Engineering Process Ontologies for Communication, Co-operation, and Co-ordination in a Virtual Enterprise

M. Mäntylä and M. Ranta
Helsinki University of Technology
Otakaari 1, FIN-02150, Espoo 15, Finland, phone: +358-9-451 3230, +358-9-451 4807, fax: +358-9-451 3293, e-mail: Martti.Mantyla@hut.fi, Mervi.Ranta@hut.fi

Abstract

To work together, members of a virtual organisation must be able to establish co-operation, exchange information with each other, and co-ordinate shared activities across open interfaces with strong support from information and communications technologies. This paper promotes the concept of a *process ontology* as a groundwork for communication, co-operation, and co-ordination for virtual enterprises and teams. We discuss the background of our work and related activities in ontology development, give a scenario of a future mode of operation for a virtual enterprise, and describe our work in progress aimed at visualising and prototyping shared engineering ontologies and using them as a basis of agent-based process co-ordination across the Internet.

Keywords

Engineering processes, integration, virtual enterprise, virtual organisation, agents, ontologies, Internet

1 INTRODUCTION

A long-standing research area in engineering applications of computing is the integration of heterogeneous systems such CAD, CAM, operations management, product data management, and workflow management. Various methods, tools, and techniques have been identified in this work, notably standards for product data representation (STEP and related ISO standards) and protocols for accessing such

data (SDAI), and various tools for electronic commerce (such as EDI). The area has also proved attractive for methods and tools developed elsewhere, such as in distributed computing (CORBA, DCOM, Java), computer-supported co-operative work (CSCW), and artificial intelligence.

Yet integration of engineering systems has proved a tough nut to break. Although this difficulty must be attributed to several sources, one major culprit for the limited success is that the actual engineering processes that underlie the systems are unknown, badly understood, or complex — or most often, all three of these.

At the present, the Internet and its technologies are rapidly changing the basic platform of information and communications technologies (ICT) that forms the basis of process integration. With the rapid emergence of Internet-savvy CAD, PDM, document management, workflow management, and other engineering systems, the technological prerequisites for engineering process integration are rapidly becoming a reality.

The work described in this paper is based on the hypothesis that even with these improved tools, engineering process integration will remain elusive unless the underlying, shared knowledge of engineering processes can be made explicit, visible and accessible by its participants, and applicable to tailor and control engineering systems and system integration frameworks. In other words, we propose that integration must be based on an explicit and shared *domain ontologies* for engineering processes — the shared nomenclatures describing products, their structure in terms of subsystems and components, the engineering activities that build them, the resources and actors that are needed to implement the activities, the major events and checkpoints occurring during the evolution of the process, and the various relationships between such ontology entities.

This conceptual paper is structured as follows: First, we discuss the concepts and background of our work and describe briefly some related activities in ontology development in section 2. Based on this, section 3 will specify the scope and objectives of our work. Next, section 4 gives a scenario of a future mode of operation of a virtual enterprise facilitated by shared ontologies and related tools. Section 5 follows with a brief description of our work in progress aimed at visualising and prototyping shared engineering ontologies and their capability as a basis of agent-based process co-ordination across the Internet.

2 BACKGROUND AND RELATED WORK

The basic approach of our paper is to combine ideas and methods originating from three directions of research, namely *product modelling*, *ontologies for knowledge sharing*, and *enterprise integration*. All of these are broad research areas of their own, and we only can indicate the most central sources of our inspiration in this section.

2.1 Product Modelling

Product modelling aims at creating computerised product representations that capture various aspects of product data such as its geometry, engineering attributes, manufacturing data, or bill-of-materials data. During the last decade or so, the focus of product modelling has moved from relatively low-level representations of product geometry to higher-level issues such as preservation of design intent, capturing various viewpoints to product data such as those of design and manufacture, and life-cycle management of product data. Many of these issues have been investigated under the title Knowledge Intensive CAD (KIC) as witnessed by the recent series of IFIP workshops bearing the name.

A direct precursor of the work reported in this paper was the Main Joint Demo (MJD) experiment performed as a part of the GNOSIS Test Case of IMS. As reported by Ranta et al. (1995), the experiment focused on sharing product data across several independently developed systems representing different product life-cycle stages. MJD implemented product data sharing through a conceptual map between the heterogeneous product representations utilised by the various systems. MJD showed us the potential power of the high-level mapping approach, in contrast to the more conventional sharing of neutral low-level product data.

To preserve design intent, product modelling community has investigated approaches such as variational and feature-based product models (Shah and Mäntylä 1996) and design history oriented models that relate the incremental evolution of product data with an explicit design process model. Our own past experience in this direction was reported by Lahti et al. (1996). Meanwhile, research on product data management has studied the same issues from somewhat different perspective. This research studies the evolution of product data using concepts such as versioning, configuration management, and change propagation.

Summing up, in product modelling research it has transpired that proper articulation of product evolution must be based on a reasonably fine-grained model of the engineering process that originates the changes. Symmetrically, to articulate properly the semantics and inter-dependencies of engineering process steps, a model of the design data used or created by the steps is needed. We conclude that development of integrated product and process models a topic of high current interest in product modelling.

2.2 Ontologies and Knowledge Sharing

The concept of an *ontology*, as used in this paper, was originally introduced in artificial intelligence research for sharing knowledge between independently developed knowledge-based systems. A particularly significant contributor was the DARPA Knowledge Sharing Effort conducted in the USA during early 90's. It developed several tools intended to facilitate knowledge transfer across heterogeneous systems, including the Knowledge Interchange Format (KIF, see X3T2 (1995)), a human-readable knowledge representation language based on first-

order logic and some second-order capabilities, the Knowledge Query and Manipulation Language (KQML, see Finin et al. (1994)), a transport mechanism intended to support the development of agent command languages for Internet agents, and the Ontolingua language intended for describing shared ontologies. A recent significant effort in knowledge sharing is the Open Knowledge Base Connectivity (OKBC) proposal by Chaudhri et al. (1998).

Of these efforts, the work on ontology development and sharing is particularly relevant for the scope of this paper. Tom Gruber (1993), the father of Ontolingua, defines simply: *"An ontology is a specification of a conceptualisation … That is, an ontology is a description of the concepts and relationships that can exist for an agent or a community of agents … for the purpose of enabling knowledge sharing and reuse."*

So, a shared ontology is intended to form the basis of communication by giving a meaning to the terms and structures used in the discourse between some communicating partners. It also represents a commitment made by the partners to obey the defined constraints, relationships, and semantics in their communication.

Ontolingua provides a KIF-based language and a system for describing ontologies in a form that is compatible with multiple representation languages. In particular, it provides a standard declarative language with forms for defining classes, relations, functions, objects, and theories. It translates such definitions into the forms of supported representation systems. Therefore, ontologies written in Ontolingua can be shared by multiple users and research groups using their own representation systems, and ported from system to system.

The present collection of available Ontolingua ontologies covers quite specialised areas such as abstract algebra, chemical elements, mechanical components, and simple geometry. As we can see from these examples, Ontolingua ontologies typically describe knowledge of a domain rather than the process in which the knowledge is created or used.

2.3 Enterprise Modelling

Enterprise modelling using informal graphical models such as IDEF-0 is now common industrial practise in areas such as Business Process Re-engineering. IDEF-0 is also used in STEP application protocol specifications to clarify their scope and information requirements. In both cases, the models are intended for documenting the processes and communicating their structure to human readers. We believe that future expansion and use of STEP must be based on more formal, machine interpretable process models.

Process modelling specifically for facilitating process integration is addressed by enterprise integration frameworks such as the CIM Open Systems Architecture (CIMOSA) developed in a number of European projects, the GRAI GIM model developed over the years at the GRAI laboratory of University of Bordeaux, and the PERA model developed at Purdue University. Bernus, Nemes and Williams (1996) give a useful introduction to these models, while Bernus and Nemes (1996) give a

view of the depth and breadth of current work. These activities do not directly address the issues arising in distributed virtual enterprises — the area where we aim to concentrate our efforts.

Recently, several research groups have proposed approaches where the knowledge sharing methods of AI are utilised for enterprise and process modelling.

A notable example is the Enterprise Ontology developed at the University of Edinburgh by Uschold et al. (1998). Available in Ontolingua form, this interesting ontology defines concepts such as activities, processes, organisational units, and strategies, and various relations amongst these and more specialised concepts derived from these. The ontology is mainly aimed at enterprise modelling and integration.

The Toronto Virtual Enterprise (TOVE) project at University of Toronto (see Fox and Gruninger 1994) also aims to develop an ontology that can be used for enterprise integration. More specifically, the ontology provides a shared terminology that every application can jointly understand and use; defines the semantics of each term as precisely and unambiguously as feasible; implements the semantics in a computable form; and provides a graphical notation for facilitating human comprehension. The present TOVE ontology appears to be oriented towards providing a basis of agent interaction and simulation of logistics systems.

3 VIRTUAL ENTERPRISES AND PROCESS ONTOLOGIES

The practical motivation of our work stems from an emerging new paradigm of enterprise operation: the *virtual enterprise*.

In its purest form, a virtual enterprise is a dynamically created entity composed of a unique collection of collaborating partners for the purpose of satisfying a unique customer need, and dissolving itself after its purpose has been fulfilled. Such an organisation can be competitive if it is capable of tapping fully into the competence of its partners, and if it can apply best practices within the limits of these competencies. To allow this, best practice processes must be made visible and available to the dynamic consortium that forms a virtual team. In addition, the processes must be supported by and augmented with applicable ICT tools to reduce the interaction costs to an acceptable level and to facilitate the communication of process participants from different disciplines and partner companies.

Much of the existing work on virtual enterprise computing has concentrated on the development of electronic markets. For the time being, this research has addressed relatively straightforward enterprise operations such as order management of commodity components or logistical operations. Even so, the development of generally understood basic concepts has proved difficult. For instance, even the deceptively simple concept of an "order" has many interpretations that make achieving truly electronic commerce a challenge.

The main hypothesis and source of motivation of our work is that *future progress in virtual enterprise computing must be based on formal and shared models of the*

shared processes and shared data of the co-operating partners — in short, *shared ontologies*.

The need of formal ontologies is evident if we study the requirements of more complex co-operative contacts between virtual enterprise partners than simple commodity market relationships. In particular, we focus our interest on engineering processes occurring between partners of a virtual enterprise, and hence the development of *engineering process ontologies*. This focus is consistent with the present trend towards increased specialisation of companies, necessitating co-operation in product development and engineering.

A shared ontology for engineering processes should codify concepts and connotations for describing products, engineering tasks, organisations responsible for carrying out the tasks, main events and checkpoints occurring during the development, and all types of relations, constraints, and axioms that may be applicable for these basic process entities.

Examples of scenarios the belong to our scope of interest range from relatively simple design subcontracting relationships to complex distributed design scenarios where complex change management, configuration control, and versioning may be needed. Section 4 will give an illustrative example of one such scenario.

To be useful, an engineering process ontology need not be all-encompassing or globally applicable. Nor does it need to be perfectly defined before it can be used as a basis of communication. On the contrary, we believe that during their life-cycle useful ontologies for engineering are born to cover specialised domains, grown in scope, mutated to cover further activities, split in parts to reuse their best practices in novel areas, and merged again — in short, they are adapted to varying business conditions and needs.

4 SCENARIO

To illustrate the concept of engineering process ontology, let us give a scenario of a shared engineering work process. This scenario is based on our observations of several industrial companies and discussions with their personnel, while not being based on any single company.

Large engineering projects, such as construction of chemical plants or various items of urban infrastructure, require the contribution of several independent engineering and manufacturing companies that provide engineering, planning, and logistics services; modules, parts, or materials; or actual construction work for the project. Typically, the scope of these projects is large, covering several fields of engineering (mechanical, chemical, electrical, construction); there are complex dependencies between the various engineering, manufacturing, and construction activities; the overall schedule is tight, with a rigid deadline; the logistics is complex; and various types of engineering changes are expected to occur during the progress of the project that must be contained by dynamic redesign and rescheduling.

Various types of computer-aided tools have been developed to address some of these issues. Prime examples include various project management tools, logistics tools, and CAD tools. Unfortunately, from the practical viewpoint, these tools are quite separate, and cannot be operated in an integrated fashion. This problem is made even worse by the inherent parallelism and inhomogeneity of the underlying engineering processes and data: data needed for a good scheduling decision may not be timely available to the decision maker, because many of the related activities are performed at other companies. Similarly, the data needed for a sound engineering decision may not be available to a designer working on a subsystem of the entire product, possibly leading to the problems of incompleteness, invalidity, and inconsistency.

The fundamental source of these problems is that the engineering processes and issues dealt with them are not local, but form a tangled web covering the entire project consortium. This makes multi-supplier engineering projects an interesting scope to study virtual enterprises and their computational infrastructure.

Clearly, to effectively deal with all engineering issues, a close linkage between the activity models of a project (typically used for project management) and the artefact models related to the actual engineering objects created and constructed by the project (typically the domain of CAD/CAM software) must be achieved. Moreover, instead of a single global activity-artefact model covering the whole project, a genuinely distributed model is required to match with the distributed and autonomous nature of a project consortium.

We hypothesise that agent technologies give the most attractive approach to satisfy these requirements. This suggests an architecture where each partner is represented by an agent that makes its capabilities and requests visible to other agents. In addition, agents can also encapsulate engineering systems to provide them an interface for data exchange and conversion. A special broker agent provides a marketplace for the agents, and facilitates establishing communication between them.

This suggests a methodology that can be described as follows:

- During the planning stage of the project, an initial rough model of the entire project is created as a side effect of project planning. This model includes the rough activities to be performed and their inputs and outputs in terms of artefacts used or created.
- The project model is "published" through the agent-based co-operation architecture. After a negotiation and contract-making phase is completed, each agent has created its own initial model of the activities to be performed by that agent by replicating appropriate entities of the rough model.
- During the execution of the project, agents trace the local evolution of the project by creating more detailed model entities. The evolution of activities and artefacts is tracked by means of an appropriate life-cycle model. Life-cycle transitions are registered to other interested agents by change propagation.

- The project model can be investigated to determine, for instance, whether the project is on schedule or what consequences a certain disturbance could have.

In summary, the methodology integrates a time-oriented model of the project (activity schedule, events) with engineering-oriented models of the artefacts (product models, documentation, etc.) in a unified whole.

To illustrate the suggested methodology, let us give a "storyboard" which relates to a scenario where four companies participate in the execution of a project: the project manager company, two manufacturing companies, and one construction company.

Initially, the project manager creates a rough activity model of the whole project with identifications of artefacts created and consumed by the various activities. Project manager registers the created model to his agent (Figure 1). The broker is aware of the other agents representing the partners relevant to the project.

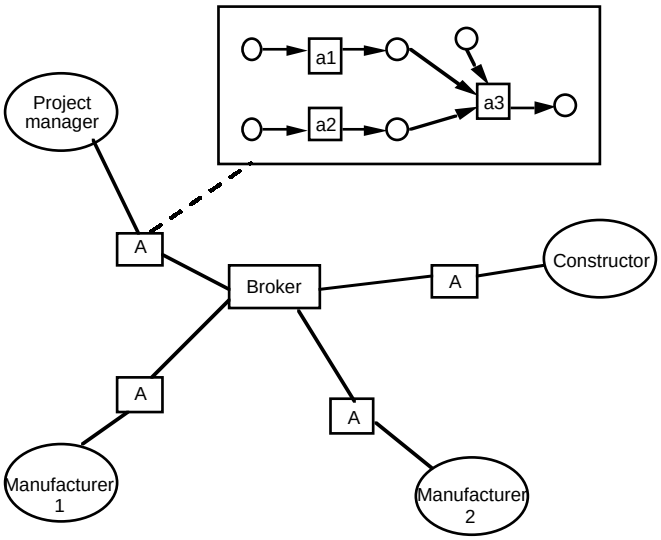

Figure 1 The initial activity model is registered

A matchmaking and negotiation process follows. As a result, activities of the rough model are allocated to partners and the appropriate entities of the rough model are replicated (Figure 2). During the execution of the project, the life-cycle evolution of activities and artefacts is tracked by the agents. In this process, some entities may be decomposed into more detailed ones, as in the case of the entity `a1` in the figure. To maintain global consistency of the model, changes are propagated between agents through replicated model entities.

The specific change propagation protocols are likely to vary according to the more specific types of the activities and artefacts in question. Sometimes, all changes in lower-level entities must be notified to other partners; sometimes, they are of no interest outside the domain of the agent performing the changes.

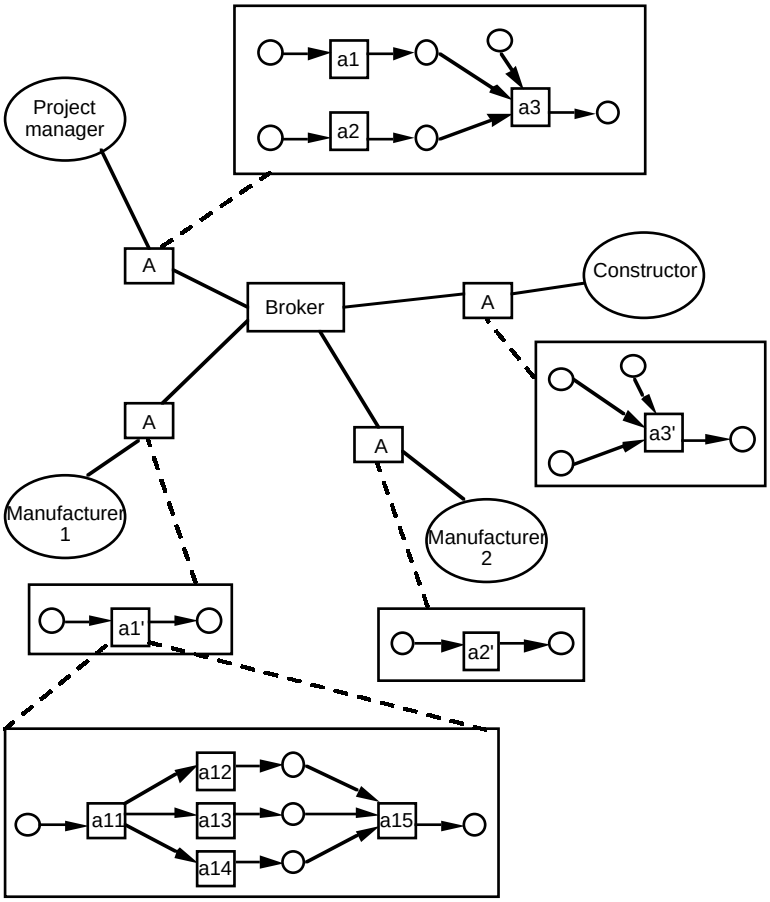

Figure 2 Activities are allocated to partners

5 WORK IN PROGRESS

The longer-term objective of our work is the development of a computational framework for engineering process integration in virtual organisations. To explore the issues related to any such framework, and their potential resolutions, we are presently developing a concept prototype system called *A3E* (standing for *Artefacts-Activities-Actors-Events*).

To capture engineering process scenarios such as the one discussed in the preceding section, the framework must provide conceptually rich enough process and product models. Moreover, the model concepts must be shared by the participants — that is, they must form a *shared domain ontology*.

To be useful for mediating between several autonomous partners, the framework must support communication, co-ordination and co-operation (C^3) amongst the

participants of a distributed engineering process on the basis of minimal compatibility assumptions and precompiled knowledge.

To satisfy these requirements, we designed A3E to consist of two parts:

- a model framework that combines models of engineering artefacts and processes in a single domain ontology
- a co-ordination environment that supports virtual enterprise creation and distributed dialogues on ontology entities on the basis of Internet agents.

5.1 A3E Ontology

To be able to exchange information at all, some minimal shared ontology between communicating partners must be assumed. In our work, the *A3E ontology* fulfils this role.

As the name suggests, the A3E ontology is based on the following base entities and their taxonomic siblings:

Artefact: Artefacts are the things used as the input or created as the end result of the tasks of an engineering process. Artefacts form a taxonomy of different types of things, such as specifications, test plans, user manuals, software modules, and CAD models.

Activity: Activities represent the actual work being performed during engineering. An activity is a temporal thing: it starts and ends somewhere in (known or unknown) time. It is also a process in that it consumes input artefacts and resources, and produces output artefacts.

Actor: Actors are the things that make activities happen; they model the people and organisations that really do the engineering work. They are also used to model other things useful to have such as money, physical space, or machines. Unlike artefacts, which are "short-lived" things, actors are thought of as fairly rigid things that evolve only slowly (as compared to the duration of activities).

Event: The preceding concepts can be used to capture a static snapshot of the state of an engineering process. The (discretised) evolution of the process, therefore, is a sequence of such snapshots. Events denote the signals causing transitions from one such state of the entire process to a next state. Events may originate from the actors modelling human users, or from side effects of other events during event propagation as discussed below.

In addition to these basic entities, the base ontology has *relations*, modelling various sorts of dependencies between the entities. As the main entities, relations too form a taxonomy.

5.2 Life-Cycle Models

The main purpose of events is to grasp the evolution of engineering processes and the related entities. Therefore, all artefacts, activities, and actors have a *life-cycle model* associated to them. A life-cycle model can be thought of as a finite state machine which is associated to an entity. Transitions between life-cycle states are

controlled by incoming events. A fired transition may cause further events to be created and propagated to related entities.

For instance, the artefact `product` might have the following life-cycle states:

`Initial` - the need for the existence of the product has been recognised and it is instantiated

`Planned` - the product has a defined project associated to it

`Specified` - the product has a committed specification document

`Designed` - the product has a committed design document

`Engineered` - the product has a committed engineering document

`Complete` - the product has a committed process plan.

We may observe that the states are partially defined in terms of the analogous states of related documents. Indeed, the life-cycle model of the artefact `document` has its own life-cycle states:

`Initial` - the need for the document is recognised

`Defined` - the role of the document is defined in relation to a product

`Contents-assigned` - the contents of the document has been created

`Committed` - the document has been declared ready

`Accepted` - the document is accepted.

Figure 3 gives a definition of the life-cycle model of `product` including the above states using the Lisp list notation of our implementation. Each state is followed by a list of transitions; in this case, each state has just one transition leading to the next state. The transitions are represented as a list consisting of the name of event causing the transition (e.g., `define-created-by`), name of the next state, and a predicate that controls when the transition is enabled.

```
((initial
   (define-created-by planned
     (or (defined? (value self 'created-by))
         (dec-defined? (value self 'created-by)))))
 (planned
   (commit-spec specified
     (committed? (value self 'specified-in))))
 (specified
   (commit-design designed
     (committed? (value self 'designed-in))))
 (designed
   (commit-engineering engineered
     (committed? (value self 'engineered-in))))
 (engineered
   (commit-process-plan complete
     (committed? (value self 'is-process-planned-in))))
 (complete))
```

Figure 3 Life-cycle model of `product`

Each artefact, activity, and actor includes a notification engine that handles incoming events, fires enabled transitions, and propagates events to related artefacts, activities, and actors. An important aspect of the notification engine *is event mapping*.

Consider again the `product` and `document` life cycles above. A likely scenario of events is as follows:

- The end user wants to commit that a specification document is finished, and will not be changed further. In A3E, this is signified by the actor `end-user` submitting the event `commit` to the artefact `specification-doc`.
- The event `commit` causes a transition from the state `contents-assigned` to the state `committed` to fire for `specification-doc`. Now, `specification-doc` is related to a `product` artefact through the relation `specification-document`. Therefore, the event is propagated to `product`.
- However, from Figure 3 we see that the life cycle model is expecting an event called `commit-spec`. Therefore, an event mapping operation that maps the incoming `commit` event to `commit-spec` is required.

Each artefact, activity, and actor includes such an event mapper that handles all incoming events, filters out uninteresting events, and maps the resulting events to events recognisable by the recipient. Such a mapping operation has proved to be useful in many ways. It allows modularity in ontology definitions in that life-cycle models can be developed quite independently of each other, as long as event mappings are possible. Moreover, the filtering operation makes it possible to adopt the most straightforward model of all for event propagation: propagate events across all relations.

5.3 Co-ordination Framework

The objective of the co-ordination framework is to provide facilities by which several independent agents, each having its own a process/product model based on the A3E ontology, can co-ordinate their shared activities. To match with this requirement, the implemented co-ordination framework presently consists of three related components:

- *matchmaking* for locating agents on the basis of their declared capabilities
- *dialogue control* for creating and keeping track of dialogues amongst agents on the basis of replicated model entities
- *event propagation* of replicated entities.

All three tasks are implemented via passing appropriate KQML messages between the agents. A message dispatcher transforms the incoming messages to events that are passed to the actor entity representing the agent itself (matchmaking or dialogue control messages) or to a local copy of a replicated entity (event propagation).

The matchmaking service is based on a single matchmaking broker assumed to be globally known to all agents. In the typical scenario, agents willing to provide

services to other agents *advertise* their capability to the broker. This allows the broker to respond to subsequent *recommend* messages of other agents willing to use the services. This is readily implemented using the standard KQML facilities.

The dialogue control service is based on replication. In the simplest scenario, an agent A may *open* a dialogue with another agent B on an A3E ontology entity E residing with A. As a result of this operation, the dialogue control service creates a replicate E′ of E with B, and an identity relation between E and E′.

Initially, E and E′ are exact (shallow) copies of each other. However, during subsequent manipulation of the model, they may evolve independently. For instance, E might model an activity that A wants to subcontract to B. In this case, B's model of the activity, E′, may evolve to become quite different from E by decomposition, further subcontracting, or like. In particular, the life-cycle states of E and E′ can be different.

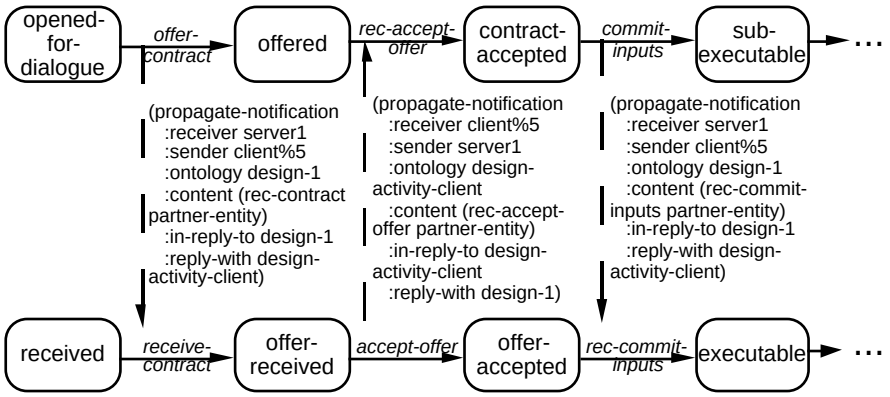

Figure 4 A sample distributed dialogue

To track such evolution, the identity relation between E and E′ is used for event propagation — just like any other relation. To avoid passing unnecessary messages across network, a proxy mechanism is included that distributes event filtering to the sender's site. Figure 4 gives a simple example of the life cycle evolution of two replicate entities. Observe that no separate protocol definition for the notification messages beyond the life cycle models of the entities is needed: the messages are created automatically as a side effect of transitions by the notification engine.

5.4 Implementation

A3E ontologies must be open to extensions to adapt them to various specific products and processes. Therefore, new kinds of artefact, activity, actor, and event entities must be able to be defined and used in a model. This requires a data-driven implementation where the entities and their behaviours can be flexibly tailored.

To achieve this, the present prototype uses a hybrid architecture consisting of a small core system and an extension system. The core system, written in C++, provides knowledge representation based on the frame ontology and a KQML interface for agent communication. The extension system supports introducing new types of A3E model entities. It is implemented by introducing a C-based interpreter (Laumann and Bormann 1992) for the Scheme language to the core. Using frame notation in Scheme, new model entities and their behaviours can be described and loaded into the modeller, where the described data are translated to frame instances of the C++ core.

6 FUTURE WORK

Our research is still at an early stage of completeness, and many issues and lines of work still remain to be studied.

At the present, the process ontologies covered by the A3E ontology are still very simple. The most important line of work for the near future must be to study real-world processes and capture their essential characteristics in the ontology, refining it as needed. We expect that in addition to identity relationship, other relationship types between replicated entities will be found to be useful.

The actual mechanisms of maintaining replicated artefact models will require further attention. Most likely we will include some type of checkin/checkout scheme to maintain the consistency of several copies of "same" data. We do not think that a locking mechanism can work in realistic cases.

The assumption that agents can remain permanently on-line is not a realistic one. To deal with errors and network latencies, a caching mechanism is needed.

In another line of further work, we will study the integration of the life-cycle model with some design model supporting non-monotonous evolution. Particularly in this context, the formal properties of ontologies will become of interest, such as their completeness, consistency, and correctness.

Last but not least, in future work we also plan to instrument real systems with A3E wrapping, such as CAD and PDM systems or document preparation software tools. In the near future, we also plan to replace the present Tcl/Tk user interface with a Java-based interface embeddable in a WWW browser.

7 CONCLUSIONS

We have discussed the concept of engineering process ontologies, and discussed the first steps of our work indented to develop a computational framework for virtual enterprise computing on the basis of Internet agents, shared ontologies, and distributed artefact-activity models.

Will these methods become useful in the practical sense? If yes, when and how? We believe that the emergence of Internet computing is presently changing the

basic platform of enterprise IT systems in a direction that favours the technologies and ideas we are suggesting in our work. Thus we deem that our approach is at least feasible.

We also believe that our approach is compatible with the economic imperatives that are at work in global industries. Integrating markets, global competition, concerns of quality, cost, and environmental performance are all forcing companies towards increased specialisation and co-operation. This creates a significant industrial market for tools that can support virtual enterprise computing.

In our opinion, the likely path towards identifying shared engineering ontologies will be through industry segment specific domain ontologies. At the present, several industrial sectors are already at work in developing specific ontologies covering electronic markets, subcontracting logistics, or configuration management. Standards are likely to follow, ultimately leading to generally available, shared engineering ontologies for virtual enterprise computing.

8 REFERENCES

Bernus, P. and Nemes, L., eds. (1996) Modelling and methodologies for enterprise integration. Chapman & Hall, London.

Bernus, P., Nemes, L., and T. J. Williams, eds. (1996) Architectures for enterprise integration. Chapman & Hall, London.

Chaudhri, V., Farquhar, A., Fikes, R., Karp, P., and Rice, J. (1998) Open knowledge base connectivity 2.0.2. Available through `http://WWW-KSL-SVC.stanford.edu:5915/doc/release/okbc/okbc-spec/okbc-spec.pdf`.

Finin, T., Fritzson, R., McKay, D., and McEntire, R. (1994) Using KQML as an agent communication language. Proc. of the Third International Conference on Information and Knowledge Management (CIKM'94), ACM Press.

Fox, M. and Gruninger, M., (1994) Ontologies for enterprise integration. Proc. 2nd Conference on Cooperative Information Systems, Toronto, Ontario.

Gruber, T. (1993) A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199-220.

Lahti, A., Mäntylä, M., and Ranta, M. (1997) Capturing and deploying design decisions. In M. Pratt, R.D. Sriram and M.J. Wozny (editors), Proc. IFIP WG 5.2 Geometric Modelling Workshop, Airlie, Virginia. IFIP Proceedings, Chapman & Hall, London.

Mäntylä, M., Finger, S., and Tomiyama, T., eds. (1997) Knowledge intensive CAD, Volume 2. Proceedings of the Second IFIP WG 5.2 Workshop on Knowledge-Intensive CAD, IFIP Proceedings, Chapman & Hall, London.

Ranta, M., Mäntylä, M., Umeda, Y., and Tomiyama, T. (1996) Integration of functional and feature-based product modelling — The IMS/GNOSIS experience. *Computer-Aided Design* 28(5):371–381.

Shah, J. and Mäntylä, M. (1996) Parametric and feature-based CAD/CAM, John Wiley and Sons, New York.

Tomiyama, T., Mäntylä, M., and Finger, S., eds. (1996) Knowledge intensive CAD, Volume 1. Proceedings of the First IFIP WG 5.2 Workshop on Knowledge-Intensive CAD, IFIP Proceedings, Chapman & Hall, London.

Uschold, M., King, M., Moralee, S., and Zorgios, Y. (1998) The enterprise ontology. *Knowledge Engineering Review*, Vol. 13.

X3T2 Ad Hoc Group on KIF (1995) Knowledge interchange format specification. Working Draft for an American National Standard, available through `http://logic.stanford.edu/kif/specification.html`.

9 BIOGRAPHIES

Martti Mäntylä received his Dr.Eng. in 1983 at the Helsinki University of Technology. In 1983–1984 he was a visiting scholar with the Computer Systems Laboratory at the Stanford University. In 1989 he was a World Trade Visiting Scientist at the IBM Thomas J. Watson Research Center in Yorktown Heights, NY. In 1996–1997 he was a visiting scientist at the Fraunhofer Institute for Computer Graphics in Darmstadt, Germany. Currently he is Professor of Information Technology and heads the Laboratory of Information Processing Science at the Helsinki University of Technology, Finland, where he is also the Director of the Helsinki Graduate School of Computer Science and Engineering.

Dr. Mäntylä's research interests include the full range of computer applications in engineering, such as CAD, CAM, CAE, and CIM, and their computational methods such as enterprise integration, product modelling, computer graphics, user interfaces, and data base management. He has published two books, three edited volumes, and some 60 refereed papers and articles in these fields. He is a member of the editorial board of *Computer-Aided Design* and a member of the ACM, the IEEE Computer Society, and the Eurographics Association, where he is also a Member of the Executive Committee. He is also a member of IFIP working groups 5.2, 5.3, and 5.10.

Mervi Ranta is a research scientist at the Laboratory of Information Processing Science in the Helsinki University of Technology. She is currently working in the Product Modelling and Realisation Group of the TAI Research Centre on an product modelling approach that promotes the reuse of product knowledge through its life-cycle. She received her MS in Computer Science from Helsinki University of Technology in 1987. During 1988-1992 she was a visiting researcher at the Kimura Laboratory in the Department of Precision Machinery Engineering of the University of Tokyo. Her research interests include product modelling, product data and knowledge management, design co-ordination, virtual engineering, life-cycle and reuse of product models, feature based modelling, and geometric modelling.