our contention is that representation of complex enterprises can and should be done using logic as a unifying principle. Furthermore, the very same language should be used for modeling, reasoning, *and* coordination of these enterprises. We have already shown in [5] that a logic-based formalism for workflows is as expressive as any current method. The advantage of using a common logical framework is that scheduling of activities in a virtual enterprise as well as verifying its operational properties naturally reduce to one and the same problem — logical unsatisfiability. Consequently, there is no need to devise distinct, complicated algorithms to deal with these seemingly unrelated tasks.

Example 2.1 illustrates the potential of using powerful logical formalisms, such as $\mathcal{CTR}$, for workflow management. A number of formal approaches have been proposed [14, 15, 11, 2, 8, 1, 4], but, unfortunately, most are incomplete: one approach might be appropriate for modeling workflows, another might be able to reason, and yet another one to schedule. The problem is that it is not easy to get the different approaches to work together (see [5] for detailed comparisons).

It is therefore desirable to find a formalism where all three tasks can be done in a uniform way. For instance, we would like to be able to represent the graph in Figure 1 *and* the constraints of Example 2.1 as a formula, and then use the semantics and the logic's proof theory to decide the properties of the workflow.

An Overview of Concurrent Transaction Logic This section provides a short summary of the $\mathcal{CTR}$ syntax, which is used in this paper to represent workflows. Due to space limitation, we cannot discuss the model theory of the logic or its proof theory. Instead, we rely on the procedural reading of $\mathcal{CTR}$ statements.

Underlying the logic and its semantics is a set of database *states* and a collection of *paths*. A *path* is a finite sequence of database states. For instance, if $s_1, s_2, ..., s_n$ are database states, then $\langle s_1, s_2, ..., s_n \rangle$ is a paths of length n. Just as in classical logic, $\mathcal{CTR}$ formulas assume truth values. However, unlike classical logic, the truth of $\mathcal{CTR}$ formulas is determined over paths, *not* at states. If a formula, ϕ, is true over a path $\langle s_1, ..., s_n \rangle$, it means that ϕ can *execute* starting at state s_1. During the execution, the current state will change to s_2, s_3, ..., etc., and the execution terminates at state s_n. With this in mind, the intended meaning of the $\mathcal{CTR}$ connectives can be summarized as follows:

- $\phi \otimes \psi$ *means*: execute ϕ then execute ψ. In terms of control flow graphs (cf. Figure 1), this connective represents arcs connecting adjacent tasks.

- $\phi \mid \psi$ *means*: ϕ and ψ must both execute concurrently, in an interleaved fashion. This connective corresponds to the "AND"-nodes in control flow graphs.

- $\phi \wedge \psi$ *means*: ϕ and ψ must both execute along the *same* path. In practical terms, this is best understood in terms of *constraints* on the execution. For instance, ϕ can be thought of as a transaction and ψ as a constraint on the execution of ϕ. It is this feature of the logic that lets us specify temporal constraints as part of workflow specifications.

- $\phi \vee \psi$ *means*: execute ϕ *or* execute ψ non-deterministically. This connective corresponds to the "OR"-nodes in control flow graphs.

- $\neg\phi$ *means*: execute in any way, provided that this will not be a valid execution of ϕ. There are many uses for this feature. One is that, just as in classical logic, the negation lets us define deductive rules which, in terms of the workflows, correspond to sub-workflow definitions. Negation is also an important component in temporal constraint specifications.

Example 3.1 (Re-visiting Bid Evaluation). The following is a representation of the control graph in Figure 1 in the language of $\mathcal{CTR}$, where $A \hat{\vee} B$ means that one of the two actions represented by A must execute (or, perhaps, both must execute together, in parallel.

$$\begin{array}{l} Bid\_Eval \leftarrow \\ A \otimes \big((B \otimes C) \mid D \otimes \big((E \otimes F) \hat{\vee} (G \otimes H)\big)\big) \otimes I \end{array} \quad (1)$$

The following is a representation of the coordination dependencies #2 to #7 in the language of $\mathcal{CTR}$,where ∇E means that action E occurs somewhere on the execution path. (∇ is not a new operator in Transaction Logic; it can be expressed through other logical connectives).

2. $\nabla[B.cost < \$1000] \rightarrow \neg\nabla G$
3. $\nabla E \wedge \nabla G \rightarrow \nabla G \otimes \nabla F$
4. $\nabla[E.recomm = consultant] \rightarrow \nabla G$
5. $\nabla[C.contractor = low] \rightarrow \neg\nabla G$
6. $\nabla E \rightarrow \nabla E \otimes \nabla C$
7. $\nabla G \rightarrow \nabla C \otimes \nabla G$

However, expressing constraint #1 is more involved and requires the necessity modality, "$\Box$," and the non-monotonic aspects of $\mathcal{CTR}$. In modal terms, $\Box\phi$ means that ϕ is the only transaction that can succeed from the present state. The following $\mathcal{CTR}$ expression represents the preference constraint #1.

1. $\nabla[B.cost > \$1000] \rightarrow (\nabla E \ \vee \ (\Box\neg\nabla E \otimes \nabla G))$

Formally, this means: if $[B.cost > \$1000]$ then either execute E or, if this is not possible, execute G.

Once constraints and the graph are specified in the logic, the entire workflow can be represented as:

$$Bid\_Eval \ \wedge \ (\wedge_{i=1}^{7} Constr_i) \quad (2)$$

Representing workflow control structure as logical formulas opens up a host of possibilities. For instance, it is now possible to prove formally that the specifications are consistent (*i.e.*, there is at least one valid schedule), that event E (internal technical evaluation) always occurs, and that if it is necessary to hire a consultant (*i.e.*, G occurs), then contractor evaluation must finish before doing risk analysis (*i.e.*, C must happen before F). An even more important question is, will every bid evaluation workflow reach a decision stage (node I)? It is not immediately obvious that the latter is *not* guaranteed![2]

Another interesting consequence of the logical representation of workflows is that there is a close relationship between proving a formula like (2) and run-time scheduling of the corresponding workflow. Namely, valid schedules are by-products of proving such formulas. As a result, there is a direct relationship between the complexity of finding a proof and the run-time cost of finding a schedule!

This leads to the following schedule optimization. Suppose we can transform the workflow representation (2) into an equivalent formula but one that has a more efficient proof. Then, in view of the above discussion, we can obtain a more efficient run-time scheduling algorithm for (2). This line of research was pursued in [5], where it was shown that (2) can be transformed into equivalent formula of the form

[2]Indeed, suppose that the bid prices the job under \$1000 and activity E recommends hiring a consultant. However, the latter is prevented by Constraint 4.

$\Psi \wedge Constr_1$, where Ψ has a much more efficient proof than $Bid\_Eval \wedge (\wedge_{i=2}^{7} Constr_i)$.

In other words, constraints 2 to 7 can be "compiled away" and never need to be checked by the workflow scheduler at run time. Constraint 1, which expresses a preference relation, is more difficult to handle and is a subject of further research. □

4. Information Interchange

So far we have illustrated our approach by modeling and reasoning about coordination requirements in a virtual enterprise. Note though that there are other types of interactions in a virtual enterprise, notably interchange of information which is the topic of this section.

As discussed in Section 2.1, the information needs of a virtual enterprise are best addressed by *mediation* technologies. Although much work has been done in the development of mediation techniques for heterogeneous information systems, there are particular issues in both security and semantic mediation that arise in virtual enterprises, which remain to be addressed. Below we describe these issues and outline our approach to resolving them.

paragraphSecurity Mediation in Virtual Enterprises A key aspect of information interchange in a virtual enterprise is the need for entities to share possibly sensitive or proprietary information without compromising the security of other information. For an information consumer in the enterprise, the main concern is to have timely access to all required information. At the same time, the information provider[3] is most concerned with shielding its proprietary information from unauthorized access. The following example illustrates these issues.

Example 4.1 (Component Manufacturing). Consider the manufacturing phase of the ODM scenario (Section 2). The bid process has been completed, and the virtual enterprise is now engaged in producing the replacement landing gear. One manufacturer, Entity A is responsible for producing a component strut, which will be integrated into a larger strut assembly by another manufacturer, Entity B. The manufacturing plan calls for Entity B to certify to the management team that the entire strut assembly meets its specifications. To make this certification, Entity B requires the quality assurance report on the component strut from Entity A. Entity A's quality report details critical properties and testing results for the strut at various stages in the manufacturing process. While this information is needed by Entity B to certify the strut assembly, the report also reveals details of a proprietary manufacturing process used by Entity A. To satisfy Entity B's requirements while protecting its own proprietary information, Entity A agrees to give Entity B access to the quality report, provided that Entity B does not disclose proprietary information from the report to third parties. □

[3]Note that a single entity in a virtual enterprise may act as both an information consumer and an information provider.

Thus, the primary goal of security mediation in a virtual enterprise is to ensure that *all* and *only* the information needed by other entities be made available to them. To achieve this goal, we must first cope with the fact that a virtual enterprise is composed of autonomous entities that may employ a wide range of incompatible security mechanisms. More specifically, an effective solution for security mediation in a virtual enterprise must address the following areas of heterogeneity:

- *Security interfaces:* Different organizations use different mechanisms for communication, identification, and authentication.
- *Security policies:* Different organizations may formulate their security policies in terms of different authorization models and enforce access control at varying levels of granularity.

Moreover, a virtual enterprise brings an added dimension in that security considerations become intertwined with the flow of information throughout the enterprise, as the nondisclosure requirement in Example 4.1 indicates. Indeed, it is the interaction between security policies and information dissemination in a virtual enterprise that is the focus of our proposed research in security mediation.

Approach to Security Mediation Given that the autonomy of entities in a virtual enterprise precludes internal reorganization, the best option for integrating them for secure information interchange is to *wrap* them. Such a wrapper will provide a bridge between the system-specific interfaces of individual entities and a uniform external interface that enables interaction with other wrapped entities. The capabilities of a virtual enterprise wrapper specific to security mediation will include:

- A uniform interface for secure communications (e.g., SSL), identification, and authentication.
- Mapping between entity-specific security models and an enterprise-level security model.
- Support for fine-grained access control.

Furthermore, virtual enterprise wrappers will be specified at a high level in a uniform logical framework that supports their automated generation.

The idea of using wrappers (and even wrapper generators) to integrate heterogeneous systems is not new; it has been explored previously, e.g., in [10, 9]. In addition, our ongoing research effort at SRI on *Secure Access Wrappers* (SAW) involves the development of wrapping techniques for integrating multilevel secure (MLS) databases in high-assurance information systems. The novel aspects of wrappers in a VEMS context are: (1) the specification of wrappers that manage both security and semantic heterogeneity in a uniform logical framework; and (2) support for automated generation of such wrappers that are guaranteed to meet their specifications.

Security Constraints in a Virtual Enterprise At a high level, information interchange in a virtual enterprise involves three distinct considerations: (1) semantic interrelationships among the data of different entities, (2) the local security policies, and (3) the requirements of information dissemination among the entities. Often considerations (2) and (3) will conflict. Detecting and resolving such conflicts is crucial to proper functioning of a virtual enterprise and requires analysis of the *interactions* among all three of the above considerations.

For illustration, consider again the situation in Example 4.1. Since Entity B depends, in part, on Entity A's quality report for certification of the strut assembly, Entity B's information model includes a semantic relationship that captures the inclusion of information from Entity A's quality report in Entity B's own certification report. In addition, Entity B's workflow specifies the transmission of its certification report to the management team. In the absence of other information, it can be deduced that Entity B will forward (parts of) Entity A's quality report to the management team. Note that this release of information would be in conflict with Entity A's nondisclosure constraint on Entity B. *Detecting* this conflict requires knowledge of the semantic relationship, the control and information flow, and Entity A's security policy. *Resolving* the conflict requires either a relaxation of Entity A's nondisclosure constraint, or a new constraint on Entity B's release of its certification report (to remove the proprietary portions of Entity A's quality report).

Recall that the primary goal of security mediation in a virtual enterprise is to permit all and only required information to be shared. Implicit in this goal is the need to limit exposure of any entity's security policy (since otherwise it may be possible to draw inferences pertaining to sensitive or proprietary information). This need to protect individual security policies further implies that detection and resolution of security conflicts in the virtual enterprise, which is inherently global (enterprise-wide), should be carried out *locally* and in a distributed manner.

We briefly outline an approach for solving this problem. During the formation of a virtual enterprise, each entity develops a specification of its information holdings, information requirements, security policy, and workflow. From this specification, a set of *access requirements* is automatically deduced, detailing what access to information (from other entities) will be needed. A negotiation phase ensues in which each entity requests the needed access rights from others. Each request for an access right is either approved or denied by the target entity, based on its own security policy and access rights granted by others. Observe that a solution to this problem will likely involve an intricate protocol that must be verified. A logical framework will greatly facilitate the specification and verification of such a protocol.

As with workflow coordination, automated support for specification, reasoning, and enforcement of security constraints in a virtual enterprise is essentially a problem of logical inference. Moreover, since security policies interact with process rules, reasoning about these interactions within the same logical framework proposed for workflow coordination is the best approach to security mediation in a virtual enterprise environment.

5. Towards a Prototype Virtual Enterprise Management System (VEMS)

As a proof of concept, we are currently building a prototype VEMS tool kit for modeling and enacting virtual enterprises. A primary purpose of this effort is to verify our ideas in practice and use it CASP, which is an on-demand manufacturing venture.

We envision our prototype to have a user-friendly graphical design tool, which would permit the user to specify workflows and mediators at a high level. The graphical tool will be structured such that graphical and textual specifications can be intermixed (*e.g.*, complex temporal and transition constraints, or complex semantic mappings between data sources could be specified textually). We have already begun implementing a cross-platform workflow design tool. At present, this tool can specify complex control flows, and we are now working on the specification of a protocol that will enable it to communicate with the logical subsystem at the semantic level.

The virtual enterprise system infrastructure will be supported by XSB, the logic-based deductive engine that implements $\mathcal{CTR}$. The XSB system is a deductive engine developed here at Stony Brook. XSB is our choice for several reasons: it is currently known as the most efficient implementation of deductive databases that outperforms other similar systems by one to two orders of magnitude [13]; it extends logic programming with higher-order programming (HiLog [3]); it provides support for non-monotonic reasoning (through its support for well-founded semantics for negation); and it incorporates special indexing structures that considerably simplify the implementation of $\mathcal{CTR}$. Furthermore, XSB is well-integrated into the overall computing infrastructure. It runs on most platforms (including Windows and the various flavors of Unix), it interfaces to database systems through ODBC drivers, has a Perl interface, and Java interface is currently under development. XSB has been installed in over a thousand sites around the world. More information on XSB can be found at `http://www.cs.sunysb.edu/˜ sbprolog`.

References

[1] N. Adam, V. Atluri, and W. Huang. Modeling and analysis of workflows using petri nets. *Journal of Intelligent Information Systems*, 10(2):131–158, March 1998.

[2] P. Attie, M. Singh, A. Sheth, and M. Rusinkiewicz. Specifying and enforcing intertask dependencies. In *Intl. Conference on Very Large Data Bases*, 1993.

[3] W. Chen, M. Kifer, and D. Warren. HiLog: A foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, February 1993.

[4] P. Chrysanthis and K. Ramamritham. ACTA: A framework for specifying and reasoning about transaction structure and behavior. In *ACM SIGMOD Conference on Management of Data*, pages 194–203, May 1990.

[5] H. Davulcu, M. Kifer, C. Ramakrishnan, and I. Ramakrishnan. Logic based modeling and analysis of workflows. In *ACM Symposium on Principles of Database Systems*, pages 25–33, Seattle, Washington, June 1998.

[6] A. Dewey et. al. The impact of NIIIP virtual enterprise technology on next generation manufacturing. In *Proceedings of Conference on Agile and Intelligent Manufacturing Systems*, 1996.

[7] C. Gilman, M. Aparicio, J. Barry, T. Durniak, H. Lam, and R. Ramnath. Integration of design and manufacturing in a virtual enterprise using enterprise rules, intelligent agents, STEP, and workflow. In *Proceedings of SPIE Vol. 3203*, pages 160–171, 1997.

[8] R. Gunthor. Extended transaction processing based on dependency rules. In *Proceedings of the RIDE-IMS Workshop*, 1993.

[9] L. Haas, D. Kossmann, E. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB '97)*, 1997.

[10] J. Hammer, H. Garcia-Molina, S. Nestorov, R. Yerneni, M. Breunig, and V. Vassalos. Template-based wrappers in the TSIMMIS system. In *Proceedings ACM SIGMOD International Conference on Management of Data (SIGMOD '97)*, pages 532–535, 1997.

[11] M. Orlowska, J. Rajapakse, and A. ter Hofstede. Verification problems in conceptual workflow specifications. In *Intl. Conference on Conceptual Modelling*, volume 1157 of *Lecture Notes in Computer Science*, Cottbus, Germany, 1996. Springer-Verlag.

[12] R.Hopkins, D.Warren, A.Kahn, I. Ramakrishnan, J.Jones, M.Kifer, T.Swift, and L. Pokorny. CASP: A virtual parts supplier. In *Proceedings of International Conference on Agile Manufacturing*, 1997.

[13] K. Sagonas, T. Swift, and D. Warren. XSB as an efficient deductive database engine. In *ACM SIGMOD Conference on Management of Data*, pages 442–453, New York, May 1994. ACM.

[14] M. Singh. Semantical considerations on workflows: An algebra for intertask dependencies. In *Proceedings of the International Workshop on Database Programming Languages*, Gubbio, Umbria, Italy, September 6–8 1995.

[15] M. Singh. Synthesizing distributed constrained events from transactional workflow specifications. In *Proceedings of 12-th IEEE Intl. Conference on Data Engineering*, pages 616–623, New Orleans, LA, February 1996.