In the rest of this paper, we first discuss in Section 2 the industrial requirements of virtual engineering on the basis of the business processes that underlie and delineate engineering activities. This gives us a basis to analyse, in Section 3, the basic requirements that computational tools for virtual engineering should satisfy.

Section 4 follows this by describing a challenging application scenario of the concepts discussed earlier: coordinating a large distributed engineering process such as the design of an industrial plant. Finally, Section 5 gives an overview of the present state of our ongoing work, with comments on future issues and directions for research.

2. Motivation

The concept of virtual engineering is introduced to respond to a number of development trends and problems faced by industrial companies. To understand these, it is useful to first discuss the operation of industrial companies on the basis of their basic business processes.

For nearly all companies, the *core process* is the sequence of activities directly needed to make and sell products to customers. Typical activities include offering products on the marketplace; selling them to the customer; engineering the ordered product to customer specifications; planning the production; buying the required materials, semi-finished parts, modules, and services; manufacturing the products; shipping them; and installing them at customer's site. Hence, the core process is end-to-end: it starts from the customer and ends with the customer. In the following, we use the term *customer order satisfaction* to denote this process. See Figure 1 for illustration.

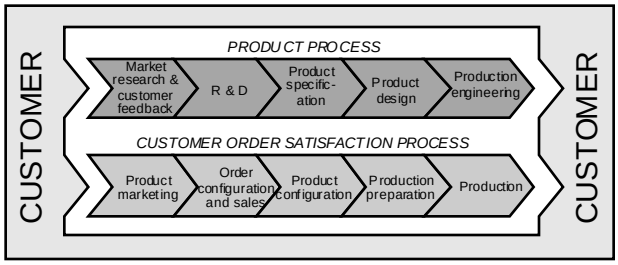

Figure 1 Engineering processes of an industrial company

Another main process of interest for virtual engineering is the *product process*. Its purpose is to develop products that can successfully be offered on the market by means of the customer order satisfaction process. It is ideally also end-to-end, starting from perceived market opportunities and customer needs, and ending with learning about market response.

The division of labour between these two processes varies according to the level of customisation needed to address the requirements of an individual customer. Mass-marketed products require no customisation from the manufacturing company (but may need customisation from the sales or service network). In the other extreme case lie complex one-of-a-kind products such as industrial plants or public infrastructure items where extensive design and engineering are required for every order. Most businesses lie somewhere between these extremes.

The customer order satisfaction process should be as effective, simple, and dependable as possible:

- The information should be linear and one-directional. Customer requirements are gathered completely and accurately at the start of the process, leaving no need to go back to the customer to clarify the original order.
- Reuse of existing product information such as solution principles, product structures, modules, interfaces, subassemblies, parts, materials, or process plans should be maximised.
- Effective and reusable business for establishing customer requirements, translating these into a product specification, configuring a product that meets the specification, and turning the product description to manufacturing should be defined and used. These processes should be repeatable and measurable.

The above goals define implicitly requirements also for the product process:

- The outcome of a product process is not a single product, but rather a customisable product platform consisting of a range of variant modules and parts which can be combined in multiple ways to create product instances satisfying the specific requirements of a customer.
- Methods and tools for gathering user requirements and translating these to valid and effective customised product configurations should be developed during the design process ("design for customisability") and made available to the customer order satisfaction process.

The challenge to product developers is to address all these issues (and other product life-cycle issues) already during product development by applying principles of simultaneous engineering. To meet this challenge once and for all during the initial design of a new product may be too much to ask; therefore, product development is seen as a process that continues through the entire life of a product platform. The continuous product process should follow some clear principles through which new generations and versions of the platform, its modules, and related information are specified, generated, and released for use by the customer order satisfaction process.

A common mistake for companies is to allow the distinction between product process and customer order satisfaction process to blur. To win an order, entirely new functionality is promised by sales to the customer. To realise these, "private" new versions of existing platforms and modules are created, without paying attention to their reusability. As a result, the number of product versions and configurations proliferate, adding to the cost of production, logistics, installation, and service, while designers are too busy with hacking the product to develop the base platform.

The basic motivation of virtual engineering from industrial viewpoint is that customer order satisfaction and product processes increasingly often take place not within a single company (as implied by Figure 1), but cross company borders with the result that the enterprise executing them is *virtual* (Figure 2).

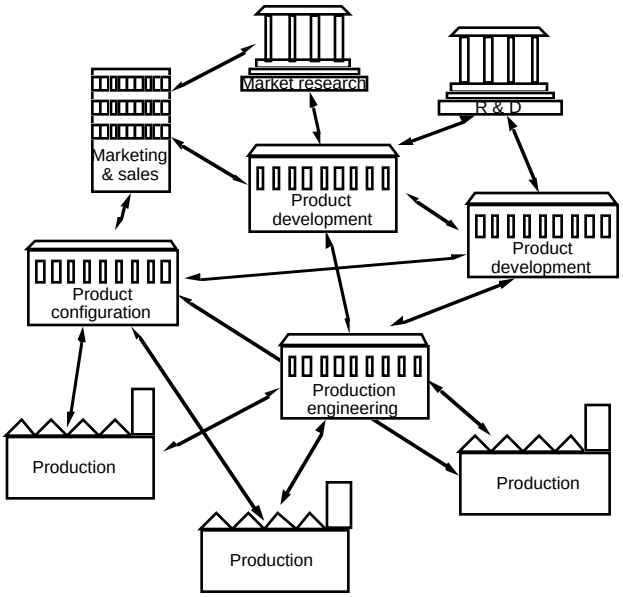

Figure 2 Engineering processes in a virtual enterprise

Among the prime reasons for this development, the following bear mention:

- Increasing complexity of products requires the application of increasingly specialised technologies in products, necessitating the use of external design and manufacturing resources.
- To be competitive and effective ("agile"), companies concentrate on their core competencies, resulting in increased use of extenrnally bought designs, manufacturing, and services.
- Partnership thinking and simultaneous engineering principles lead to close coupling of the design, planning, and manufacturing activities of companies and their main suppliers and service providers.
- Pressures for rapid order turnaround and application of just-in-time principles have resulted in the elimination of material and semi-finished product buffers, coupling the supply chain directly in the order satisfaction process.

In addition, also the internal structure of companies is increasingly "virtual" through application of principles such as profit centres and activity based costing. Instead of the hierarchically managed, military-like organisations of the past, a modern company is better characterised as a network of teams, each managed and working almost like an independent small enterprise.

3. Infrastructure for VE

The industrial objective of virtual engineering is to achieve the desirable qualities of engineering processes discussed in Section 2 also when they are executed by a virtual organisation, instead of a single company. A superior level of performance is actually expected: otherwise, the virtual organisation would have no economic justification. A fundamental hypothesis of our work is that to achieve fully this objective, a computational infrastructure specifically aimed at supporting virtual engineering is needed.

On the basis of the above discussion, the following qualitative requirements for the computational infrastructure can be stated:

- *Life-cycle support*: The infrastructure must provide explicit support to life-cycle engineering (Alting and Legarth 1995) covering the entire range of product and customer order satisfaction processes. In consequence, it must provide facilities for artefact (product) and activity (engineering process) modelling covering all stages of Figure 1 and beyond.
- *Co-operation and co-ordination*: The infrastructure must support engineering processes crossing company boundaries. This includes supporting information sharing between parallel engineering activities, and information flow between sequential processes. Support of co-operation independently of time or location differences is essential. In addition to pre-planned co-operation, also *opportunistic* co-operation emerging dynamically to address a unique customer requirement should be supported.
- *Distributed*: The infrastructure must be capable of dealing with truly distributed problems, where no centralised or shared data are available. Instead, co-ordination and communication must be based on symmetric distributed processing on the basis of potentially inhomogeneous systems.
- *Adaptable*: To maximise its industrial usefulness, the infrastructure must be adaptable to various business conditions. For instance, individual instances of the producer-supplier scenario vary considerably in the degree of existing shared knowledge from long-term strategic alliances and partnerships to purchasing of commodity components from a competitive open market.
- *Learning*: During the operation of a virtual organisation, its partners will learn about effective solutions to engineering problems and effective ways of sharing work and knowledge. These product and process innovations should be reusable in later co-operation.

To satisfy the above requirements, large-scale application of product models, engineering process models, and product realisation process models is needed. For reaching the specific objectives of virtual engineering, it

is important that product and various process models are closely integrated.

A central aspect of virtual engineering is that we cannot assume that such models of co-operating partners are directly shared (at least initially), nor can we assume that some central shared repository of models integrates the partners. Instead, shared product models, engineering process models, and product realisation process models evolve during the progress of the co-operation.

Such learning and information sharing must be based on joint understanding of common concepts and terminology — shared ontology. To certain extent, "standard" ontologies common in engineering — such as those in (Farquhar et al. 1992) — may be used to the extent applicable. More detailed shared ontologies should emerge during the co-operation as the result of a collaborative learning process (Gaines et al. 1996; n-dim 1995).

4. Application Scenario

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 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 engineering and its 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 distributed model is required.

We hypothetise 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 communication between agents by message conversion and mapping.

This suggests a methodology that can be described as follows:

- During the planning stage of the project, an initial 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.
- During the execution of the project, the project model is maintained with the help of the agents by registering locally created more detailed model entities to the distributed repository maintained by the agents and the broker.
- The evolution of activities and artefacts is tracked by means of an appropriate life-cycle model. Life-cycle transitions are registered to the broker and other interested agents and used for 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) with engineering-oriented models of the artefacts (product models, documentation, etc.) in a unified whole that can be used as a basis of simulation and simulation-based scheduling.

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 (empty or rough) models of the artefacts created and consumed by the various activities (Figure 3).

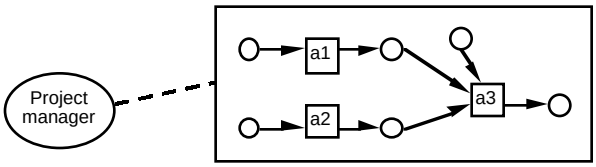

Figure 3 Initial activity model.

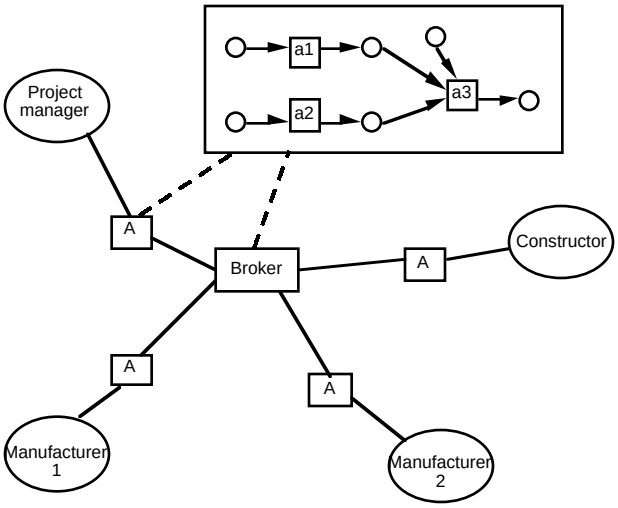

Figure 4 The initial activity model is published.

Project manager registers the created model to his agent (Figure 4). The agent further publishes the model to the broker. The broker is aware of the other agents representing the partners relevant to the project.

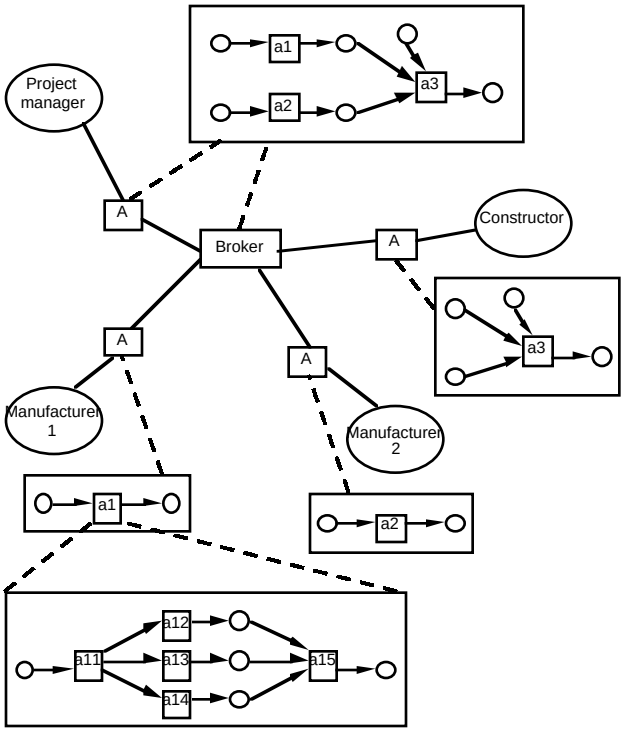

Figure 5 Activities are allocated to partners.

A negotiation follows. As a result, activities of the rough model are allocated to partners and a contract network maintained collectively by the agents is created (Figure 5). Each agent maintains a model of the relevant part(s) of the project. During negotiation, partners may need to decompose the activities allocated to them to create more detailed local models. The activities of the local models may be planned recursively; for instance, in Figure 5 manufacturer 1 decomposes activity a1 in a lower-level activity network.

During the execution of the project, the life-cycle state of activities and artefacts is maintained. Transitions are registered at the broker, who may decide to propagate the transitions to the interested agents on the basis of the life-cycle model and links to data maintained by the other brokers.

5. Current Work

At the present, research towards realising a computational system with the behaviour discussed in the previous section is in progress. In this work, a simple prototype modelling environment, titled A3 (for Artefacts-Activities-Actors) has been created.

To be able to exchange information at all, some minimal shared ontology between communicating partners must be assumed. In our work, the *A3 ontology* fulfils this role. As its name suggests, the A3 ontology is based on the following base entities:

Artefact: Artefacts are the things created as the final or intermediate product during an engineering process. Available artefacts may also be used as a basis of further redesign activities. Artefacts form a taxonomy of different types of things, such as specifications, test plans, user manuals, software modules, and CAD models (and their fine-grained constituents, such as features).

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 some input artefacts and resources and produces some output artefacts. Similarly to artefacts, activities also form a taxonomy.

Actors: Actors are the things that make activities happen; they are intended to model generically the people and organisations that really do the engineering work. They might also be used to model other useful things to have such as money, physical space, or machine. Unlike artefacts, which are "short-lived" things, actors are thought of as fairly rigid things that evolve only relatively slowly (as compared to the typical duration of activities). As before, they form a taxonomy.

In addition to the basic entities (and their taxonomic siblings), the base ontology has *relations*, modelling various sorts of dependencies between the entities. Like the other entities, relations form taxonomies.

A specific objective of the base ontology is to grasp the time evolution of engineering processes. Therefore, all entities of the model 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 a model entity. Different entity types have different life-cycle models.

According to the virtual construction scenario, the engineering work is performed in many parallel concurrent threads with complex inter-dependencies. A central aspect of the model must be the proper recording of these dependencies and their use to propagate changes and maintain the overall consistency of the distributed process.

To deal with this requirement, we have followed the approach chosen by the SHADE and SHARE projects, and implemented a separate ontology for change control and management following the principles of the Redux' server (Petrie 1993). This ontology characterises design in terms of the goals, decisions, assignments to design variables, and the rationale of the decisions. If two decisions (i.e., the assignments performed by the decisions) are found to contradict, the model can be used to propagate appropriate state changes to affected model.

To make use of Redux', it must be linked to the base A3 ontology. In our work, this is implemented by associating KQML (Finin et al. 1992) messages in the life-cycle models of the base A3 model entities following an approach similar to (Bradshaw 1996).

Of course, A3 ontologies must be open to extensions to adapt them to various specific products and processes. Therefore, new kinds of artefact, activity, and actor entities must be able to be defined and used in a model. This clearly requires a data-driven implementation approach where the entities and their behaviours can be flexibly and completely tailored. To achieve this, the present prototype uses a hybrid architecture consisting of a small core system and an extension system.

The core system provides a knowledge representation method based on the frame ontology and a KQML interface for agent communication. The core is written in C++. The extension system supports introducing new types of A3 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 behaviour can be described and loaded into the modeler, where the described data are translated to frame instances of the C++ core.

6. Conclusions

Virtual Engineering is a necessary next step needed to realise the full potential of virtual organisations. Of course, not all companies will need to become virtual. However, the development trends discussed in Section 1 are strong enough in many lines of business to suggest that those companies who learn to work effectively in virtual organisations are more likely to survive than those who do not. Moreover, although the work discussed was mainly motivated by problems arising in virtual organisations, also conventional enterprises can benefit from Virtual Engineering.

A good time to commence research and development intended to support Virtual Engineering is now. Through introduction of tools such as product models, product data management systems and workflow management systems, most advanced companies have reached the level where the integration of all these (and other) view of engineering work is becoming within reach.

References

1. The n-dim Group, N-dim – *An Environment for Realizing Computer Supported Collaboration in Design Work*, Technical Report EDRC 05-93-95, Engineering Design Research Center, Carnegie-Mellon University.
2. L. Alting and J. B. Legarth, Life cycle engineering and design, CIRP keynote paper, August 1995, *CIRP Annals*, vol. 1995.
3. J. M. Bradshaw, KAoS: An Open Agent Architecture Supporting Reuse, Interoperabiliby, and Extensibility. The Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, on November 9-14, 1996, http://ksi.cpsc.ucalgary.ca/KAW/KAW96/bradshaw/ KAW.html.
4. A. Farquhar, R. Fikes, W. Pratt, and J. Rice, *Collaborative Ontology Construction for Information Integration*, Knowledge Systems Laboratory Department of Computer Science, KSL-95-63, August 1995. Available as http://www-ksl.stanford.edu/KSL_Abstracts/KSL-95-63.html.
5. T. Finin, D. McKay, and R. Fritzson, *An Overview of KQML: A Knowledge Query and Manipulation Language*, Technical Report, Computer Science Department, University of Maryland, 1992. For on-line information on KQML, see http://www.cs.umbc.edu/kqml/.
6. B. R. Gaines, D. H. Norrie, A. Z. Lapsley and M. L. G. Shaw, Knowledge Management for Distributed Enterprises, *Proc. Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta, Canada, November 9-14, 1996, http://ksi.cpsc.ucalgary.ca/KAW/KAW96/gaines/KMDE.html.
7. O. Laumann and C. Bormann, Elk — An Extension Language Kit, *USENIX Computing Systems*, vol. 7, no. 4, 1994. For information on Elk, see also http://www-rn.informatik.uni-bremen.de/software/elk/.
8. C. Petrie, The Redux' Server, *Proc. International Conference on Intelligent and Cooperative Information Systems (ICICIS)*, Rotterdam, May 1993, file://cdr.stanford.edu/pub/CDR/Publications/Reports/design-nav.ps.