Teknillinen korkeakoulu Tietotekniikan osasto

Tietojenkäsittelyopin laboratorio B

Helsinki University of Technology Department of Computer Science and Engineering
Laboratory of Information Processing Science B

Espoo 2003 TKO-B 153/03

PRODUCT AGENTS FOR HANDLING INFORMATION ABOUT PHYSICAL OBJECTS

Kary Främling, Jan Holmström, Timo Ala-Risku, Mikko Kärkkäinen

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY

Distribution:
Helsinki University of Technology

Laboratory of Information Processing Science
URL: http://www.cs.hut.fi/english.html

This is an electronic version of the report. It can be found in Adobe Portable Document Format (PDF) at the following address:
http://www.cs.hut.fi/Publication/Reports/B153.pdf.

There is no separate paper version of this report.

ISBN: 951-22-6853-1

ISSN: 1239-6893

Espoo 2003

Abstract

Authors:	Kary Främling, Jan Holmström, Timo Ala-Risku, Mikko Kärkkäinen	
Title:	Product Agents for Handling Information about Physical Objects	
Publisher:	Helsinki University of Technology Laboratory of Information Processing Science	
Report code:	TKO-B 153/03	**Number of Pages**: 20
Date:	November 28, 2003	

The concept “Internet of Things” launched recently refers to the idea that product information would be easily available everywhere through a suitable product identifier and the Internet. RFID technology and new identification standards have been presented as the cornerstone of this system. In this article, we show how the “Internet of Things” can be implemented using existing identification standards and technology. Handling product assemblies is one example of this, for which this paper presents an agent-centric solution. An industrial track-and-trace pilot project provides the necessary proof-of-concept for the technical solutions.

Keywords: Product assembly, composite product, product agent, track and trace, middleware

Table of contents

1. Introduction

Imagine a world, where you would no longer have to search for days in order to find the old user manual of your video recorder that you saw the last time two years ago. Or that your vacuum cleaner could tell by itself what dust bag model is the one that you should buy. Or that the book that you ordered by Internet from another continent instead of arriving too late for the deadline when you need it, will inform you in time for you to change the plan and use courier deliveries to get your book on time. In this paper, we will present an information system architecture that enables these scenarios to become reality using already available technology. The information system architecture has been piloted in an industrial tracking context, where the benefits of it are immediate.

Product life-cycle management requires that product information should follow the product from when it is planned and manufactured, through its time of use until the time of disposal. Current practices of conveying all this information by paper documents accompanying the product is not a viable solution. This is because of the huge amount of data needed in different phases of the life cycle. Product information also changes during the life cycle of a product item. Handling the evolution of this information is not an easy task, especially when done on a product item level.

In order to handle information during the whole life cycle of a product item, we propose using an agent-based architecture where each product item has a corresponding "virtual counterpart" or agent associated with it. Agents provide services for their physical counterparts. For instance, it would be possible that a shipment could automatically react to unexpected problems on the way. If the ship that the shipment was originally scheduled for is delayed, the shipment agent could try to get the shipment onto another one instead. Or if a critical shipment has been damaged, then the agent of a less urgent shipment could agree to change its destination and replace the damaged one.

Now that Internet is accessible almost anywhere (at least in the industrial context), product information is in theory available anywhere. Many companies already have existing web services, where they make product information accessible. The challenge for product data management is how to easily know where that data is and how to get access to it. A simple solution is that the manufacturer of a product would put an identifier on all products manufactured, which also includes sufficient information about the Internet address where the product (or product item) information is found. We propose using a simple coding of the format ID@URI [Främling, 2002; Huvio et al., 2002; Kärkkäinen et al., 2003b], where the URI is an Internet address owned by the manufacturing company and the ID part can be a manufacturer-specific identifier. This identification is, by definition, globally unique. The uniqueness of the URI part is guaranteed by the Domain Name System (DNS) used on the Internet, while the manufacturing company should be able to guarantee the uniqueness of the ID part.

Some of these scenarios have been implemented at the Helsinki University of Technology. In this paper we show how intelligent physical objects can be built with only simple amendments to the present IT services. In Section 2 we start by analysing current practices, which are based on what we call the "transaction model". Then we describe the "agent model" proposed from the conceptual, functional and implementation points of view, followed by discussion and conclusions.

2. Agent-centric life cycle management

The life cycle of a product can typically be divided into the following phases: design, production, use, service & maintenance and retirement (or EOL, End-of-Life). Product data is created, modified and used during all these phases. With the way that product data is currently handled, it is difficult to be sure that the right information is indeed available during the phases of the product life cycle.

2.1 How life-cycle data is managed now

During the production phase, the product may be manufactured and transported by many different companies as it passes through the supply chain. The current way of managing product data in the supply chain is based on making the information follow the same path as the products themselves. In a normal supply chain, this may mean that the information has to pass through the information systems of many different companies. If the information is printed onto paper, then the information flow indeed has to follow the products themselves in order to be accessible when needed. However, treating printed information requires manual work and is error-prone. This is why at least bigger companies are trying to get rid of the printed information and replace it by digital documents instead. Since most companies already have product information in some digital form, it would seem that transmitting data digitally would be the most efficient procedure.

Digital documents are typically transmitted from the information system of one company to the information system of the other company using EDI messages or some file-transfer protocol [Angeles and Nath, 2001]. EDI messages are usually batch processed once or twice per day due to how the information systems are implemented and to the cost to pay per message [Witt, 1998; Linthicum, 2001]. This means that the arrival of product data is no longer necessarily synchronized with the arrival of the product itself (Figure 1), which causes extra overhead for the companies handling the products [Johnston&Yap, 1998; Scholz-Reiter&Höhns, 2003].

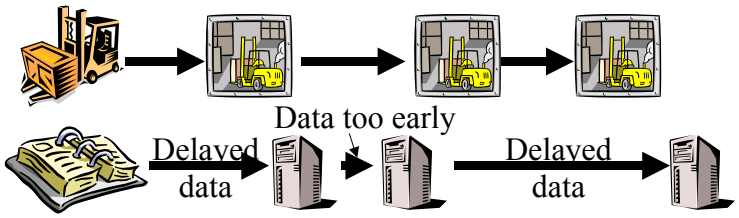

Figure 1. Lack of synchronisation between product data and the products themselves when using batch processing (the "transaction model") in company-to-company information transfer.

Handling product information in this way is problematic due to several reasons:

- Many companies in the supply chain may not need the information for their own activities, but they still have to be able both to receive and transmit the information to all their partners. This is typical for transportation companies, who in most cases do not need the information at all or only a fraction of it.
- All companies have to be able to communicate with each other. If one of the companies involved is unable to receive and transmit the information, then the information flow is interrupted.
- EDI communication is usually long and expensive to set up between two partners, so it makes collaboration links more rigid than they would otherwise need to be [Johnston&Yap, 1998].

- Transmitting the information by batch processing imposes a limit on how quick the supply chain can be. If information is passed in two batches per day, for instance, that gives an upper limit of two or less inter-company transfers per day.
- Integrating current EDI-based solutions is too expensive for most small companies [Timm et al., 2001]. Therefore many interesting collaboration opportunities become impossible to realise in practice when small companies are involved.
- The same product data may be copied to a great number of databases. Managing changes in the product data quickly becomes a nightmare; which databases should be updated, which one has the latest status etc.?

One of the attempts to overcome these problems has been establishing **portals** for doing the communication instead. Portals are Internet-based services located at a known computer, which is usually accessible to pre-determined partners. Big companies, who let their sub-contractors and clients access the portal through a web browser interface, have created most portals. A portal is more or less real-time in the sense that sub-contractors can update information at any time and clients can access information at any time. Portals are centralised solutions, which are convenient for the company who owns the portal since everyone else is using its system. However, this is only possible for very big companies who can dictate the conditions of their partners. For the sub-contractors and clients, who may have to work against several portals, it is not a good solution. Not only do they have to be able to cope with different user interfaces and requirements of different portals, but they may also lose control of their own data.

2.2 Agent-centric life cycle management

The notion of *virtual enterprise* [Aerts et al., 2002] describes a setting where supply chains become increasingly dynamic and network-like. Agent-oriented methods have been proposed as a solution for handling information [Fox et al., 2000] in the virtual enterprise. There is no universal agreement on what an agent is, but common aspects to most definitions seem to be that an agent should be autonomous, social, reactive and pro-active [Wooldridge&Jennings, 1995; Jennings&Wooldridge, 1998; Helin, 2003]. Autonomy signifies that agents operate without direct intervention of humans or others. Social ability means that agents interact with other agents via some communication language. In order to be reactive, agents perceive their environment and respond in a timely fashion to changes that occur in it. Finally, agents do not simply act in response to their environment; they are also able to exhibit goal-directed behaviour by taking the initiative (pro-activity).

Agents have been used for representing the participants of the supply chain, e.g. order acquisition agents, logistics agents, transportation agents, scheduling agents etc. [Fox et al., 2000]. The purpose of the agent architecture is typically to model, simulate and analyse supply chain operations in order to achieve better control of them [Scholz-Reiter&Höhns, 2003; Szirbik et al., 2003]. Agents have also been successfully applied to manufacturing processes [Gou et al., 1998; van Brussel et al., 1999; Brandolese et al., 2000; Cavalieri et al., 2000; Langer&Alting, 2000; Huang et al., 2002]. Product items can have associated agents [Holmström et al., 2002; Kärkkäinen et al., 2003c], which can greatly simplify access to product information. It can also simplify updating product information in tracking applications, for instance. In a multi-company setting, agents usually communicate over Internet connections.

Internet has become nearly ubiquitous for companies in all developed countries, making point-to-point connections obsolete. In places that are not equipped with an Internet connection, it is usually possible to create one for a moderate cost through the normal wire

telephone network, the GSM telephone network or through a satellite link. Internet makes it possible to access information located anywhere in the world. So if Internet access is available, there is no point in moving all product data along with the physical product. It is sufficient to make the data available on the Internet and to make it sure that there is a link from the physical product to the corresponding information. A challenge is that the link should be valid for the whole product life cycle. The information should also be constantly available (24/24h, 7/7days). Data security is also important, which is treated in a later section.

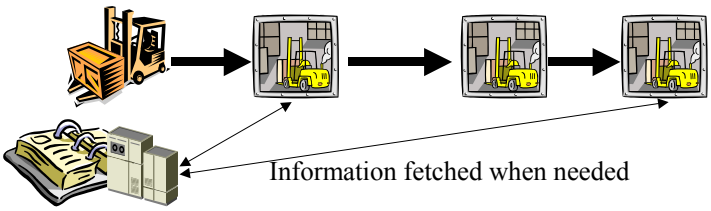

Figure 2. The "agent model" for real-time access to product information.

In the agent model, information is fetched and/or updated only when needed. The fetched information depends on who is asking for it, which means that only data that is useful for the inquirer is transmitted. Information access can be split into two main functions, namely:

1. *Accessing product data*. Access to product data is currently mainly done through web sites. Typical product data that needs to be accessed are user instructions, maintenance records, assembly instructions etc.
2. *Updating product data*. Typical updates concern tracking of shipments, maintenance records, status monitoring of machines etc.

These functions can usually be handled in a single-company setting, but it becomes challenging to make these solutions work in a multi-company setting [Luckham, 2002].

With the agent model, all information requests for a given physical product item is available at one single address on the Internet. It is the *product agent* that handles information requests and maintains the product data and links towards other sources of information about the product item. Therefore, the link between the physical product item and the corresponding agent plays a critical role in the system, which is developed in the next section.

Moving to the agent model can be compared with the big paradigm shift in computer programming during the 1980's. The old procedural programming paradigm changed into an object-oriented paradigm. A main reason for this was that object-oriented programming makes it easier to manage data and functionality of a program by concentrating them around the object-concept. This means that anyone (usually another object) that has a reference to the object can access information about the object through the object's methods. In software engineering, object-oriented programming has become dominant.

A Multi-Agent System (MAS) could be considered as an evolution of the object-oriented paradigm [Brandolese et al., 2000]. Objects of object-oriented programs often have some physical counterpart, such as a machine in a factory simulation model [Gou et al., 1998]. Therefore it is quite natural to apply object-oriented thinking to managing product information [Jennings&Wooldridge, 1998; Langer&Alting, 2000]. Objects are usually accessible only inside a computer program, while agents are usually implemented as distributed services that communicate through some public protocol like RMI [Sun Microsystems, 2002b, 2003], Corba [Orfali et al, 1997], SOAP [W3C, 2000], ebXML [ebXML, 2003] or Jini [Oaks&Wong, 2000]. These protocols all make agents accessible through the Internet [Främling&Holmström, 2000; Aerts et al., 2002; Goncalves et al., 2003]. Agent communication frameworks like FIPA (Foundation for Intelligent Physical Agents)

[FIPA, 2003] offer additional functionality, e.g. service priority negotiation between agents [Helin, 2003].

2.3 Linking products and information together; the ID@URI concept

In order to be universally applicable for linking together physical product items and their product agents, it is necessary that the physical product item have a globally unique identifier. This is one goal of the Auto-ID centre at MIT, with whom we share many research objectives. Still, there are also some significant differences. The Auto-ID centre is developing new standards that need to be widely accepted before they become universally usable. Information about the Auto-ID approach can be obtained at http://www.autoidcenter.org/. The ID@URI approach uses existing (and future) standards, which makes it immediately usable.

Other possibilities to ensure ID uniqueness exist. One possibility would be to use some existing infrastructure capable of dynamically generating globally unique identifiers [Järvinen, 2002]. The Jabber protocol for chatting applications (http://www.jabber.org) is an example of this kind of infrastructure. Jabber Id's (JID) are automatically generated and their uniqueness is assured by the Jabber server infrastructure. However, JID has no connection to existing identification standards and requires a well-established server infrastructure in order to be useful.

The Finnish company Stockway (http://www.stockway.fi/) has patented a peer-to-peer infrastructure, where product agents are not linked to any specific server machine or company. Instead, the Stockway server infrastructure maintains product item-specific link lists, which are accessible through any server that has any information about the product item. Stockway product identifiers are made up in a hierarchical fashion, where any existing member of the hierarchy can allocate new identifiers and integrate new members from his own identifier subspace. This approach is interesting, but it is still too early for us to evaluate the potential impact of this technology. It might also be partially in conflict with the concept of unique product agent presented here.

2.3.1 Description of ID@URI concept

An e-mail address-like notation "ID@URI" has been chosen for creating the link between a physical product item and its product agent. The URI part is a domain name of a computer, whose uniqueness is guaranteed by the DNS [Oat Systems&MIT Auto-ID Center, 2002]. The ID part then only needs to be a locally unique identifier inside the address space of the URI. Just like e-mail addresses are globally unique by using DNS-unique URIs and locally unique user names, the same is true for product item identifiers. Our solution therefore implements similar functionality as Auto-ID centre's Electronic Product Code (EPC) [Brock, 2001] and Object Name Service (ONS) [Oat Systems&MIT Auto-ID Center, 2002].

The URI is directly the address at which the physical product item's product agent is located. In order to assume product life cycle management, the URI would normally be the address of a server machine that belongs to the company that manufactured the product. However, in some applications it might be useful to use some other URI than the manufacturer's URI. One example could be tracking of book orders, where the URI could point to the receiver's tracking agent instead.

If the URI belongs to the manufacturing company, then it should be possible to make sure that the ID part is unique. The ID can be generated locally at the manufacturing company when the product item is created. If existing Radio Frequency Identification (RFID) tags are

used, then their serial numbers should be globally unique by convention. Global Trade Identification Number (GTIN) numbering standards [EAN Int., 2001] or existing serial numbers can be used as well. ID@URI does not impose any particular identification technology, since the ID@URI string can be read and written as a barcode, a magnetic stripe, an RFID tag or using other technologies.

2.3.2 Industrial example: using ID@URI for global, company-independent tracking

The ID@URI concept has been piloted and is now in industrial use for tracking shipments in global investment projects [ISI Industry Software, 2003]. Such projects typically involve numerous sub-contractors and the transport of all needed parts might involve tens of different transportation companies. When using current tracking systems that are based on transportation company-specific tracking numbers, it is nearly impossible to track all shipments. Even tracking one shipment might involve several transportation companies. This means that the project manager has to log in to web-based tracking systems of several companies, which all use different tracking numbers (Figure 3). If the number of shipments to track counts in hundreds or thousands, the task becomes impossible. [Kärkkäinen et al., 2003a]

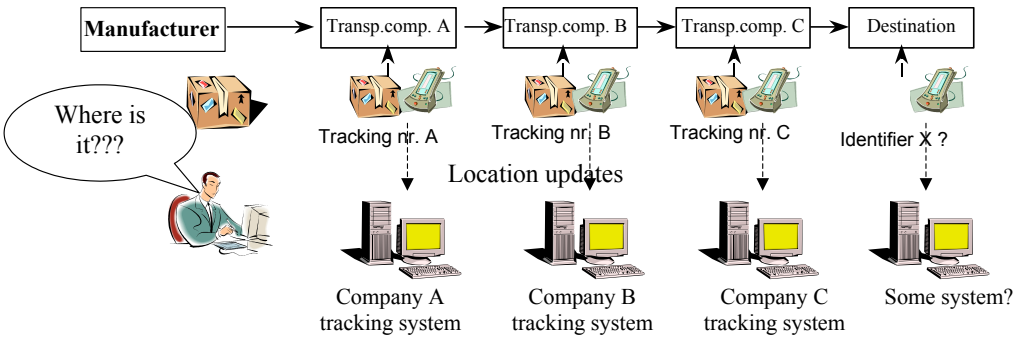

Figure 3. Use of current tracking systems.

Often it is not useful to do tracking continuously, the main issue would be to get automatically notified about exceptions to some pre-defined shipping schedule. Doing this with current tracking systems would require information system integration between many different companies, which is not feasible in practice. Especially for the transportation companies this could mean information system integration with a huge number of different companies. Furthermore, such integration is not possible for small companies or for individuals.

Using the ID@URI concept, the shipment itself knows who is interested in tracking its location; it is the agent at the Internet address indicated by the URI part. In our current implementation of the system, called the Dialog platform, tracking is implemented as shown in Figure 4.

The Dialog platform currently contains two software components. The first component is the "product agent" that acts as the tracking agent in Figure 4. The second component is the one being used at the checkpoints where the shipments pass. The checkpoint component can handle both barcode and RFID tags. In the case of barcode identifiers, the ID@URI is written using the Code-128 barcode standard [AIM, 2002]. When using RFID tags, the tag serial number has been used as the ID part, while the URI part has been written into the tag read/write memory.

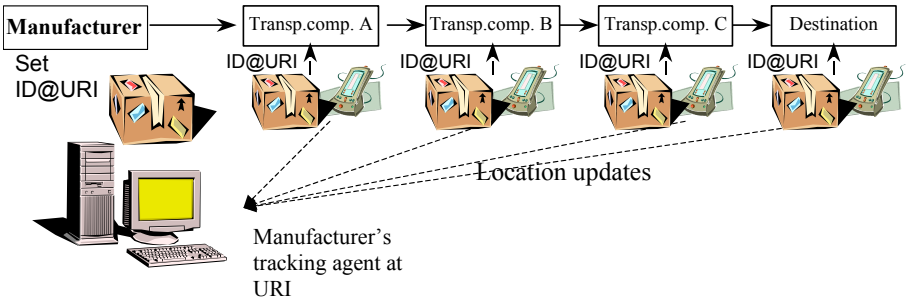

Figure 4. ID@URI based tracking.

Installation and set-up times of each component is typically less than five minutes, which includes automatic creation of database tables needed by the system in the manufacturing company's databases. After this installation, location updates arrive directly in real-time to the manufacturer's database, where it is possible to react to detected problems, such as schedule delays or damaged shipments.

In practice, many companies do not use one single number for identifying their shipments. Since modifying existing and operational numbering systems is usually not desirable, it is easier to create a link between the ID used and the numbering system used internally by the company. This linking feature is included in the Dialog platform, which avoids making changes to existing numbering systems in the companies. Still, generating unique IDs for product items might require introducing a new numbering system in many companies. Using RFID tags is one way of avoiding this since RFID tags already have unique identifiers.

2.4 Accessing and updating product information

For physical products identified by the ID@URI notation, the product information can be made available everywhere, where Internet access is available. As for the tracking example, the URI part directly tells where to find the information, while the ID part tells what physical product item the information is asked for (Figure 5). The software component at the given URI can therefore act as the product agent of the particular product item.

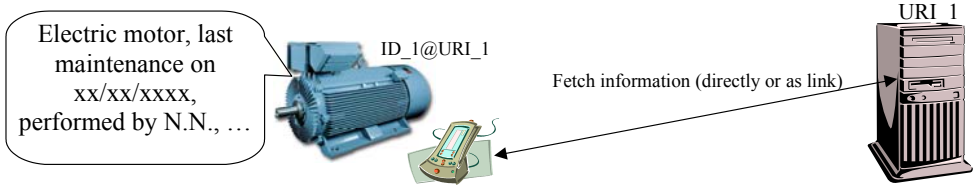

Figure 5. Accessing product information through ID@URI.

In the Dialog platform, product information can be obtained in two ways:

- *Direct information*: product information is sent directly as text, Hypertext Markup Language (HTML), eXtended Markup Language (XML) or any other format that can be understood by the asking system.
- *Link to information*: a link to another Internet address where the information can be accessed, typically as a HTML link including the protocol to use (http, ftp, …).

Direct information is useful especially when the amount of information is not too big and when it is easy to store the whole information directly in a database. In the current implementation, there are no access or security checks implemented for direct information, so it is typically to be used for non-sensitive data. It is, however, possible to encrypt the data being transmitted using standard Secure Sockets Layer (SSL) connections. SSL was

originally developed by Netscape Corporation, but is now supported by all web browsers and many other programs.

Providing product information by a link to the address where the information is makes it possible to use existing web pages and web-based user interfaces. The advantage of this approach not only comes from using existing infrastructure, but also from the fact that authentication and other security aspects can be taken into account. Let us suppose that the information about the electric motor in Figure 5 is given as a link to an existing web page, which is secured by password. In that case, the user cannot access the information unless he has the appropriate access rights.

Even though product information can be *retrieved* through links and existing services, this is not always the case for *information updates*. Location updates sent in a tracking application are an example of a situation where both the identity of the physical item and the indicated location might need to be validated. Validating the identity of the physical item is easy to do by checking that the given ID has indeed been issued at the indicated URI. In applications where supplementary validation is needed, two-key validation can be used as provided by RSA, DSA [NIST, 2002] and other encryption systems. The main problem is that one encryption key needs to be stored with the physical item itself. This is possible both using barcodes and RFID tags, supposing that the supplementary data required fits into the available space.

Validating the location, i.e. the identity of the party asking for or updating information can also be done using two-key validation, possibly combined with a Public Key Infrastructure (PKI) based solution [Biennier, 2003]. Another approach is to use the identifier of the reading device, which is registered among those authorized to access product information. Choosing what technique to use still seems to depend largely on the application area, so the current Dialog platform does not include any of these. Instead, we have selected to keep the implementation as open as possible so that different data security protocols would be easy to use without creating a need to modify the system itself.

Updating the location of the physical item is not the only imaginable product information update. Examples of other information updates are damage reports, maintenance reports, installations of new parts etc. Just as for product information requests, the Dialog platform provides two possibilities for updating product information from remote locations:

- *Direct information*: product information is updated directly as text, Hypertext Markup Language (HTML), eXtended Markup Language (XML) or any other format that can be understood by the product agent. The information can be sent as a part of a location update message, which means that all information updates are associated with a given timestamp and a given location or user.
- *Through link*: if the product information is given as a link to existing maintenance or other web-based services, then those can be used for updating the information.

The same security considerations apply for information updates as for access to information. The technology needed for implementing the needed level of security exists and can be used in our approach. However, obtaining a high level of security also means managing encryption keys, PKI certificates and other security-related information. Managing this information is not an easy task. Therefore it is essential to identify an “appropriate” level of security depending on the application and the nature of the information. For instance, it is hard to see why accessing user instructions of a mobile phone would require validating the phone’s identity or validating the identity of the person asking for the information.

2.5 Composite products

Both products and shipment units used in transport usually contain parts that come from many different companies. This signifies that physical product items become parts of each other, so the information related to them becomes interconnected. Often the product individual forms a tangled hierarchy, in which a product individual consists of a set of other product individuals. Such relations are handled by the concept of *composite products* [Aerts et al., 2002; Främling, 2002], where the uppermost product is at the top of a product containment hierarchy as in Figure 6.

Composite products usually do not change too much during the product life cycle of most products. A major exception to this is shipments, where containment hierarchies are created, modified and deleted very rapidly. We can then speak about a *shipment life cycle*, where the uppermost element of the hierarchy changes every time that the vehicle transporting the shipment changes. As boxes are unloaded from one container and put into another, parts of one containment hierarchy are removed and merged into another containment hierarchy. It is essential to notice that although product life cycles and shipment life cycles have very different life duration, they can still be represented by the same composite product concept.

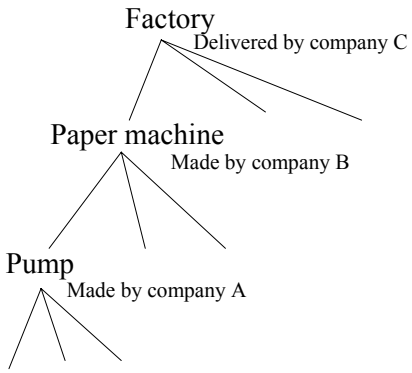

Figure 6. Example of composite product hierarchy.

RFID technology has been presented as a universal solution at least for handling shipment contents since the entire contents of a package, pallet or container could be read in one single operation. However, this is rarely possible in practice with current technology due to collisions between the tags when reading them, insufficient reading distance or electrical interference with metals or other material. Furthermore, reading all the tags does not directly indicate if the tag is on the container, on a package or on a product item. Filtering out such information might require accessing the product data related to all the tags read, which might be time-consuming and error-prone. This is one reason why the notion of composite products is useful also for handling shipments.

When speaking about product life cycle, it is necessary to identify who has the responsibility for maintaining life cycle data. In most cases, it is the manufacturer of the product who has this responsibility, even though exceptions exist. In a composite product, different parts of the product may come from different manufacturers. When the product is created in a supply chain, the partners of the supply chain should agree who has the responsibility for what part. The company who has the responsibility for a part then puts his identifier onto the part and links it to his product information system. If this part is used in a bigger assembly, then the manufacturer of the assembly can put his identifier on the assembled part. This procedure is illustrated in Figure 7.

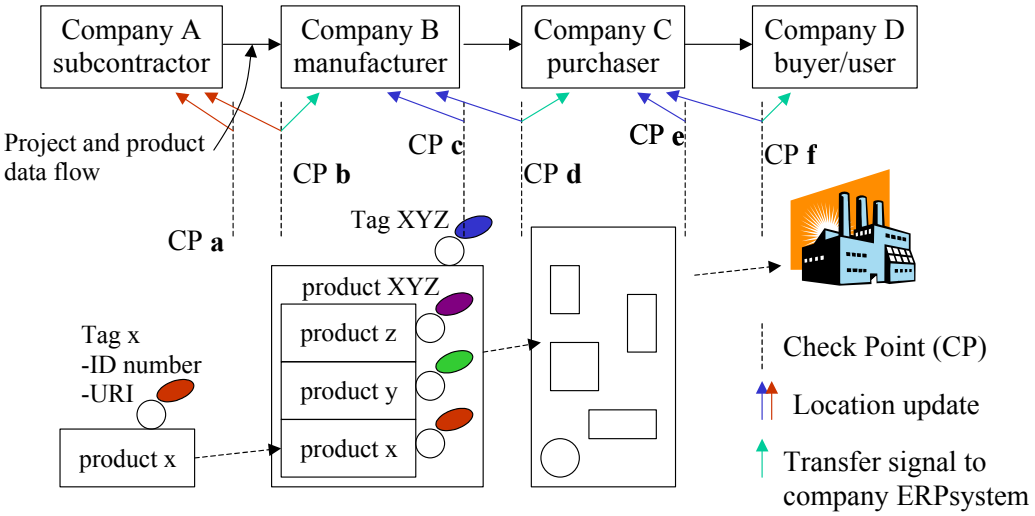

Figure 7. Example of how a composite product is created along the supply chain.

Keeping a link to the product information of all parts of an assembly is important for maintenance operations, like replacing a part with another. When a part is replaced with a part manufactured by another company than the original one, it is sufficient to change the product containment hierarchy. This functionality is even more important in logistics, where container contents can change rapidly.

Figure 8 shows how composite products are handled using the ID@URI concept. The whole containment hierarchy of a composite product can be managed through ID@URI identifiers, which greatly reduces the need to modify Enterprise Resource Planning (ERP) or other existing information systems.

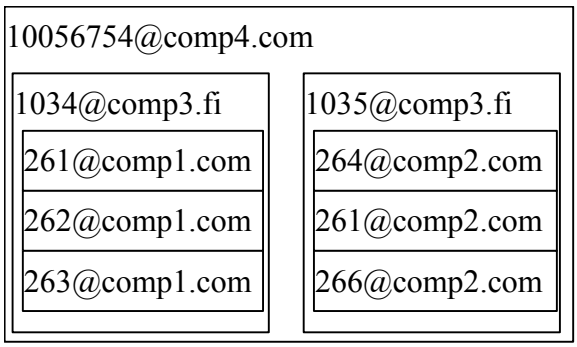

Figure 8. Example of composite product.

Figure 9 illustrates how an information update, e.g. location update, is propagated through the containment hierarchy of the composite product in Figure 8. In the case of a location update, it is created somewhere along the supply chain. In many cases, only the identifier of the "outermost" part is accessible for reading. The outermost part is also the one at the top of the containment hierarchy, which makes it easy to propagate the location update to all the parts of the containment hierarchy, even though the different parts have different "owners".

Containment hierarchies should be bi-directional, which means that parts know what other parts they are composed of and parts also know what composition they belong to. Bi-directionality makes it possible to perform an information request or update for the whole composite product by reading the identifier of *any* part of the composite product. Bi-directional links are also essential when changing parts of the composite product or when breaking it up completely.

Updating the information structure can be done explicitly or implicitly. In a transportation context, for instance, reading the identifier of a part that is not at the top of the containment hierarchy could indicate that the part has been taken out of the transportation units higher up in the containment hierarchy. In a maintenance context, changing a part into another requires doing an explicit update of the containment hierarchy.

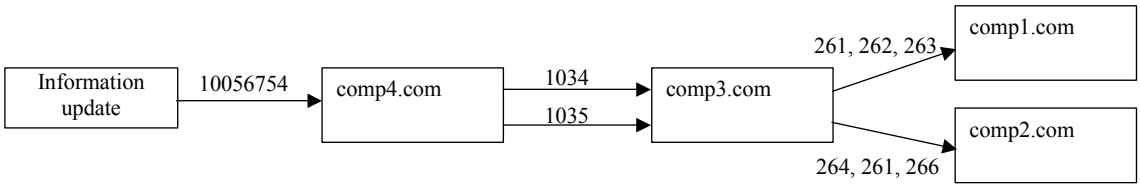

Figure 9. Propagation of information updates through the composite hierarchy.

In the future, it can be expected that the need for explicit updates will decrease. With the evolution of RFID tags, BlueTooth chips or similar technologies product items will be able to communicate directly with each other. This means that composite products could self-detect their containment hierarchy. Such technology is being tested already for automatically detecting the contents of shopping carts or shop shelves [Cole, 2002].

2.6 Technical challenges

The software implementation developed for testing the concepts presented here is called the Dialog platform. All presented functionality can be implemented by two software components, which are some tens of kilobytes each (i.e. same size as a typical small picture on a web page). They are written in Java, so a Java-enabled platform needs to be available in addition to the Dialog software components themselves.

The first software component is called the *client*. A client component is used for reading product identifiers and connecting to the product agent whose Internet address is indicated by the ID@URI product identifier. The second software component is called the *server* and implements the product agent functionality. A server component has access to a database through Java Database Connectivity (JDBC) [Sun Microsystems, 2002a], which enables it to communicate with virtually any existing database product, including those used by most ERP systems.

For the client component, the only mandatory set-up information is the physical location, which is entered by the user when the client is started for the first time. Server component set-up only requires indicating the JDBC connection parameters for connecting to the database to use. Server set-up can automatically create all needed database tables (currently four). Tracking and support for composite products is immediately operational. Access to product information is operational as soon as the information has been inserted into the database. The information can either be inserted directly or as a HTML link in a field of one of the created tables. Installation and set-up of both client and server can be done in less than five minutes.

Since these two software components implement the whole chain "product identifier" ► "product agent" ► "database", there is no need for any further components for implementing the functionality described in this paper. Even accessing and updating information of composite products as in

Figure 9 is possible by letting product agents act both as clients and servers in a peer-to-peer fashion.

Information exchange between clients and servers is performed by *messages* [Monson-Haefel&Chappell, 2001]. The implementation of the functionalities previously described is done by the following messages:

- LocationUpdateMsg
- ProductInformationRequestMsg
- IdentifierLinkMsg
- GetCompositeComponentsMsg
- AddToCompositeMsg
- RemoveFromCompositeMsg

A *LocationUpdateMsg* contains the product identifier (ID@URI), a timestamp and a location. It can also contain free-format data for transmitting supplementary information about the product item, such as damage reports, maintenance information or whatever. Free-format data here signifies that it can be text, HTML text, an XML document containing links to other documents or even binary data.

ProductInformationRequestMsg contains product identifier (ID@URI) and possibly a timestamp and a location. It returns a message with the product identifier and the product information. As explained in Section 2.4, the product information can either be returned directly or as a link to where the information can be accessed by the client itself or by an Internet browser. The product information also contains a field that indicates if it is a composite product, i.e. if it has a containment hierarchy below it. Finally, if the product is a part of a composite product, then the ID@URI of the "parent" product is returned.

IdentifierLinkMsg contains a product identifier (ID@URI), a list of reference numbers and possibly a timestamp, a location and free-format data for supplementary information. It is used to link the ID part in ID@URI to existing reference numbers used in the owning company's ERP systems. For instance, when using serial numbers of RFID tags, they have to be linked to company internal references in order to be useful. This would typically be done when the RFID tag is attached to the physical product item.

GetCompositeComponentsMsg makes it possible to browse downwards through the containment hierarchy of composite products. This information could also have been included in the response to a *ProductInformationRequestMsg*, but that would cause transfer of useless data in cases where containment information is not interesting for the client.

AddToCompositeMsg contains a product identifier (ID@URI) and a list of ID@URI identifiers to add to the containment hierarchy of the product. It can also include a timestamp, a location and free-format data for supplementary information.

RemoveFromCompositeMsg contains a product identifier (ID@URI) and a list of ID@URI identifiers to remove from the containment hierarchy of the product. It can also include a timestamp and a location.

The current implementation (see http://dialog.hut.fi) can use both Java RMI messaging [Sun Microsystems, 2002b, 2003] and XML-based communication through the SOAP protocol [W3C, 2000]. In order to make the system as open as possible, a major challenge is to standardise these messages so that any software producer could implement them and communicate with each other successfully. We will try to propose such standards ourselves and take into use such standards as soon as they emerge.

What comes to performance issues, the proposed system is lightweight in most aspects, which is confirmed by load tests performed. However, communication speed and capacity

also depends on the communication protocol used. For instance, RMI seems to be at least twice as quick as SOAP [Järvinen, 2002]. The URI part of the product identifier can also contain a protocol part, which is useful if no single best choice protocol can be defined. As it seems unlikely that such a best solution could be identified for all application areas, we will instead try to establish a best-practices list based on experience from industrial applications.

3. Discussion and conclusions

Currently, product data is often partially duplicated into many different places, which makes it difficult to access the data and keep it up-to-date. Using the agent model for accessing information about tangible products avoids this by concentrating the information to the product agent. The ID@URI identifier can be compared to an object reference in an object-oriented program, because it is a reference to the product agent that corresponds to the physical product item. Composite products are similar to object tree structures used in many computer programs, e.g. drawing programs where parts of the drawing may be combined into groups. Just like an object-oriented program is essentially built up by object references and communicating objects, ID@URI and communicating agents can do the same thing.

Agent-based computation has already proved that it is a good solution to the information handling needs in supply chain management. The Dialog platform has been piloted for tracking and tracing in a multi-company delivery network for an international project of a Finnish company. This installation offers a proof-of-concept that the ID@URI and product agent-based solutions work in an industrial context.

The proposed system is **open**. This means that solutions based on the methods presented here can be built without any need to pay royalties or offend any patents known by the authors. Furthermore, the concepts presented here do not require the involvement of third-party actors in order to be usable. It is therefore easy for companies of all sizes to start using these concepts, without losing future upgrading and scalability possibilities when collaborating with other companies.

Communication messages proposed here are not yet standardized, neither have we identified any suitable existing standards. As such standards evolve, they are easy to take into use since they only concern a part of the system. We believe that standards will appear rapidly as an increasing number of companies will start using concepts presented here or similar ones, such as those of Auto-ID centre.

Another critical factor for adopting the agent model is Internet availability and reliability. For a pure agent-based implementation, Internet has to be nearly 100% reliable, as well as the server machines that host product agents. Recent attacks against the Internet have shown that Internet can be vulnerable. Therefore the agent model should be implemented in activities where minor downtimes are not critical or can be compensated for through message persistence mechanisms [Monson-Haefel&Chappell, 2001]. Track-and-trace is one activity where Internet failures can be compensated for. Accessing user instructions is another activity where nearly guaranteed real-time availability is sufficient. For maintenance of machines or other devices, current Internet reliability should also be sufficient in most cases. This could be the case also for many (or most) other activities.

In practice, it seems like the first applications will be in an industrial context, not only due to the economical benefits, but also due to changes in legislation concerning traceability of raw materials and product life-cycle management in general. The biggest challenge on the way might be to achieve a shift in the general viewpoint on service provision and information management.

Acknowledgements

This research has been supported mainly by Tekes, the National Technology Agency of Finland. We also thank the companies who have contributed in the financing and validation of the methods described here.

References

Aerts, A.T.M., Szirbik, N.B., Goossenaerts, J.B.M. (2002). A flexible, agent-based ICT architecture for virtual enterprises. *Computers in Industry*, Vol. 49, No. 3, pp. 311-327.

AIM, (2002). *Code 128 overview*. AIM Global. Available online (December 13th, 2002): http://www.aimglobal.org/standards/symbinfo/code_128_overview.htm.

Angeles, R., Nath, R. (2001). Partner congruence in electronic data interchange (EDI)-enabled relationships. *Journal of Business Logistics*, Vol. 22, No. 2, pp.109-128.

Biennier, F. (2003). Security Integration in Inter-Enterprise Business Process Engineering. In Jagdev, H.S., Wortmann, J.C., Pels, H.J. (eds.) *Collaborative Systems for Production Management*, Kluwer Academic Publishers, pp. 207-217.

Brandolese, A., Brun, A., Portioli-Staudacher, A. (2000). A multi-agent approach for the capacity allocation problem. *International Journal of Production Economics*, Vol. 66, pp. 269-285.

Brock, D.L (2001). *The Electronic Product Code (EPC) - A Naming Scheme for Physical Objects*. MIT Auto-ID Center White Paper, January 2001. Available online (December 13th, 2002): http://www.autoidcenter.org/research/MIT-AUTOID-WH-002.pdf

van Brussel, H., Bongaerts, L., Wyns, J., Valckenaers, P., van Ginderachter, T. (1999). A Conceptual Framework for Holonic Manufacturing: Identification of Manufacturing Holons. *Journal of Manufacturing Systems*, Vol. 18, No. 1, pp. 35-52.

Cavalieri, S., Garetti, M., Macchi, M., Taisch, M. (2000). An experimental benchmarking of two multi-agent architectures for production scheduling and control. *Computers in Industry*, Vol. 43, pp. 139-152.

Cole, P.H. (2002). *A Study of Factors Affecting the Design of EPC Antennas & Readers for Supermarket Shelves*. MIT Auto-ID Center White Paper, June 2002. Available online (December 13th, 2002): http://www.autoidcenter.org/research/MIT-AUTOID-WH-002.pdf

EAN International (2001). *Global Trade Item Numbers (GTIN), Application Guideline*. EAN International. Available online (January 8th, 2002): http://www.ean-int.org/

EbXML (2003). *ebXML - Enabling A Global Electronic Market*. Available online (October 14th, 2003): http://www.ebxml.org/

FIPA (2003). Foundation for Intelligent Physical Agents. Available online (November 7th, 2003): http://www.fipa.org/

Fox, M.S., Barbuceanu, M., Teigen, R. (2000). Agent-Oriented Supply-Chain Management. *International Journal of Flexible Manufacturing Systems*, Vol. 12, pp. 165-188.

Främling, K., Holmström, J. (2000). A Distributed Software for Collaborative Sales Forecasting. In: *Proceedings of the Management and Control of Production and Logistics MCPL'2000 conference, 5-8 July 2000*. Editor: Binder, published by IFAC Publications, Elsevier Science Ltd.

Främling, K. (2002). Tracking of material flow by an Internet-based product data management system (in Finnish: Tavaravirran seuranta osana Internet-pohjaista tuotetiedon

hallintaa). *Tieke EDISTY magazine*, No. 1, 2002, Publication of Tieke (Finnish Information Society Development Centre), Finland.

Goncalves, G.M., de Sousa, J.B., Pereira, F.L., Dias, P.S., Santo, A. (2003). A framework for e-cooperating business agents: An application to the (re)engineering of production facilities. In: Jagdev, H.S., Wortmann, J.C., Pels, H.J. (eds.) *Collaborative Systems for Production Management*, Kluwer Academic Publishers, pp. 189-204.

Gou, L., Luh, P.B., Kyoya, Y. (1998). Holonic Manufacturing Scheduling: Architecture, Cooperation Mechanism, and Implementation. *Computers in Industry*, Vol. 37, pp. 213-231.

Helin, H. (2003). *Supporting Nomadic Agent-based Applications in the FIPA Agent Architecture*. PhD thesis, Dep. of Computer Science Report A-2003-2, University of Helsinki, Finland. 200 p.

Holmström, J., Främling, K., Tuomi, J., Kärkkäinen, M., Ala-Risku, T. (2002). Implementing Collaboration Process Networks. *International Journal of Logistics Management*, Volume 13, Number 2, pp. 39-50.

Huang, B., Gou, H., Liu, W., Li, Y., Xie, M. (2002). A framework for virtual enterprise control with the holonic manufacturing paradigm. *Computers in Industry*, Vol. 49, pp. 299-310.

Huvio, E., Grönvall, J., Främling, K. (2002). Tracking and tracing parcels using a distributed computing approach. In: Solem, Olav (ed.) *Proceedings of the 14th Annual Conference for Nordic Researchers in Logistics (NOFOMA'2002)*, Trondheim, Norway, 12-14 June 2002, pp. 29-43.

ISI Industry Software (2003). *Consignment Tracking for Heavy Industry*. Available online (October 16th, 2003): http://www.isiindustrysoftware.com/news/kvaerner.html

Jennings, N. R., Wooldridge, M. J. (1998). Applications of Intelligent Agents. In: N. R. Jennings and M. Wooldridge (eds.) *Agent Technology: Foundations, Applications, and Markets*, Springer Verlag, pp. 3-28.

Johnston, R. B., Yap, A. K. C. (1998). Two-Dimensional Bar Code as a Medium for Electronic Data Interchange. *International Journal of Electronic Commerce*, Vol. 3, No.1, pp. 86-101.

Järvinen, V.P. (2002). *Language independent software communication in distributed applications*. M.Sc. Thesis, University of Helsinki, November 2002.

Kärkkäinen, M., Holmström, J., Främling, K., Artto, K. (2003a). Intelligent products - a step towards a more effective project delivery chain. *Computers in Industry*, Vol. 50, No. 2, pp. 141-151.

Kärkkäinen, M., Främling, K., Ala-Risku T. (2003b). Integrating material and information flows using a distributed peer-to-peer information system. In: Jagdev H.S., Wortmann J.C., Pels H.J. (eds.) *Collaborative Systems for Production Management*, Kluwer Academic Publishers, Boston, USA, pp. 305-319.

Kärkkäinen, M., Ala-Risku, T., Främling, K. (2003c). The product centric approach: a solution to supply network information management problems? *Computers in Industry*, Vol. 52, No. 2, pp. 147-159.

Langer, G., Alting, L. (2000). An Architecture for Agile Shop Floor Control Systems. *Journal of Manufacturing Systems*, Vol. 19, No. 4, pp. 267-281.

Linthicum, D.S., (2001), *B2B application integration: e-business-enable your enterprise*. Addison-Wesley, Boston.

Luckham, David (2002). *The Power of Events*. Addison-Wesley, Boston, USA. 376 p.

Monson-Haefel, R., Chappell, D.A. (2001). *Java Message Service*. O'Reilly&Associates, Sebastopol, USA. 220 p.

NIST (2002). *Digital Signature Standard (DSS) and Secure Hash Standard (SHS)*. National Institute of Standards and Technology, 2002. Available online (December 13th, 2002): http://csrc.nist.gov/cryptval/dss.htm.

Oaks, S., Wong, H. (2000). *Jini in a Nutshell*. O'Reilly&Associates, Sebastopol, USA. 400 p.

Oat Systems & MIT Auto-ID Center (2002). *The Object Name Service, Version 0.5 (Beta)*. Available online (December 5th, 2002): http://www.autoidcenter.org/research/MIT-AUTOID-TM-004.pdf

Orfali, R., Harkey, D., Edwards, J. (1997*). Instant CORBA*. John Wiley & Sons, New York. 313 p.

Scholz-Reiter, B., Höhns, H. (2003). Agent-based Collaborative Supply Net Management. In: Jagdev, H.S., Wortmann, J.C., Pels, H.J. (eds.) *Collaborative Systems for Production Management*, Kluwer Academic Publishers, pp. 3-17.

Sun Microsystems (2002a). *JDBC™ Data Access API*. Available online (December 13th, 2002): http://java.sun.com/products/jdbc/

Sun Microsystems (2002b). *RMI Specification*. Available online (December 13th, 2002): http://java.sun.com/products/jdk/1.2/docs/guide/rmi/spec/rmiTOC.doc.html

Sun Microsystems (2003). *Java Remote Method Invocation (RMI)*. Available online (October 14th, 2003): http://java.sun.com/products/jdk/rmi/

Szirbik, N.B., Wagner, G.R., La Poutré, J.A. (2003). A generic framework for simulation of supply networks with bargaining agents. In: Jagdev, H.S., Wortmann, J.C., Pels, H.J. (eds.) *Collaborative Systems for Production Management*, Kluwer Academic Publishers, pp. 323-339.

Timm, I.J., Woelk, P.-O., Knirsch, P., Tönshoff, H.K., Herzog, O. (2001). Flexible mass customisation: Managing its information logistics using adaptive co-operative multiagent systems. In: Pawar, K.S.;Muffatto, M. (eds.): *Logistics and the Digital Economy*. Proceedings of the 6th International Symposium on Logistics, Salzburg, Austria, pp. 227-232.

W3C (2000). *Simple Object Access Protocol (SOAP) 1.1*. Available online (October 14th, 2003): http://www.w3.org/TR/SOAP/

Witt, C.E. (1998). Crossdocking: Concepts demand choice. *Material Handling Engineering*, Vol. 53, No. 7, pp. 44-49.

Wooldridge, M., Jennings, N.R. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, Vol. 10, No. 2, pp. 115-152.