The Future IT Systems for Virtual Enterprises: Product-Oriented Agent Providers?

Nick B. Szirbik  
Department I&T, Eindhoven University of Technology,  
Den Dolech 2, 5600 MB Eindhoven,  
The Netherlands

Harinder S. Jagdev  
Computation Department, UMIST  
Manchester  
United Kingdom

Abstract - This paper proposes a new business architecture for agent-based IT integration between enterprises participating in Virtual Enterprises. The architecture is presented in terms of design decisions. These decisions are grouped at two levels: business related decisions, and purely software architecture related decisions. However, between these two levels there is a strong connection. En empirical argumentation for the business related decisions is constructed via a three example case-based study. Because the agent paradigm and the virtual enterprise paradigm are not yet clearly defined and overused in many domains, we focused first on views and definitions about these two. The conclusion of the paper is that an IT integration in a virtual enterprise should have a central approach, starting from the community marketplace concept, which is enriched with the concept of the Virtual Product and upgraded to the rank of Agent Provider concept. Finally, this architecture settles clearly the problem of agent ownership, by giving this to the Agent Provider.

I. INTRODUCTION

Modern IT systems, even the ones implemented by small enterprises can be complex. The principal reason for this complexity is due to the fact that individual components of the IT systems do not operate in isolation any more. Thus it is almost a prerequisite of modern IT systems that they seamlessly integrate among the various functions of an enterprise. It is this core philosophy that gave birth to the Enterprise Resource planning (ERP) systems. Therefore, most modern IT systems, even within small enterprises, are organized in the form of intranets. Hence, with the wide availability of the Internet, it is a small evolutionary step to expect intranets to be openly connected to the Internet. However, the inter-intranet connection is a very serious issue, and we present in this paper an agent oriented conceptual solution, in the form of a business architecture.

The paper is structured as follows: section 2 is focusing the domain of application and clarifies to what kind of Virtual Enterprise we are referring. Section 3 presents what kind of agent system model we are using, section 4 is presenting the concept of virtual product, as an extension of the classical Bill-of-Material concept, section 5 is presenting the potential and currently existing starting points for an agent-based VE IT system provider, section 6 is presenting our proposed architecture in the form of design decisions and section 7 concludes the paper.

II. THE VIRTUAL ENTERPRISE

Enterprises frequently have to choose between making or outsourcing (i.e. buying) components for their products. These decisions are mainly costly driven. Due to vertical integration in large manufacturing enterprises the organization is burdened with many non-comparable activities that had to be coordinated and managed. A current trend is that manufacturing enterprises, large, medium or small, are going back to their core-competencies (see [7]). Focusing on these resulted in outsourcing of all activities that were not considered to be of strategic importance to the organization’s market position [12]. The specialization allows them to design and manufacture better components more efficiently. In many industries it was observed [5] that the value added by the final assembly plant is marginal in the cost structure of the final product. A study [15] reveals that on average, 70% of the value of the final product is added by other than manufacturer of the final product.

Because the manufacturing of complex products implies a strong link between the enterprises that participate, the inter-organizational cooperation can thus be of strategic importance for those who wish to offer state-of-the-art products to their customers. In order to achieve a high level of cooperation, the enterprises are enacting more stronger bilateral relationships, technically enabled by modern ITC infrastructures. A non-empty set of bilateral relationships, enacted with the goal to enhance the cooperation for a specific product (i.e. engineering, manufacturing, servicing of this product) will form an enterprise network. An interesting feature of these networks is that these are not hierarchical, but have the structure of a flat organization, where no enterprise is controlling all relevant relationships. However, there is no complete “equality” in the terms of engagement between the enterprises. These terms are determined by the market conditions of the product or service offered by each enterprise, by reputation and knowledge, technology within the enterprise, type and uniqueness of the products, brand names and patents.

Formation of enterprise networks is principally dictated by the rule of “return on investment”. The accepted view in economics is that enterprises cooperate for the sole purpose of making more profit. However, participating in a network (not necessarily with all or most of the enterprise’s
resources) can have more deep consequences than mere profit. Inter-organizational cooperation may directly influence the continuity of the enterprise. One of the main goals of any enterprise is to remain in business, and their dependence on the other’s competencies and willingness to cooperate with them gives the enterprises less room for manoeuvre. Consequently, enterprises are putting now more effort in maintaining and setting the relationship with their suppliers and customers.

Our research is focused on a special case of inter-organizational cooperation, namely the Virtual Enterprise. This concept is unfortunately viewed in many different ways. We stick with the definition that “a VE is set up with the objective of providing one particular type of product or service” [8], a definition which is centered on “why?” and not “what?”. A similar approach is presented in [4], where “virtual organizations” are built around “a new kind of product, delivering instant customer gratification in a cost-effective way”. The authors believe that the future manufacturing company will not be an isolated facility of production, but rather a node in a complex network of suppliers, customers, engineering, and other service functions. The fast adaptation of the product to the customer needs requires the virtual corporation to maintain an integrated information infrastructure for customer data, product and production data, and design methods. In a simpler context, the VE is a highly information integrated supply-chain. Jagdev and Thoben propose (see [9]) a typology where the level of integration is deciding in which group a virtual organisation is located.

On a scale, there are two extremes, the market-place (where transactions are punctual and are not necessarily repeated) and the fully integrated company (where a vertical integration is insuring the high degree of internal interaction consistency). Between these two, we have the supply chain (the classical case), the extended enterprise (where enterprise coordinate their manufacturing schedules), and virtual enterprises (where enterprises share resources, engineer the product in a collaborative manner, and highly integrate their production and logistic planning), as in figure 1.

There is a temporal aspect in this definition. Supply chains are long term networks, having a relatively simple final product, which is in many cases manufactured-to-stock (MTS) using forecasting of customer demand techniques. Extended enterprises are enacted to manufacture a certain product or product line for a medium time window (a specific type of a car, for example). The manufacturing situation is assembly-to-order and it is customer driven. A VE is better suited for one-of-a-kind (OKP) products, in an engineer-to-order situation (ETO). The complexity of products is high in a EE and very high in a Virtual Enterprise (see figure 2). A dimension which was left out from this picture, is the legal one. Inter-enterprise cooperation can be based on in three different types of agreement: non-contractual agreement (as in cartels), contractual agreements (as in alliances) or joint ventures (as consortium, where joint ownership can exist). Our view is that it is better to have for the VE the first type: non-contractual agreements. This has some certain advantages: simplicity, facilitates the enactment of the VE on trust and reputation of the partners, avoids the settlements of occurring problems in court (that is costly and relation-damaging). We strongly believe that the way to use this type of legal (non) binding is to have a very highly integrated IT infrastructure, which allows the partners to have good Situational Awareness (the problems are detected and tackled in time) and good e-business support for negotiation and problem resolution.

The conclusion of this section is the following: our research is directed to find novel architectures and functionalities of IT systems that can support a VE, assuming that there is a high degree of trust and willingness to cooperate and the enterprises involved are not bound by a strong legal framework

III. AGENTS

To design an architecture for an IT system is basically a problem of software engineering. A new and rich metaphor in software engineering is the Agent-Oriented Development (AOD). Still in infancy, the methods proposed are still not offering an unitary picture, some being just a slight differentiation from the Object-Oriented Development methods (which are today very well established and supported by specific language and tools), others being very vague conceptual modelling techniques, which can be useful only in the initial phases of software development. Another methods are proposing only to tackle the modelling part of the process, leaving implementation details aside. These are called Agent-Based Modelling (ABM). The idea of using AOD and ABM for Virtual Enterprise software is not new. Our original idea is that for such an undertaking, one needs an adapted method, both for ABM and AOD. We found that the current work for this adaptation is not sufficient and
there are a lot of issues which still has to be solved. We discussed these issues in previous papers about this topic, as [2] and [17]. The main issues which are still in debate are:
- agent ownership
- how the coherence of the agent system is achieved
- which objects in the real world the software agents has to represent (and act on their behalf).

We strongly believe that AOD is not suited for all possible applications, and also, there is no generic method which covers all the applications that are well suited. Only distributed applications where the interaction between components mimic human interaction are well suited to AOD, and there is an important differentiation between various applications. In the case of typical VE interaction, only specific modeling, development and deployment techniques should be employed. We considered this approach in accordance with the assumption that the AOD method which is trying to “solve the world” (i.e. it is applicable in all contexts) is a mistaken path for research [10].

Modelling is not only useful as a prerequisite to develop software, but also it is a domain in itself. Enterprise reference models like CIMOSA and ARIS have shown the necessity of specialized languages for modelling business processes, enterprise data, information flows, enterprise operations and organizational structures along with their functionality. A powerful enterprise modelling language can improve adaptive activities like Business Process Reengineering, organizational change, upgrade of legacy systems and functions and especially the enactment of inter-organizational relations (people which can understand the same “modelling language” can interact more easily). In our approach we consider the following categories of agents:
- human agents (sometimes referred as actors in modelling)
- institutional agents (formed by groups of humans)
- artificial agents (software agents and mechanical robots, which are not included in this discussion, apart from the fact that their software can be viewed as a separate agent). There are other categories, like animals and other living creatures (ants are an interesting example, by showing a very rich society behavior, in contrast with their individual lack of complexity) but we do not discuss about them here. In our view, software agents are extensions of the human or institutional agents. There is no room for a software agent who is an entity on its own (it does not represent another agent or it is owned by another agent). The necessity to link any artificial agent to a human agent is to link the responsibility of the agent actions to the human (or institutional) responsibility. Any harm, malicious or due to faulty design, caused by an artificial agent makes its owner liable. An example are the computer viruses, which are mobile, auto-reproductive agents. The authors are considered criminals in any legal framework around the globe. Faulty software in agents can produce more damage than in normal software, because the agents are usually empowered to execute actions which are usually taken their human owners (who can delegate the agents to take action on their behalf, and they do not check the outcome of these actions). This imposes strong requirements on the software development method.

In classical supply chains, the agents are viewed as wrappers of the IT systems of each enterprise in the SC, as in the approach of Kjenstad [11]. The main emphasis is put on the communication language. If the SC adopts a unique language, the agents can ask and respond to various queries. The global functions implemented by the agents are the typical SC management functions. But these agents are mere extensions (language interfaces) of the existing legacy systems present in the SC. They are owned by the enterprise and they are representing the enterprise as a whole institutional agent. The level of granularity is very coarse, we have one agent for each enterprise and we have only one type of agent in the overall system. Such a modelling method is in our view very restrictive. Also, the implementation effort required at each enterprise is considerable, and each enterprise has to agentify its IT interface with the rest of the supply chain. Moreover, there is no proof that the system will work, because only the language is uniform over the SC but the processes activated by the messages are not and inconsistencies can appear, as demonstrated by the

Figure 2: The location of our interest within the manufacturing coordinates
research in the field of inter-organizational workflow (for an introductory reading in this domain, see [1]). One last observation is that the resulting agents via this method are very “fat” agents, they are extremely complex and heterogeneous pieces of software.

We propose to increase the granularity of the agent system, having more than one agent per enterprise. Also the model of the agent should be unified across the enterprise. A simple model is given in figure 3. In this model, each agent is executing the following loop:

```java
while (true) {
    state = env.senseEnvironment ( agent_own_id );
    action_list = infer_what_to_do (state, beliefs, rules);
    beliefs = local_effect ( state, beliefs, rules );
    env.executeAction ( action_list )
}
```

An agent can sense the environment (which can give different input to different agents, according with their own identification - in this sense, the environment can be considered as another agent, sometimes called an infohabitat). Each agent has a specific type, and this is given by the behavior he shows. The behavior is specified by rules (simple reactive production rules, or fuzzy rules, or chained deduction rules, etc.). According with the current state of the agent, given by the set of beliefs (local knowledge of the agent about its state and interpretation of trajectory of states - history - he considered worthy to record in the environment). We have not considered in this picture the passive objects, which are information objects manipulated by the agents. An ActionsEvents object could be for example a message sent by an agent and received (sensed) by another through their environment. We can have more environments, and these can be grouped together as agents (in a holonic way) by a higher level environment. An agent here can be also a human agent, because his interaction with the software system (via an interface with the agent environment) can be credibly modelled by the above presented interaction loop.

This way, environments can represent (and are owned by) institutional agents, because they group human and software agents.

Groups of these institutional agents can recursively form higher level institutional agents and so on.

IV. THE SHIFT FROM BILL-OFF-MATERIAL TO VIRTUAL PRODUCT

One of the main achievements in industry today is the ease of the use of the customizing techniques. When buying a low price car like the Peugeot 206, the customer (who has to wait four months on average for this very demanded car - the situation in March 2001) can select a lot of customized features. These are sent by the car-dealer to the manufacturer, and this one can introduce the customer data in his ERP system. In this way, we can consider that virtually the car started to “exist”, as a planned entity. Also, there exists already a link between the planned entity and the customer, who already started to pay for the car. The main advantages of this manufacturing situation are increased customer satisfaction and lower inventories (due to the shift upstream the supply chain of the Customer-Order-Decoupling Point, CODP). All ERP systems today can support this style of manufacturing, called Assemble-to-Order, by using the method of the parametrized (or “generic”) Bill-of-Material (BOM).

This style of manufacturing is not at all new. Actually, it is the first style of manufacturing which ever existed. One-of-a-Kind, Engineer-to-Order (ETO) was the case for the first complex products made by the human society, like ships and mining steam machinery. Some high priced automotive businesses (Rolls-Royce, Bugatti, etc.) have always been committed only to this style of manufacturing. For mass-production, this style was impossible in the times of paper based information systems due the exaggerated overhead of propagating the information in the supply chain. Such a scheme as one used by Peugeot, applied as an exercise of imagination to the Ford-T manufacturing style, will...
 imply a financially non-feasible army of clerks (very difficult to coordinate, also) and a huge amount of paper messaging between the dealers and the enterprise and between the lower level factory compartments as well.

Modern information technology was the enabler of this shift from mass manufacturing to “mass customizing”. Dealers can send the information to the enterprise using the Internet, the ERP system can plan efficiently how and when the customer ordered product will be assembled, intranets and extranets will facilitate the seamless work-orders and purchasing orders (in the case external suppliers are needed) release, and monitoring and control systems will identify and correct deviations from the initial plans. More than that, advances in e-commerce permit now direct customer to business interaction, the customer can order and configure its product without physically going to the dealer shop, and he also can negotiate the price vs. the delivery date. After the point of sale, the business can interact with the customer, by offering him upgrades of the product, or even new versions of it.

There are some comments here. The first is that customizing is limited by the level of module parametrization. All the complex products of today, with few exceptions, are highly modular. Each generic module has a number of parameters (like the number of cylinders for a car engine, or the external color of the car). Usually the product has a low number of parametrized modules, each with few parameters. Because the number of combinations is combinatorial, there are in any case many possible configurations the customer can choose. However, the number of combinations is drastically reduced by the design and functional interdependencies (a variant with the most powerful engine cannot have a low power gear-box for example). Actually, the customer can have only one of the versions which was designed from start to exist. There is no room for Engineer-to-Order in the information infrastructures used today in mass customizing. This is still only used in very specific ETO style, One-of-a-Kind manufacturing situation, like leisure yacht and cruise-ship building.

The second comment is that the configuration of the customized product is frozen after the customer decided for his particular set of parameters. The desire of the customer to change its parameters after the order has been made can be a result of changes occurring fast in the market (for example, a V6 turbo engine can prove too costly for a customer if the fuel price rose abruptly, and he may want a 4 cylinder variant). The fact is that the manufacturing company could easily accept this change if the change is made before the assembly process is on. This is especially the case when the demand is high, because the assembly of a car, which has been already customized and ordered, is taking place very close to the delivery date. The manufacturing of the modules is also taking place late, just-in-time before the assembly. In fact, such a change can imply only modifications in the planning data. Due to the powerful IT hardware infrastructure that exists today, such a change should be easy to perform. The problem is that the existing ERP software packages are not yet powerful enough to cope with such kind of change. This type of change is viewed still as an “annoyance”, by giving the customer the possibility to “disturb” the manufacturing plans of the enterprise. In our opinion, this is a mistaken view. The technology is mature enough to adopt a changeable order and this can be viewed as a source of income. Any post-ordering change should be charged according with the level of disturbance created, and of course, some changes, especially the late ones, will be not accepted. However, in the defense sector for example, last minute changes are frequent and considered very important.

Not only the customer can induce changes in the configuration after the order was issued. There is a possibility that the factory cannot make the product according to the parameters, due to unexpected factors (a module with a specific parameter value is not produced any more or there are some current problems that make impossible the manufacturing of a specific type of a module). This problem is exacerbated by the fact that a relatively complex product is manufactured and assembled in more than one plant (the case of multi-site enterprise) or enterprise (the case of a network of enterprises). Propagating a parameter change in an inter-enterprise environment is a more difficult task than propagating the change in a single organizational unit. In this case, the customer must be considered as being part of the enterprise network, because he has to accept the forced change and its propagating implications, and also he must be able to negotiate new price/delivery date coordinates for the changed product.

The most complex type of change is when one of the enterprises involved in the making of a product is applying dynamically a design change which has nothing to do with the generic BOM. Not the customized parameters of a module are changed, but the product design is altered in a way which is more profitable for the enterprise making it. Even if the parameters observable by the customer are the same, such a change can have implications in the following manufacturing/assembly steps, and probably on the general performance of the overall product. The classic approach to avoid the propagation of change in this situation, the interface between the modules is standardized, and each enterprise is seeing the rest of the modules as black boxes, with a visible interface and performance requirements. But no standardisation procedure can prevent that modifications of detailed design induce loss of reliability and expected performance. The development of version independent interfaces is also a method to “pour concrete on the feet” of future possible developments, which have the role to optimize the product, but cannot be enacted because of a rigid or sometimes an obsolete technology-based interface.

A desirable system to cope with this type of change is supposed to support any kind of change, in customized parameters as well as in more detailed design features. An important case is when the internal change can affect the customized parameters set by the user. In this case, both types of change can appear and spread together. Our proposal is to use a new concept, which enlarges the BOM concept. We called this concept the Virtual Product. The
concept is not new, it was introduced by Davidow and Malone, but here it has a slightly different semantics. In our view, the virtual product is related more to the capabilities of a VE than the products that are currently manufactured. The same enterprises participating in a VE can produce today One-of-a-Kind aeroplanes or space ships, next year a special type of train and the next year formula 1 cars. But the capabilities and competencies across the VE remain basically the same. These capabilities must be captured first by a virtual product (VP).

We want to differentiate in this paper between the virtual product represented by a mere collection of CAD files and the VP we are discussing here. Although the CAD data is an important part of any virtual product and it enables collaborative engineering in a multi-enterprise environment, this data represent only the basic level in a three layered framework (figure 4).

On the first layer we need a high level description of the technological capabilities that are necessary to manufacture (assemble) the product, together with a description of the human competencies which are needed. On the second level we need a description of the processes that are carried on by the resources. Only the third level is dedicated to a detailed description of the product, which can be customized, using the parameters, or changed, by intervening slightly in the design. There is a temporal differentiation between the levels of volatility of the information on these layers. The CAD data is changed more often, according with the market, and in “light” ETO cases, due to each customer demand. The processes however, remain the same for different products, and the conceptual information is the most stable. The problem facing the research in this area is that the information at the higher levels is very difficult to be formalized and consequently transformed into e-documents that enable automated and efficient processing.

In our approach, the VP should refer first to the two higher levels, containing only as few detail data as possible. The main feature of our VP concept is that it is distributed. The capabilities, competencies, processes and designs are scattered over a number of different enterprises creating sub-Virtual Products (sub-VP). An interesting feature here is that an enterprise can be very easily replaced by another enterprise with the same sub-VP on the conceptual and process level (designs can be easily copied and transferred to other parties). Moreover, enterprises with the same conceptual level of their sub-VP can adapt their processes and join the VE (though not immediately). The terminology of our VP concept defines three related concepts:

1. The VP-model: a generic model of a class of virtual products.
2. The VP: the concept already presented.
3. The VP instance: when a customer orders and customizes a product from the VE, a copy of the VP, with actual parameters is created in the IT support system of the VE. Each order has its instance, which can be kept by the VE after the point of sale.

This paper is presenting a preliminary research. In our current testbed framework (called PROVE [Szirbik 2001]) we considered until now only a simple BOM as the VP model. A possible scheme for a VP model should comprise:

- at the conceptual description level:
  • a high-level description of: why this product is realized and how it is realized
  • a functional description: what the product can do
  • a structural description (in very simple terms, the BOM and the modularized decomposition)
  • the customized parameters
  • the materials and fabrication process types
- at the process level:
  • the module interfaces and the interdependency description
  • the routing and the assembly sequence
  • the Analysis & Evaluation process description
  • the Change acceptance process description
- at the detail level:
  • only the changeable (geometrical, physical, etc.) features

Any propagating change of a parameter or a feature must obey the requirements of the FFF paradigm (form, fit, function must not influence the overall assembly requirements). In [13] we have presented how the change process can be described for simple BOMs and product change cases.

When a VP is instantiated by an order, in the created copy of the VP, the parameters are fixed, the mapping of the routing is filled with the enterprises’ identities (we assume that the enterprises in the VE can be selected from a bigger group) and the these enterprises can pick the local detail design they consider appropriate.

V. THE CURRENT SITUATION

To measure the level of integration, Jagdev and Thoben [9] propose a series of dimensions. For a simple demonstrator case study we will use here only 5 dimensions: legal, informational, organizational, financial, operational. Each dimension has three quantified values (low, medium and high). We investigate three cases: the extended enterprise (EE), the virtual enterprise (VE) and the community marketplace (MP), with instances from typical real-life cases.

An example of an extended enterprise is the Boeing civilian airliner building division. Boeing has streamlined its base of suppliers, integrating closely the first tier suppliers by enacting strong bilateral relations. The suppliers are...
financially dependent on Boeing orders, but long term contracts with Boeing insure their stability. The IT integration has been achieved by Boeing by extending its own ERP system to the suppliers. ERP systems are currently very expensive solutions for medium and small enterprises (the majority of Boeing’s suppliers are SMEs) and Boeing is leasing “seats” on their own system. This improves drastically the information sharing and operational alignment, because plans and schedules are kept consistent along the supply chain by the centralized ERP. The suppliers how eventually have their own ERP system, have this linked to the central Boeing division ERP system.

The Boeing company contains also the military division, the space transport division, the helicopter division, the satellite division and other smaller divisions, dominating all the related markets. In order to be able to compete, the European aerospace industry community formed EADS, a consortium based on joint ownership and unique stock market value. This alliance financially consolidated the composing enterprises, shared R&D resources, prevented harmful competition, optimized centralized purchasing and controlled better centralized sales and marketing. It has also the advantage that offers an unique image (and brandname) to the customers.

From the IT point of view, the integration is relatively low and covers mainly the financial and the collaborative R&D aspects. Moreover, the IT integration targets only the participants in the EADS consortium, which are big companies, having each one its proprietary IT system with a more or less integrated supply chain or extended enterprise of suppliers. But the overall reunion of suppliers is not integrated in any way. In our view, this is not the virtual enterprise we are looking for, as described in section 2. The strong legal binding of the consortium and the vertical (hierarchical) organizational structure brings a level of rigidity that is not desired for an agile VE (as defined by Goranson, in [6]), despite the advantages offered by the centralized financial consolidation. And it is clear that the driver for this type of integration is not the advance in IT, but the motivation is purely economical and political and the network is not a result of community interest driven coalescence of bilateral relations.

The third case is the community e-marketplace called Inventory Locator Service (ILS - www.ilsmart.com). This MP is dedicated for the general aviation community (including the US Department of Defense), and helps enterprises to search for hard-to-find aeroplane parts. It is more than a search engine, offering the sellers immediate access to parts inventories, repair capabilities, repair capabilities and overhaul services, even assisting the seller and the buyer to interact successfully. It has 7000 users in 78 countries and it is making visible a product database of 5 billion available parts. Each day, 27000 transactions are made on average. The company is 21 year old, first it has used a mainframe with terminals at 72 customers, in 1989 switched to modem connected PC’s, when the number of customers exceeded 500. In 1994, e-mail was used for inquiries and responses. With the advent of direct Internet technology and automated support for procurement processing in 1999, the turnover increased 210% in two years. The driver is that airlines and maintenance/overhaul enterprises tend to keep the spare parts inventories as low and diverse as possible. ILS transform virtually these local stocks of all the participating enterprises in an overall “single” stock, helping considerably the logistics of airline servicing.

The IT integration is relatively high, the selling participants (which are buyers also, in different transactional contexts) are publishing their capabilities and parts inventories. To automate the process, electronic catalogues of products are currently enhanced using ontological engineering, in order to achieve better semantic alignment (it happens often that same parts are named differently, of different parts have the same name in different organizations and countries). Also, the procurement process is supported by dedicated software application that can assist the buyer to decide the selection from multiple sources for a part (possible by parametrized manufacturing) or by indicating alternate parts and solutions. ILS is currently a small independent firm (110 employees), without presence on the stock market. The result is that a whole community is networked by a logistic service provider which is a neutral company (however, there are also other aerospace exchanges backed by big companies like General Electric or United Technologies, but these are also stand-alone enterprises). The organizational structure of the network is flat and there is no strong legal of financial binding, the selling customers paying a meager USD400 fee per month. Therefore ILS is achieving a more stable financial dynamic than in the case of pay-per-transaction situation.

We summarize in table 1 the results of this case study.

<table>
<thead>
<tr>
<th>type</th>
<th>legal</th>
<th>organiz.</th>
<th>financial</th>
<th>IT</th>
<th>operat.</th>
</tr>
</thead>
<tbody>
<tr>
<td>EE</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>high</td>
</tr>
<tr>
<td>VE</td>
<td>high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>MP</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>medium</td>
</tr>
</tbody>
</table>

The conclusion we have drawn is that the ideal VE is a combination of the VE consortium (like EADS) and the electronic marketplace. The IT integration of a VE must be higher, the financial interdependence, with the organizational and legal binding lower and the operational integration as high as possible (IT enabled).

VI. THE ENVISAGED ARCHITECTURE

The business process in a VE starts with an order by a customer for a composite product. The order has to be decomposed into suborders, which are dispatched to the participants in the VE, which may be regarded as subcontractors for a specific component. When the component again is a composite product, the participant in turn may
decompose its order into orders for sub-components and so on, until the component can be produced by a single enterprise. The decomposition will follow the VP structural scheme, and the manufacturing tasks are allocated to enterprises who have the necessary resources and accept the order. In order to be able to make the allocation, one enterprise in the network, has to have knowledge about the sub-VP of the other enterprises. Our approach is to consider that a special firm (as is the marketplace) has this information, at any sub-VP level (the structure of the VP is recursive). As a supplementary function, the firm has to perform the Customer Order Management (COM) for the customers of the networked VE.

To present our ideas about the proposed architecture to the reader, ideas implemented in our testbed system (the PROVE system), we consider that it is worthwhile to discuss the design decisions we made when developing this system. We can define two levels of decisions: business and architecture related decisions, respectively. The business decisions concern the formation and leadership of the parties involved in a VE. The VE needs to become self aware and take commonly agreed decisions about how co-operation and overall performance (especially in terms of customer satisfaction) can be improved. The architectural decisions are related to the ICT support development for a VE. They are, however, strongly based on the business decisions that determine the design variables of a software system. The architectural decisions are taken by the software and system architects in co-operation with the stakeholders (participants in the network). The business decisions are:

**Decision B1:** Adopt a mixed initiative interaction system based on human and software agents. This decision is the basic idea in ROVE. It entails the following consequences:
- Software agents are empowered to take supply chain control decisions.
- The human to human interaction, which underlies current supply chain control systems, is replaced by an agent mediated scheme in which some of the decisions are initiated by humans and some by the software agents. The first consequence implies that humans will trust the decisions taken by the agents. To ensure this, it is absolutely necessary that the human users understand how the agents work. For this purpose the agent behavior is defined by rules that are understandable by the human VE controllers. This conclusion supports a subsequent architectural decision, that is, to use rule-based reasoning systems in the agents. In our opinion, the main challenge of the research in this domain is to see how agents can develop more initiative and take some of the routine decision making burden in the VE control system from the human actors.

**Decision B2:** For IT agent-oriented support, the parties in a VE rely on a common software provider. This provider plays a role similar to the role played by the marketplace or a provider of complex software who provide a reliable and trustworthy product. In this case, the agents (dedicated to perform uniform tasks over a large area) are developed by a single provider, which we call the SACP (software agent common provider). Such a specialized firm can be provider for more than one VE. The SACP could be an independent party, which enjoys a high level of trust. Each SACP should be grouped around a certain class of Virtual Products. The SACP should have the following responsibilities:
- Develop the required agents and keep record of the parties who are using them (the enterprise in the network, who join via a pull or push procedure, depending also on the reputation of the SACP or the consultancy firms which are supporting the SACP).
- Help the parties to install related software and teach users to use it. (this software should be kept as simple and small as possible)
- Help the users with expertise to enact interfaces, in order to make local data available to the agents.
- Maintain and upgrade the system.

**Decision B3:** The security of the agent system is supported by a trusted third party by monitoring and detecting hostile activities. Because of the dynamic composition of the VE and its distributed nature, it is best to contract secure communication channels as a service, e.g. in the form of a Virtual Private Network, from a trusted, specialized provider. Also, the agents are owned by the SACP, and for any harmful action of the agents, the SACP should be responsible.

**Decision B4:** Centralization of common services. In the PROVE framework, a central site has, amongst other things, the following functions:
- Maintain a central repository of data about the VPs and the available resources in the VE, updated as often as possible.
- Calculate Global Schedules for orders.

The rationale behind this decision is that one consistent body of global information makes it simpler to generate optimal global schedules, and to avoid the resource conflicts, inconsistencies and inefficiencies implied by distributed alternatives. It furthermore relieves the participant enterprises from the hard task of bilaterally achieved global coordination. A Central Repository for Availability data, Global Scheduling capabilities and a Trader (for VP data management) represent the minimal centralized infrastructure, leaving unaltered the inherently distributed nature of the system.

**Decision B5:** Enact a unique Web portal for the VE for the interaction with the customers. This portal exposes the portfolio of the VE and permits customers to issue orders. EADS has for example such a portal, containing the complete portfolio of the consortium. This portal is intended for the final products and not to publish the VP data. The roles of SACP, of trusted third party, and of maintaining a central site and a Web portal may be attributed to a single company.

The architecture related decisions are:

**Decision A1:** Central Repository for Resource Availability Data. This decision can be viewed as a natural consequence of Business Decision B4. The rationale is that the overall functionality of the system is simplified. It offers the possibility to have global knowledge about the status of the “controlled” system. A central repository allows agents to
access necessary data (via the VP Trader) immediately. In addition, Global Scheduling can take place using local data, without the need to collect every time the necessary information. Also, if the resources in the VE are almost fully committed, it is very easy to refuse orders without keeping the customer waiting. Because it always takes a finite amount of time to reflect changes of the local resource availability in the repository, the latter may not always reflect the latest state of the VE. This is not a real issue, because resources must be committed locally before usage. When an agent finds the information kept in the central repository to be outdated or inconsistent with the local reality, (because the local scheduler has assigned the resource to a local production process) an exception is raised. The agent can invoke the Trader to find a new available resource of similar type and time-slot, or it can ask for a calculation of a new Global Schedule.

The alternative to centralization is to keep the availability data local (i.e. in the enterprises). In this case, the scenario of using the agents is quite different. When a customer issues an order, the portal deploys agents to all sites, where they search and select the best offers. It is still possible to make a global schedule, but only after the agents have collected the data. In architectures without a central repository, the main activity of the agent will be searching. For instance, when an agent needs a new resource it has to send agents throughout the entire VE to find one. Also in case of delay management, in order to obtain the value of the penalty to be paid to the customer, a Global Schedule has to be calculated, which implies collecting all data again. Also the optimization of the Global Schedules for different products is very difficult, because this requires the entire Availability Gantt chart to be constructed. Another consequence is that the agents need more functionality (for searching, navigating, co-ordination, etc.) in order to compensate the lack of a central Trader.

**Decision A2: Centralized Global Scheduling.** One of the reasons why Virtual Enterprising exists is better customer satisfaction, especially in terms of fast order execution. This is supported by a central scheduling capability, since this allows the use of the available resources in a way that minimizes lead-times. In a minimal distributed scenario, an agent can do a best-first search of the available resources and construct a schedule for the production of the order. Such a schedule will in general be far from optimal, because for the construction of an optimal schedule, global knowledge is needed.

The way the schedules are constructed depends on architecture decision A1. Without a central repository, the schedules will have to be calculated by the agents after they have collected all necessary information about resource availability. Schedules may now be produced concurrently at different sites for different orders, based on the same availability data. Independent construction of schedules can lead to the allocation of the same resources to different targets. It is quite difficult to see how these conflicts can be avoided or how conflict resolution can be done.

**Decision A3: Serial calculation of schedules.** Global Schedules can be calculated at the central site in parallel or serially. The choice depends on the scheduling load. The simplest variant is to calculate the schedules serially. Since the number of sites, of tasks per product and of available resources is expected to be low, it should be possible to produce the schedules within a sufficiently short time. To support parallel scheduling, a concurrency control mechanism must be implemented. Also this should not present any problems, because the schedule generation is taking place at a single site based on a single set of data.

**Decision A4: Use Polling for data collection.** The next question is how the data for the central repository should be collected. There are two basic approaches: polling and publisher/subscriber. In a polling approach, the central site has a list of all sites, and it asks them, via messages or by sending agents, whether any changes in the availability of the local resources have occurred within the production horizon of the VE. This approach has the disadvantage of complicating the functionality of the central site. It also increases the communication volume between the local sites and the central one, and thus the possibility of congestion. A new updating service is needed. In a publisher/subscriber approach, the local sites have to send the information about relevant changes to the central site. This approach complicates the functionality of the local service bridges. These service bridges are pieces of software which are developed by local programmers and it is better to keep them as simple as possible. The service bridges are basically only an interface between docked agents and local data repositories or humans. Furthermore, if all sites send the information in the same time, also in this case congestion can occur. In view of the maintainability of the system, and the ease of installation and use we have chosen the polling approach.

**Decision A5: Message Based Communication between agents.** It is logical to use message-based communication for co-ordination tasks. The agents, who participate, are docked at various sites, tracking the production of the parts they are responsible for. There is no need for them to move to another location just to communicate with another agent.

**Decision A6: Use a pull model for the deployment of the agents.** There are three possible models (presented in [3]) for agent deployment (agents can be created at the desired location, or can migrate, if the implementation provides mobile software) in a distributed environment:

- **Pull:** the agents are “sent” from buyers (assemblers) to suppliers (customer driven).
- **Push:** the agents are sent from suppliers to buyers with offers (supplier driven - not the case in a VE.
- **Broker:** both buyers and suppliers are sending agents to a virtual marketplace, where agents can participate in auctions and exchange information in various ways (does not allow operational integration). According to Dasgupta the broker model is best suited for e-commerce applications and not for VE support. Since a VE is a customer (buyer) driven environment, the best model seems to be the pull model (as it is used in the MagNET system). Also the monitoring task of the agents implies a pull model.
**Decision A7: Rule-Based Systems for the implementation of Agent Behavior.** This design decision decouples the agent behavior from its implementation. When the whole system is written in Java for example, and the behavior is coded directly in a procedural way (as Java classes), any modification to the behavior of the agents implies difficult Java code re-writing. The main advantage of rules is that they are much easier to read than code and have clear semantics even for non-programmers. Furthermore, we only have to modify the rule base of the agents to change the application we want to support. The disadvantage of this approach is that it is quite difficult to see if a specific rule base is correct or not. Our current research (see [14], [16]) is directed to find methods that can give a proof for the rule bodies interacting together.

**VII. CONCLUSIONS**

This paper has presented a possible architecture for the future software systems of the Virtual Enterprises of tomorrow. The novelty of the shown ideas resides in the agentification of the Virtual Product concept and the assumption that the agents are owned by a third trusted party (the SACP - Software Agent Common Provider).

The SACP has the role to identify and enact the network, maybe with the backing of some of the final systems integrators, which are already using virtual enterprising. Alternatively, the current marketplaces could extend their functions by adopting agentified architectures built around Virtual Products.

Our testbed system (PROVE) still lacks the complexity introduced by the Virtual Product concept, using only a simple BOM as information about the product that is manufactured in the Virtual Enterprise. A next logical step is to develop first the Virtual Product concept further, at a formal level, and by using novel methods, as ontological engineering and meta-modelling.

**VIII. REFERENCES**


