Submitted for review in Computers in Industry

Agent-based Model for Managing Composite Product Information

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

*) Department of Computer Science and Engineering
Helsinki University of Technology
PO Box 5400, FIN-02015 HUT, Finland

**) Department of Industrial Management and Engineering
Helsinki University of Technology
PO Box 5500, FIN-02015 HUT, Finland

Abstract

The importance of product information management during the whole lifetime of the product has increased due to the technical sophistication of products as well as stricter governmental regulations for lifecycle management. Just sending the relevant product information downstream in the supply chain does not solve the challenges of product information management of complex products due to difficulties in updating the information and a risk of information overflow in the supply chain. This article describes an agent-based information management model that can be used for managing the information of complex products at a component level in a distributed manner. Further the paper presents an information management platform that can achieve information management requirements by using seven distinct messages.

Keywords: Product information management, product lifecycle management, product assembly, product agent, middleware

---

Corresponding author: Kary.Framling@hut.fi, Tel.:+358-50-5980451,
Fax: +358-9-451 3293.

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. Finally, the authors greatly appreciate the feedback from the anonymous referees that helped us improve the original paper.

1. Introduction

The importance of solid product information management practices and systems is increasing due to the intensifying technical sophistication of products, as well as the governmental regulations demanding efficient product lifecycle management (Töyrylä, 1999; Hamilton, 2001; Kärkkäinen et al., 2003c). Recent developments in information transfer and data storage capacities enable distributing vast amounts of product and component information forward in the supply chain and thus overcoming the problems associated with paper-based data transfer. However, in complex products this can lead to information overflow in the downstream supply chain, when the amount and sophistication of product components increases. Another challenge is to maintain the information up-to-date for the relevant supply chain members during the products' lifecycle (Kärkkäinen et al., 2003c). Therefore, Beulens, Jansen and Wortmann (1999) (as cited in van Dorp, 2002) have proposed that the supply chain should contain an *information decoupling point*. They argue that all information should not be sent forward in the supply chain. In (Jansen-Vullers et al., 2004), the concept of information decoupling point is developed further in the context of traceability and quality control in the food industry, introducing the notion of *traceability decoupling point*. At the traceability decoupling point the product data is aggregated together behind a *label*, such as "environmentally kind" or "animal friendly". After the traceability decoupling point, the detailed information can be retrieved using the label as a reference.

"Product centric information management" in which information regarding a product is retrieved over information networks when needed using unique product identities as references is one solution type complying with the information decoupling point principle (McFarlane et al., 2002; Wong et al., 2002; Kärkkäinen et al., 2003c). However, until now articles discussing product centric information management practices have focused on the principles of maintaining and updating information under a single product identity. The challenges associated with information management of complex products with complicated bills-of-material have remained unanswered.

In this paper, we will present in detail an information system architecture and components that enable the management of the lifecycle information of complex

products. The system components are simple to produce using already available technology. The system follows the principles of "product centric information management" but is operational also on the component and bill-of-materials level.

In the first section of the article we will present the current practices of product information management. Basic principles of software agents and their relevance to the challenges of product information management are presented in the second section. In the third section, an agent-based information management platform called Dialog is presented in detail. The final section proffers concluding remarks and directions for future research.

2. Challenges of managing the information of complex products

Current practices of managing product information in supply chains are reviewed in this section. The information about product information management presented here is based on a telephone survey of 36 Finnish firms from the industrial and retail sectors. The technologically most advanced firms were included in the survey sample. The firms were selected based on presentations at trade seminars and the knowledge of technological experts at the Finnish National Technology Agency (TEKES). The most relevant information from the survey is summarized here (see Kauremaa et al., 2004 for detailed results of the survey).

A typical technologically advanced product is processed by several different companies during its lifecycle. The most traditional way of managing product related information is conveying the information to the next downstream partner in the supply chain. 92% of the surveyed companies distributed product information to their customers. Only 30% of these companies relied still solely on paper-based information transfer. In a typical industrial supply chain, the information passes through a multitude of companies before reaching the final user of the product. Processing printed information in several nodes of the supply chain requires manual work and is prone to errors (Töyrylä, 1999). This is why most of the advanced companies have replaced printed information with digital data transfer means (70% of the companies conveying product information to their clientele).

The chronologically first possibility of conveying product information in electronic format was using EDI messages or some file-transfer protocol (Angeles and Nath, 2001). It was still used by 13% of the respondents. The logic of using EDI messages to convey product information is similar to the paper-based transfer, the information is sent to the next downstream supply chain partner (i.e. the primary customer).

Handling product information with electronic or paper-based messaging is problematic due to the following 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.

All companies have to be able to communicate with each other. If one of the companies of the supply chain is unable to receive and transmit the information, then the information flow is interrupted.

The product information that is sent and stored at various downstream companies is difficult to keep up-to-date. The producer of new information may not know what parties to inform about updates and thus companies with outdated information risk making decisions based on wrong information (Kärkkäinen et al., 2003c).

Transmitting all product information downstream may cause information overflow downstream in the supply chain (Beulens et al., 1999). This is especially true with complex products, in which the information related to the components of the product has also to be managed.

EDI communication is usually expensive and takes long to set up between two partners, so it makes collaboration links more rigid than they would otherwise need to be (Johnston and Yap, 1998).

Integrating current EDI-based solutions is too expensive for most small companies (Timm et al., 2001), therefore limiting the participation of small companies in the supply chain information exchange (Kärkkäinen and Ala-Risku, 2003).

The most popular way of overcoming these problems has been the use of Internet and Extranet technologies for transferring product information (45% of the respondents used these technologies to transmit product information to their customers). In practice, forerunner companies have developed portal applications through which

their customers can download the product information. Portals are Internet based services that can represent information from various systems in a single place, through a browser (Linthicum, 2001). Portals have proved to be efficient in making the product information available to supply chain participants and have thus grown increasingly common. However, portals are just the user interface representing the information to the consumer of the information. The issues related to the management of information, e.g. linking information to specific product items and handling the bill-of-materials of products, demand additional solutions.

The challenge of how to link the relevant information residing in several different data systems together while avoiding data overflow in the supply chain remains unanswered.

3. An agent-based approach to information management

During the review of current practices it was noticed that it is a formidable challenge to link the product related information to the products themselves (Kärkkäinen et al., 2003a; 2003c). Making the information of all the product components easily achievable without the risk of downstream information overflow proved to be especially challenging. Software agents were seen as one possible answer to these challenges.

Moving from the traditional model of product information management to the agent-based model is analogous to 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 methods declared in the object's public interface, while hiding the object's implementation (this is called *encapsulation*). In software engineering, object-oriented programming has become the dominant paradigm. As shown in (Främling et al., 2004), many of the key concepts of object-oriented programming also apply to agent-based product

information management. This is why we believe that similar gains can be expected by moving to agent-based product information management.

In the following parts of this section, we will first review the current use of software agents in industrial contexts and their linkage to physical products and then we will present the basics of how software agents can be used in product information management.

3.1 Software agents and the supply chain

Agent-oriented methods have been proposed for handling information in dynamic supply chains (Fox et al., 2000). 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 and Jennings, 1995; Jennings and Wooldridge, 1998). 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 various functions in 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 over the entire process (Scholz-Reiter and 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 and Alting, 2000; Huang et al., 2002). Agents in applications for these purposes often have some physical counterpart, such as a machine in a factory simulation model (Gou et al., 1998). Therefore, it is quite natural to extend agent-oriented thinking to the management of information related to individual products (Jennings and Wooldridge, 1998; Langer and Alting, 2000).

Software agents as a technology have a major strength over regular software objects, when connectivity across organisations is required. 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 and Wong, 2000). All of these protocols make agents accessible through the Internet, which allows the information managed by such agents to be available for all parties in multi-company networks (Främling and Holmström, 2000; Aerts et al., 2002; Goncalves et al., 2003). Dedicated agent communication frameworks like FIPA (Foundation for Intelligent Physical Agents) (FIPA, 2003; Helin, 2003) also exist but they are mainly used in academic environments.

3.2 Linking agents to products – the ID@URI concept

The agent-based approach to product information management has received growing attention lately (Holmström et al., 2002; McFarlane, 2002; Wong, 2002; Kärkkäinen et al., 2003c). A fundamental issue in this approach is how to associate the software agents with their physical counterparts. For creating this connection each physical product has to have a unique identity, which can be used as a reference to locate the product's agent on the Internet. Several possible solutions have been proposed.

One notable proposal is the EPC (Electronic Product Code) (Brock, 2001), combined with the Object Name Service (ONS) infrastructure (Oat Systems & MIT Auto-ID Center, 2002) developed by the Auto-ID Centre (www.autoidcenter.org). However, it is based on a naming service not yet available and the coding scheme is based on central allocation of codes, which may make the system rigid and give problems especially to small companies (Kärkkäinen et al, 2003c).

Another option is an infrastructure capable of dynamically generating globally unique identities (Järvinen, 2002). The Jabber protocol for chatting applications (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. But, since this approach is based on a well-established server infrastructure, it requires a notable initial investment in order to be useful.

While the above identification schemes are also applicable, we propose using item coding of the format ID@URI (Främling, 2002; Huvio et al., 2002; Kärkkäinen et al., 2003b), where the URI is an Internet address of the server where the product agent is

located and the ID part is a product identity that is unique inside that server. 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. This makes the allocation of the codes simple and, as current product codes can be used, the coding is potentially very easy to integrate to current information systems. Existing standards, such as the Global Trade Item Numbers (GTIN) (EAN International, 2001) can also be used for the ID part, as well as EPC numbers. The Global Location Number (GLN) (UCC, 2002) is also globally unique and provides a standard means to identify legal entities, trading parties and locations. GLN cannot directly be used as a reference to a product agent, but it is useful for identifying physical locations in tracking applications, for instance.

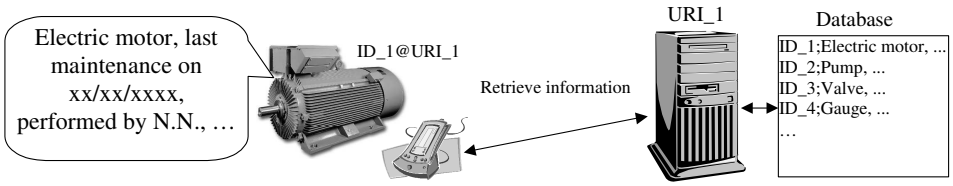

Figure 1. Accessing product information through ID@URI.

In agent-based solutions utilising the ID@URI notation, the product information can be made available everywhere where Internet access is available, without developing additional information registries. The URI part of the product identity directly tells where to find the information, while the ID part tells what physical product item the information is asked for (illustrated in Figure 1). Therefore ID@URI is a "label" in the sense given in (Jansen-Vullers et al., 2004), i.e. it allows retrieving the original data aggregated at a traceability decoupling point. The software component at the given URI can therefore act as the product agent of the particular product item and maintain the product data and links towards other sources of information on the product item. Therefore, the link between the physical product item and the corresponding agent plays a critical role in the system.

3.3 Agent-based model for managing information of products

In the agent model, product related information is retrieved and/or updated using the product-specific reference and only when needed as in Figure 2. The retrieved

information depends on the party asking for the information and the specifications of the request. Therefore only data that is useful and for which access and usage rights exist for the inquirer is transmitted. It is important to note that some data are related to groups of similar physical products (such as user instructions for products of the same model), while other data are specific to an individual physical unit (such as maintenance records). Both of these can be handled efficiently with the agent model, and in particular, a piece of information that is common for several products is easier to keep up-to-date as it needs to be stored in only one place.

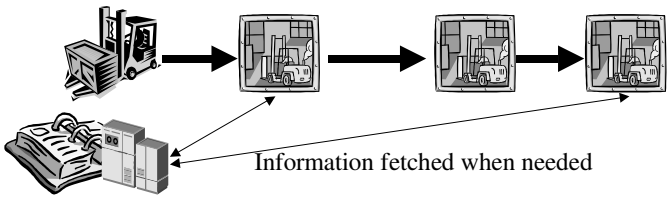

Figure 2. The "agent model" for real-time access to product information.

In product information management, information access can be split into two main functions:

1. *Accessing product data*, i.e. the ability to read and review specific information regarding a product. Examples of product data that need to be accessed are user instructions, maintenance records, assembly instructions etc.
2. *Updating product data*, i.e. the ability to append or amend the information regarding a product. Typical updates concern maintenance records, status monitoring of machines etc.

The division of these functions is important, as they demand different functionalities and often also different security settings. These functions can be handled by traditional means in a single-company setting but become challenging in a multi-company setting (Luckham, 2002). With the agent model, there is no difference whether it is applied to a single- or multi-company setting. All information requests for a given physical product item are performed with equal methods over the Internet and the information availability depends on the security class of the information and the authorisation codes of the inquirer. Each product's agent can manage the access rights as required.

3.4 Managing Information of Complex Products

Most sophisticated products, especially industrial equipment, are constructed of 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 constructed product forms a tangled hierarchy, the bill-of-materials, in which a product individual consists of a set of other product individuals. Such relations are handled by the concept of *composite products*[1] (Aerts et al., 2002; Främling, 2002), where the constructed product is at the top of a product containment hierarchy as illustrated in Figure 3. The principles of managing composite products in agent-based information management approaches are presented in this section.

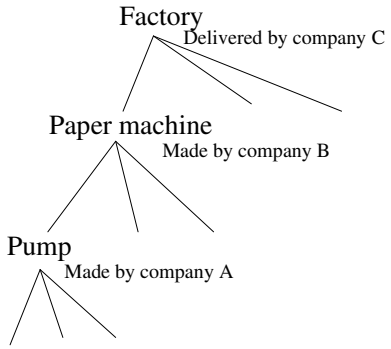

Figure 3. Example of composite product hierarchy.

When there is a need for product information management during the product's life cycle, it is essential to identify who has the responsibility for maintaining the life cycle data. Most likely, it is the brand-owner of the product (e.g. an original-equipment-manufacturer (OEM)) that has the main interest for information management as a part of its after-sales services. In most technically advanced products, different parts of the product come from different manufacturers, i.e. the product is a composite product. The providers of the various sub-parts should agree with the brand-owner on who has the responsibility of creating and managing the

[1] It could be argued whether "composite" or "aggregate" is the appropriate term (see for instance Fowler, 1997, p. 80 for a discussion on this). Destroying the "whole" of an aggregate does not destroy the parts, while parts are destroyed along with the "whole" for a composite. However, "composite" seems to be more commonly used in this context. The corresponding Design Pattern used in software engineering for this kind of object relations is also called "composite" (Gamma et al., 1995).

information of each part and how the information is disseminated. The company who has the responsibility for a part can then define the identity of the part and link it to his product information system. This procedure is illustrated in Figure 4.

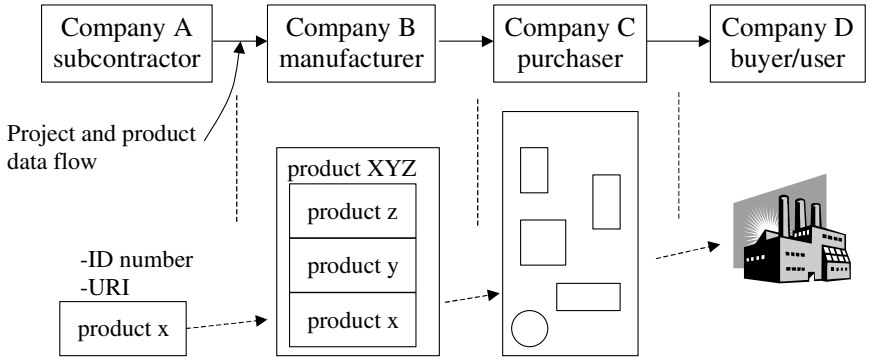

Figure 4. Example of how a composite product is created along the supply chain.

The notion of composite products could also be useful for tracking and tracing as defined in (van Dorp, 2004), where *backward traceability* determines the composition of an item and *forward traceability* determines all end products having consumed a component of particular interest. Keeping a link to the product information of all parts of an assembly is important through the whole lifecycle of the product, i.e. 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. Containment hierarchies should be built 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 useful when changing parts of the composite product or when breaking it up completely. Updating the information structure can be done implicitly or explicitly. An example of the implicit update can be found in the transportation context: reading the identifier of a part that is not at the top of the containment hierarchy can be interpreted to indicate that the part has been taken out of the transportation unit higher up in the containment hierarchy. In contrast, in a maintenance context, changing a part into a product will normally require doing an explicit update of the containment hierarchy.

4. The Dialog system

In this section, we present the software implementation that we developed for testing the concepts described in the previous section, called Dialog. In the first part, we present how product information can be accessed and updated in Dialog. The second part briefly explains implementation and installation issues, while the final part focuses on security issues. Even though messages for agent communication are presented in detail here, we do not have the intention to provide a formal software specification for messages in this article. The message formats presented here correspond to the current Dialog implementation, which is mainly used for research and piloting purposes. Therefore the messages as well as their contents are still subject to change. The Unified Modeling Language (UML) (Fowler, 1997) is used for illustrating the system structure by class diagrams while sequence diagrams illustrate agent interaction.

4.1 Agent communication

In this section we first present how information is accessed and updated in the Dialog platform: through direct information access, and accessing information through links. Then we proceed to present the currently implemented messages, and how they are used to provide the functionalities of composite products.

4.1.1 Access to product information

In the Dialog system, product information is accessible through methods in the product agent interface (Figure 5). The methods update() and getProductInformation() are used to append and retrieve information, respectively, while getCompositeInformation() relates to managing product and component hierarchies.

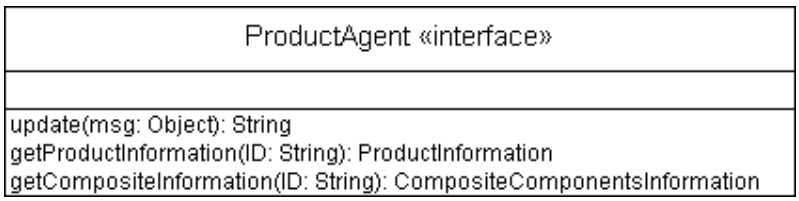

Figure 5. Product agent interface.

Information about a product item can be obtained and updated in two ways:

- *Direct information*: product information is sent directly as text, Hypertext Markup Language (HTML), eXtended Markup Language (XML) or some other application-dependent format.
- *Link to information*: URI where the information can be accessed. This would normally be the address of an existing web page or service, which can also be used for updating the information.

The two different ways were designed to accommodate various uses of the information. Direct information is useful in item-level applications, such as retrieving the expiry date of a specific item or retrieving the delivery address of a shipment. Information provision via a link is usually more effective for information that is shared by several physical units, whereas it would be wasteful to create a separate web page for every product item if the product agent can retrieve it directly from a database. When product information is provided by a link to an Internet address, it is possible to use existing web pages and web-based user interfaces also in other organisations. The advantage of this approach comes not only from using existing infrastructure but also from the fact that authentication and other security aspects can be handled by each organisation separately, if deemed necessary.

4.1.2 Agent messaging

Information exchange between agents is performed by *messages* (Monson-Haefel and Chappell, 2001). All current messages are described in Figure 6. The basic messages *InformationUpdate* and *IdentifierLinkMsg* have been used in two industrial pilots on item tracking (ISI Industry Software, 2003; Kärkkäinen et al., 2004a)[2].

The *InformationUpdate* message is used for direct information updates. It contains the product identifier (ID@URI), a timestamp and a location. It also contains a free-format data field for updating information about the product item. Free-format data

[2] The current implementation ensures message persistence, i.e. that messages do not get lost even though the receiver is not directly available. Messages are buffered until successfully received by the receiver's *update()* method. This is essential in applications like tracking and tracing.

here signifies that it can be text, HTML text, an XML document containing links to other documents or even binary data.

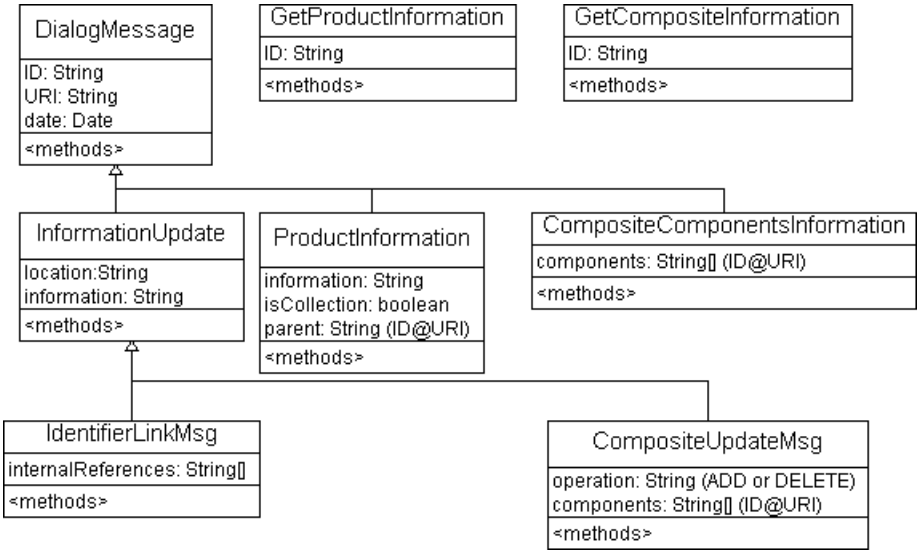

Figure 6. Class hierarchy of message objects. Methods are omitted because they are not a part of the actual messages.

IdentifierLinkMsg is used to link the ID part in ID@URI to existing reference numbers used in the owning company's business applications. The ID may have to be linked to several company internal references (e.g. project number, part number, order number, …) in order to be useful. This would typically be done when the ID is created for the physical product item. The IdentifierLinkMsg contains the same information as a InformationUpdate and a list of company internal reference numbers that the ID should be associated with.

Access to product information is requested by a *GetProductInformation* message and transmitted by a *ProductInformation* message. *ProductInformation* returns the product information either directly as free-format data (text or HTML if it is intended for visualisation) 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 itself is a part of a composite product, then the ID@URI of the "parent" product is returned.

The messages *GetCompositeInformation*, *CompositeComponentsInformation* and *CompositeUpdateMsg* in Figure 6 deal with composite products. Figure 7 illustrates

how composite products are handled in the Dialog platform with ID@URI identities. The whole containment hierarchy (bill-of-material) of a composite product can be managed through ID@URI identifiers, which greatly reduces the need to modify existing business applications or other information systems.

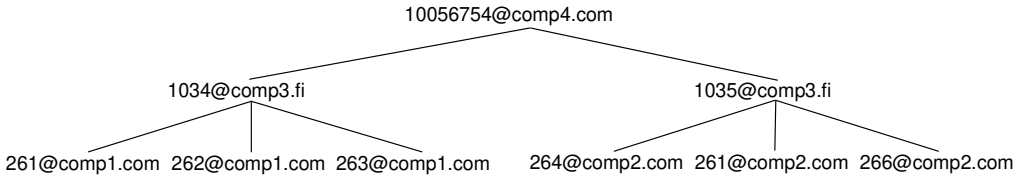

Figure 7. How composite products are handled in the Dialog system.

Figure 8 illustrates how an information update, e.g. location update, is propagated through the containment hierarchy of the composite product in Figure 7. In many instances when making product information updates, it is most convenient to use only the identity of the "outermost" part for the information request or update. There is a strong analogy to a shipment tracking application (note, that the containment hierarchy of transport packages is fully equivalent to bill-of-materials of products from a systems perspective). It may be difficult to obtain the identities of products that are inside a transport container to update their locations but if the container has been built as a composite product, the outermost container's agent has sufficient information to access the products' agents inside it. This makes it easy to propagate the location update to all the parts of the containment hierarchy, even though the different parts might have different "owners".

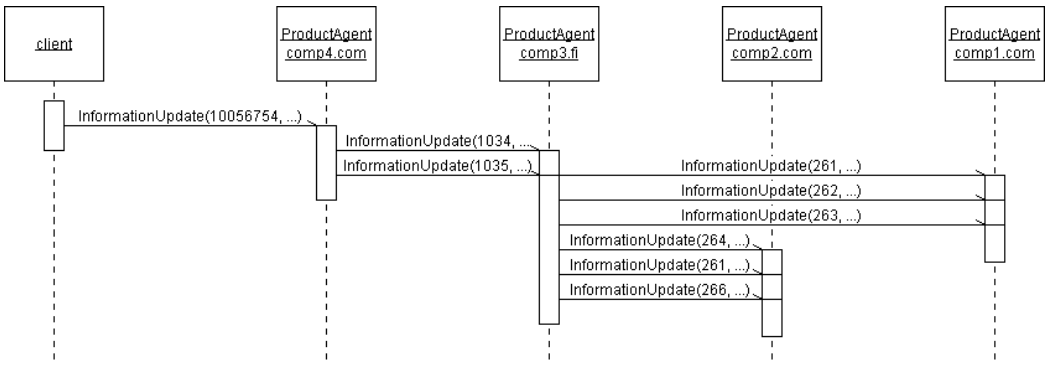

Figure 8. Propagation of information updates through the composite hierarchy by *InformationUpdate* messages.

In the Dialog platform, composite product hierarchies can be created with a *CompositeUpdateMsg*-message with the operation to perform as a parameter, i.e. add

or remove components. In addition to the information included in *InformationUpdate* messages, *CompositeUpdateMsg* contains a list of ID@URI identifiers to add to the containment hierarchy of the product or to remove from it, depending on the *operation* parameter.

The *GetCompositeInformation* message makes it possible to browse downwards through the containment hierarchy of composite products as shown in Figure 9. This information could also have been included in the *ProductInformation* message but that would cause transfer of useless data in cases where containment information is not interesting for the client.

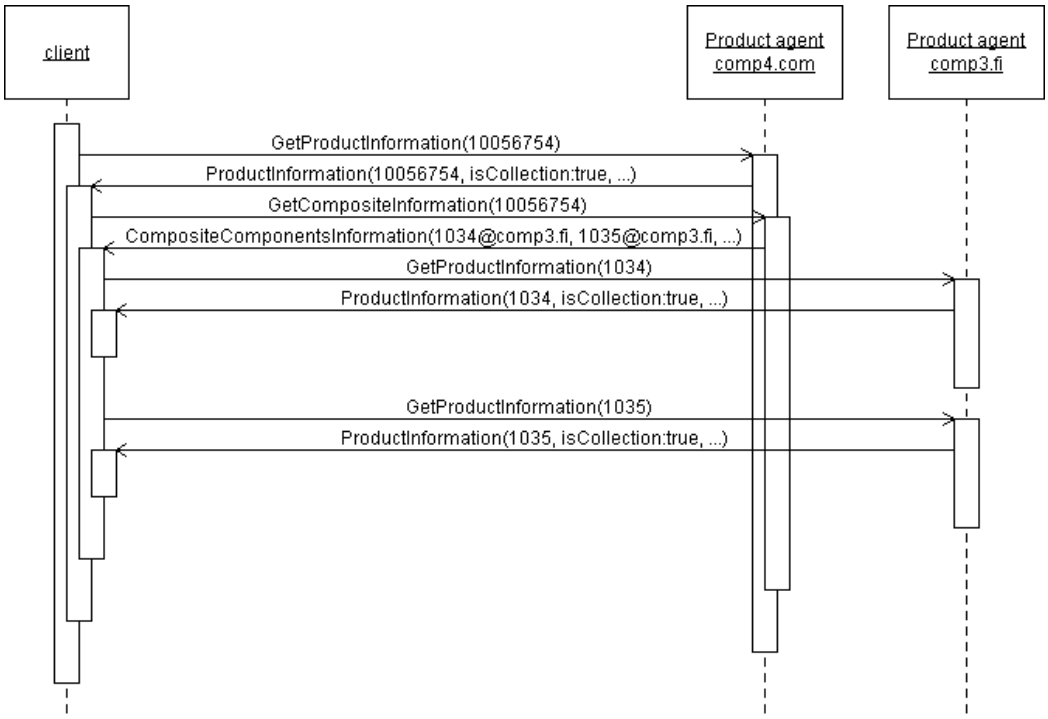

Figure 9. Fetching product information for the composite product in Figure 7 (only first two levels shown here for clarity).

4.2 Implementation

The functionalities presented in this paper are implemented with 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 for installing the components. The software components and their installation instructions are available at the Dialog web site "dialog.hut.fi". An Open Source community developing the Dialog platform also operates on the web site.

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 it 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 business applications.

Messages for communication between the software components can be implemented either as objects or as method calls. The current product agent implementation (Figure 5) is a mixture where the messages *GetProductInformation* and *GetCompositeInformation* have a corresponding method, while all other messages are sent as objects to the *update()* method. It is usually preferable to implement messages as objects than as methods because it simplifies the adding or modification of messages. However, the choice is also a compromise between other factors such as ease of implementation, performance etc.

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 also automatically create all needed database tables (currently four). Product location updates and support for composite products is immediately operational when starting the server component. Access to product information is operational as soon as the information has been inserted into the database. 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. Altogether, installation and set-up of both client and server can be done in less than five minutes.

The URI part of the product identifier can also contain a protocol part. The current implementation 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. Supporting different protocols is currently necessary because it seems unlikely that a single protocol could be defined and accepted for all application areas in the near future. We will try to propose such standards ourselves and take into use such standards as soon as they emerge.

4.3 Security issues

The security considerations associated with product information depend largely on the application area. For instance, information such as user instructions of a product may well be available without any validation of the product's identity or the identity of the person asking for the information. More restrictive authentication mechanisms are needed when updating product information in the system. Update of the product's maintenance records is an example of a situation where both the identity of the physical item and the person doing the update should be validated. Due to these differences, we have selected to keep the implementation as open as possible instead of imposing a specific security model. Therefore application-specific security protocols can be used without creating a need to modify the Dialog system itself.

Several technologies for implementing the desired level of security exist and can be used in our approach. It is, for example, possible to encrypt the data being transmitted using standard Secure Sockets Layer (SSL) connections (Netscape, 1996). However, obtaining a high level of security also means managing encryption keys, Public Key Infrastructure (PKI) certificates and other security-related information (Biennier, 2003).

A minimal validation of the identity of a physical item is possible by checking that the indicated URI owner has indeed issued the given ID. 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. Especially the increasing use of RFID (Radio Frequency Identification) tags (Kärkkäinen et al., 2004b) could facilitate the implementation of such validation techniques.

Validating the identity of the party asking for or updating information can also be done using two-key validation, possibly combined with a PKI based solution. Another

approach is to list and register the identities of the reading devices, which are authorized to access product information.

5. 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. An object-oriented program is essentially built up by object references and communicating objects and the ID@URI identifier and communicating agents uses the same approach in a multi-organisational context.

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 proved its functionality in two industrial pilots (ISI Industry Software, 2003; Kärkkäinen et al., 2004a), in which it was used for tracking international deliveries. Still, the choice of distributed architecture has some problems associated when one wants to ensure access, usage, availability and integrity over time. This is one of the main reasons for identifying parallels with object-oriented programming, which addresses many of these questions. The composite model proposed here corresponds to a *Design Pattern*, i.e. a well-known reference solution in object-oriented software engineering (Gamma et al., 1995). Other design patterns could be more applicable in other contexts than discrete products, e.g. tracking and tracing in the context of multi-echelon food supply chains (van Dorp, 2004; Jansen-Vullers et al., 2004). Food is not an aggregate of discrete parts, so it is impossible to attach information to physical subassemblies.

The proposed system is open. This means that solutions based on the methods presented here can be built without need to pay licence fees or offend any patents known by the authors. Furthermore, the concepts presented here do not require the involvement of any third-party actors in order to be usable. It is therefore easy for companies of all sizes to start using these concepts.

In practice, effective methods for data communication between organisations are needed not only due to the potential economical benefits but also due to changes in legislation concerning traceability of raw materials and product lifecycle management in general. The model presented here addresses those needs. The biggest challenge on the way might be to achieve a shift in the general viewpoint on service provision and information management.

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, 311-327.

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

Beulens, A.J.M, Jansen, M.H., Wortmann, J.C., 1999, The information de-coupling point, in: Global Production Management, IFIP WG5.7, Int. Conf. on Advances in Production Management Systems (Kluwer Academic Publishers, Boston) 51-58.

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) 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, 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, 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, 139-152.

van Dorp, K.J., 2002, Tracking and tracing: a structure for development and contemporary practices, Logistics Information Management, Vol. 15, No. 1, 24-33.

van Dorp, C.A., 2004, Reference-data modelling for tracking and tracing, PhD thesis, Wageningen University, Netherlands.

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/

Fowler, M., 1997, UML Distilled (Addison-Wesley, Reading, Massachusetts).

Fox, M.S., Barbuceanu, M., Teigen, R., 2000, Agent-Oriented Supply-Chain Management, International Journal of Flexible Manufacturing Systems, Vol. 12, 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 (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 (Tieke: Finnish Information Society Development Centre, Finland).

Främling, K., Kärkkäinen, M., Ala-Risku, T., Holmström, J., 2004, Managing Product Information in Supplier Networks by Object Oriented Programming Concepts, in:

Taisch, M., Filos, E., Garello, P., Lewis, K., Montorio, M., eds., Proceedings of IMS International Forum, Cernobbio, Italy, 17-19 May 2004, 1424-1431.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995, Design Patterns: elements of reusable object-oriented software (Addison-Wesley, Reading, Massachusetts).

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) 189-204.

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

Hamilton, J., 2001, The European Union's consumer guarantees directive, Journal of Public Policy & Marketing, Vol. 20, No. 2, 289-296.

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.

Holmström, J., Främling, K., Tuomi, J., Kärkkäinen, M., Ala-Risku, T., 2002, Implementing Collaboration Process Networks, International Journal of Logistics Management, Vol. 13, No. 2, 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, 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, 29-43.

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

Jansen-Vullers, M. H., Wortmann, J. C., Beulens, A. J. M., 2004, Application of labels to trace material flows in multi-echelon supply chains, Production Planning & Control, Vol. 15, No. 3, 303-312.

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) 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, 86-101.

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

Kauremaa, J., Auramo, J., Tanskanen, K., Kärkkäinen, M., 2004, The use of information technology in supply chains: transactions and information sharing perspective, in: Logistics Research Network Annual Conference, Dublin, Ireland, September 9–10, 2004, available online (22nd October, 2004): http://www.tuta.hut.fi/logistics/publications/The_use_of_IT_in_supply_chains.pdf

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, 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) 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, 147-159.

Kärkkäinen, M., Ala-Risku, T., 2003, Facilitating the integration of SME's to supply networks with lean IT solutions, in: eChallenges e-2003 conference proceedings, 22-24 October, Bologna, Italy.

Kärkkäinen, M., Ala-Risku, T., Främling, K, 2004a, Efficient Tracking for Short-Term Multi-Company Networks, International Journal of Physical Distribution and Logistics Management, Vol. 34, No. 7, 545-564.

Kärkkäinen, M., Holmström, J., Främling, K., 2004b, Wireless item identification: a solution for e-commerce fulfilment problems, International Journal of Electronic Business, Vol. 2, No. 1, 108-120.

Langer, G., Alting, L., 2000, An Architecture for Agile Shop Floor Control Systems, Journal of Manufacturing Systems, Vol. 19, No. 4, 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).

McFarlane, D., Sarma, S., Chirn, J.-L., Wong C.Y., and Ashton, K., 2002, The Intelligent Product in Manufacturing Control and Management, in: Proceedings of IFAC World Congress, Barcelona.

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

Netscape, 1996, SSL 3.0 Specification, available online (December 2nd, 2003): http://wp.netscape.com/eng/ssl3/index.html

NIST, 2002, Digital Signature Standard (DSS) and Secure Hash Standard (SHS), (National Institute of Standards and Technology), 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).

Oat Systems & MIT Auto-ID Center, 2002, The Object Name Service, 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).

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) 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) 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, 227-232.

Töyrylä, I., 1999, Realising the potential of traceability – A case study research on usage and impacts of product traceability, Finnish Academy of Technology, Espoo.

UCC, 2002, GLN Implementation Guide, Uniform Code Council inc., available online (October 22nd, 2004): http://www.uc-council.org/ean_ucc_system/pdf/GLN.pdf

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

Wong, C.Y., McFarlane, D., Zaharudin, A.A., Agrawal, V., 2002, Intelligent Product Driven Supply Chain, in: Proceedings of IEEE SMC 2002, Tunisia.

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