Aggregation of product information for composite intelligent products

William Derigent

9^{th} *International Conference of Modeling, Optimization and Simulation - MOSIM'12*
June 6-8, 2012 - Bordeaux - France
"Performance, interoperability and safety for sustainable development"

AGGREGATION OF PRODUCT INFORMATION FOR COMPOSITE INTELLIGENT PRODUCTS

William DERIGENT

CRAN / University of Lorraine
Campus sciences, BP 70239
54506 Vandoeuvre CEDEX
william.derigent@cran.uhp-nancy.fr

ABSTRACT: *New challenges and opportunities arise with concepts such as Internet of Things (IoT), Ubiquitous/Pervasive Computing. Through these concepts, objects of the real world are linked with the virtual world. Miniature electronic devices (i.e RFID tags) have made intelligent products (i.e products conveying at least their own information) possible. As a result, in the near future, each product might have data embedded in. As as result, intelligent containers,i.e complex products composed of intelligent items, might contain huge amounts of information divided in several pieces, each one located on a different component of intelligent container. Agregating this information to obtain a higher level of information is still a challenge that this article tries to address. The result is a simple product information aggregation scheme based on some specific data transformation rules and data modelling, implemented with XML technologies.*

KEYWORDS: *intelligent products, product information, data aggregation, XML*

1 INTRODUCTION

New challenges and opportunities arise with concepts such as Internet of Things (IoT) and Ubiquitous/Pervasive Computing (Weiser, 1993). Through these concepts, objects of the real world are linked with the virtual world. Thus, connections are not just people to people or people to computers, but people to things and most strikingly, things to things (Sundmaeker et al., 2010).Such applications rely on ever more complex information systems combined with ever increasing data volumes, which are stored in a large number of information vectors. These vectors may be either fixed (desktop computers) or mobile (wireless devices, RFID...). The IoT based on the RFID usage enables accessing information disseminated on any kind of physical object and developing new smart services.This opens up an entirely new range of networked applications such as machine-to-machine and context-aware communications. As devices became connected, they began to share capabilities over the network by introducing seamless access to remote information resources and communication with fault tolerance, high availability, and security (Saha and Mukherjee, 2003). The simple mechanisms for linking resources have provided a mean for integrating distributed information bases into a single structure.

Although widely explored in the Computer Science field, IoT applications in the framework of SCM or Material Flow Management are limited. However, such a concept may turn out to be a good strategy. In fact, different types of interoperability problems occur between companies all along the supply chain, at different levels of management (organisational, semantic, syntactic and technical level). To help solving these problems, works initiated since 2003 by the PDS (Product-Driven Systems) community advocate to allocate more information to manufactured products. In this case, a product carries physically a part, or even the totality of the information needed for managing its own evolution,thus building an interoperability link between the different actors involved in the product lifecycle. Meyer concurs with the PDS community by stressing the fact that, in an increasingly interconnected and interdependent world involving many actors, supply chain information should not be stored in a single database but should be distributed throughout the supply chain network. Substantial information distribution improves data accessibility and availability compared to centralized architectures, but they are more complicated to design because data consistency and integrity has to be maintained. In short, product information may be allocated both within fixed databases and/or within the product itself, thus leading to products with informational and/or decisional abilities, referred as "intelligent products".

(Meyer et al., 2009) provide an in-depth study of the literature related to intelligent products and propose a classification of research work according to three orthogonal dimensions which are the *level of intelligence*, the *location of intelligence* and the *aggregation level of intelligence* (see figure 1). According to McFarlane et al., an Intelligent Product has the following properties:

1. Possesses a unique identification,
2. Is capable of communicating effectively with its environment,
3. Can retain or store data about itself,
4. Deploys a language to display its features, production requirements,
5. Is capable of participating in or making decisions relevant to its own destiny.

Based on this definition, Wong et al. (Wong et al., 2002) adopts a two-level classification of intelligence. When the Intelligent Product only covers points 1 to 3, it is information oriented, and is called a product with level 1 product intelligence. A product with level 2 product intelligence covers all points, and is called decision oriented.

Morevoer, an intelligent product has its own intelligence, but not necessarily located at the product itself. Indeed, the intelligence may be *through network* (in this case, the intelligence is completely outside the physical product) or *at object*, meaning all the intelligence is hosted at the product. Intermediate solutions are also possible.

The third dimension,*aggregation level of intelligence*, is considered by Meyer et al. as an important dimension, as many products are composed from parts, which can also be products in itself. Some parts may be products with sufficient information processing and communication capabilities to embed decision-making algorithm, whereas others can only have a unique identifier.Thus, two types of intelligent products can coexist: *intelligent item* which manages information about itself but not its component information and *intelligent container*, aware of the components that it is made of e.g an intelligent shelf, which can notify its owner when a specific product is out of stock.

Managing this aggregation is an interesting topic. As stated by (Främling et al., 2006), challenges associated with information management could arise when considering complex products with complicated bills-of-material. Indeed, most sophisticated products, especially industrial equipment, are constructed of parts that come from many different companies. This

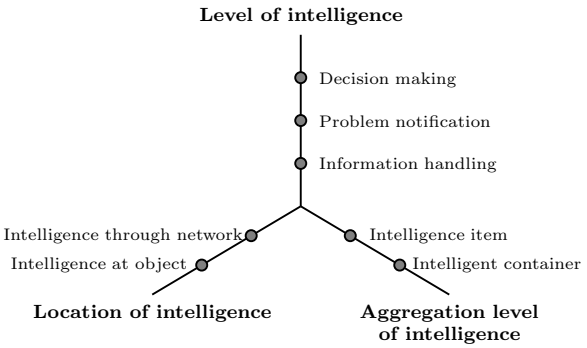

Figure 1: Classification model of Intelligent Products

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 bills-of-materials, in which a product individual consists of a set of other product individuals. For example, a set of intelligent items regrouped in a package could be associated as a single complex product. Consider that each intelligent item stores its own properties, like its volume or its weight. To determine the global volume (i.e the volume of the package), it is then necessary to get the volume of every intelligent items and to aggregate all these data in one. Of course, more complex situations could arise, where aggregation/desaggregation tools might be needed.

In this article, we propose a simple aggregation mechanism for physical values (weight, color, price,volume,...), based on the combined use of the concepts of extensivity/intensivity and automatic transformation capabilities of the XSLT technology.In the next section, previous works dealing with data aggregation and intelligent products are detailed. In the third section, we propose an aggregation process dedicated to products. the concepts of extensivity and intensivity are presented, followed by a short description of the XLST technology and its capabilities in term of data transformation. The fourth section introduces the XML Schema and XSLT model needed to perform the data transformation and, finally, the fifth section presents a simple theoritical case of study, on which a simple data transformation is perfomed.

2 PRODUCT INFORMATION AND INTELLIGENT PRODUCTS

2.1 Handling Product Information

Intelligent products enable distributing vast amounts of product and component information forward all along the product lifecycle (including the supply chain) as seen in figure 2. Many actors are involved in the product lifecycle, from designer to recyclers, and all generate product information.

Figure 2: Actors involved in the various stages of a product's life cycle

As stated by (Marsh and Finch, 1998), product information storage can be centralized or localized(or decentralized). The localized solution requires to store information on the product itself. In the centralized model, product information is transmitted and collected to one place. Both architectures have drawbacks or interests but, most of the time, product information management is done while using a centralized model (Främling et al., 2006; PROMISE, 2009), where the product only owns an unique identification (as EPC Code (Brock, 2001) or ID@URI (Kärkkäinen, 2003)). Given this unique identification, product information could be requested from a distant database. This type of product information management is interesting since it eases to perform data analysis on entire product populations and detect general rules that can be used to for improving product design and perform predictive maintenance for instance (Främling and Rabe, 2006). This mode of management gave rise to three well known industrial and academic information management systems architectures based on an object centric paradigm for managing product information at the individual product instance level (Ranasinghe et al., 2011):

- EPC network architecture with its standard interfaces for collecting and accessing product related data (Ranasinghe et al., 2007; Global, 2009),
- The approach taken by the DIALOG information system using its ID@URI approach and further developed within the PROMISE project (Kärkkaïnen et al., 2003; DIALOG, 2009),
- World Wide Article Information (WWAI) approach using a peer-to-peer (P2P) lookup method to access and store data in backend systems.

However, as said previously, it should be not appropriate to swap all data to external databases because, to be used, these systems require efficient maintenance support and communication networks.(Seitz et al., 2010; Gouyon and David, 2008; Kubler et al., 2010) advocate to store product information physically on the product via embedded electronic devices (sensor networks,high-capacity RFID tags, communicating material,..) when the product information must be accessed at anytime, which cannot be garanteed if the data is stored on the network, because of non-permanent communication possibilities. Moreover, these systems require a stable economical,social and technical environment to function, and this is less envisageable for products with a long product lifecycle (like cars or industrial systems).

As underlined by (Simon et al., 2001), product information could be separated into static information and dynamic information. Static product information refers to the specification of the product type as manufactured, e.g materials, components, used suppliers, configuration and options, servicing, instructions and end-of-life informations. Dynamic product information comprises information that is collected during the use of a product. This includes patterns of use, environmental conditions, servicing actions,and part replacements.

2.2 Managing Product Information Aggregation

For complex products, information overflow in the downstream lifecycle can happen, when the amount and sophistication of product components increases(Främling et al., 2006). Beulens (Jansen-Vullers et al., 2004) envisage to create some information decoupling points all along the supply chain, arguing that all information should not be sent forward in the supply chain. 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 points. At a traceability decoupling point, data originated from several products 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.

To comply with the information decoupling point principle, EPC and DIALOG architectures propose a "product centric information management" in which information regarding a product is retrieved over in-

formation networks when needed using unique product identities as references. DIALOG, for example, uses the ID@URI notion to identify each product instance. The URI (unique resource identifier) directly indicates the adress of the database server where the information is located, while the ID part tells what physical product item the information is asked for. Therefore, ID@URI is a "label" because it allows retrieving the original data aggregated at a traceability point, and only when requested by the user.

In the field of localized solutions, Seitz (Seitz et al., 2010; Seitz et al., 2011) propose a rule-based mechanism to handle sensor data aggregation with sensors attached to a single product. However, to the best of our knowledge, there is no localized solutions encompassing an aggregation mechanism for complex products.

3 EMBEDDED PRODUCT DATABASE AND INFORMATION AGGREGATION

As said previously, highly relevant product information should be stored on directly on the product, whereas other information could be stored on external databases and accessible via a unique identifier (ID@URI or else). In fact, each product items should contain its own information and a mean to link this information with other products.

Aggregation is a need to get over information overflow which might occur for complex products. However, there is currently no aggregation models that support aggregation when product information is located on the product. To fill this gap, an aggregation scheme for this type products is proposed and detailed. This present aggregation scheme is based on the notion of information decoupling point, as introduced in the beginning of the paper. We suppose that, all along its lifecycle, the product is processed through multiple decoupling points. Moreover, in our vision, each product instance can store physically information thanks to embedded electronic devices (e.g RFID tags,sensors,..). We also suppose this information is always accessible, at least at an information decoupling point.

While being processed at a decoupling point, all the product items (including the root product itself) communicate their information bit to a central "gathering" device which role is to centralized all the product information into a single file. Then this file is processed via data transformation process. At the end, a file containing aggregated product information is obtained. This file must correspond to the needs of the application requesting information. In fact, at each information decoupling point, product information is organised,selected, cleaned,... by an "on-the-fly" aggregation to form an appropriate view of the product which is sent to the user. Moreover, no information is suppressed neither on the root product or on its items. The figure 3 depicts the aggregation process.

As explained before, there are numerous actors and each one with a specific point of view of the product. To extract agregated information from the product embedded database, a user-defined transformation process is needed, which can select and reorganize data in a suitable form for the actor. In the next section, we propose a generic product data model and a simple data transformation that would provide some basic aggregation services to the user very easily. The concerned services are the automatic product hierarchy (bills-of-materials) generation and the automatic grouping of common item attributes like, for example, the type of material composing the product,the mass of these components,.... There is obviously an infinite number of possible aggregation schemes, and the one proposed in the next section is not exhaustive. However, it should be appropriate in a lot of situations. Moreover, its implementation is easy thanks to well-known and easy-to-learn technologies.

Keep in mind it is assumed that the different actors share a unique data exchange format with standard syntax and semantics. Indeed, the cross-domain nature of the information is a key problem of product information management. In fact, all along the product lifecycle, new relevant information must be added to the product database from various stakeholders while maintaining the database consistency. Integrating data from heterogeneous sources is a very though exercice not discussed here. Yet some research initiatives have proposed standards potentially able to help data exchange all along the product lifecycle (Tursi et al., 2009).

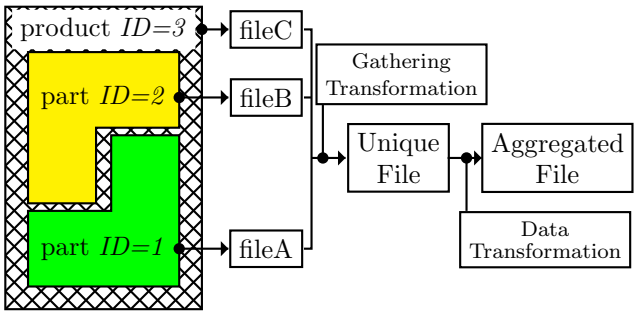

Figure 3: Aggregation process

4 DATA AGGREGATION

In this section, we detail the data transformation process, base on XML technology.A brief description of some XML technologies (XML Schema, XSLT) is first given since it is needed to understand the data transformation. This description tries to provide a comprehensive insight of XML. However, it is difficult to

resume correctly these techniques in such small space, and readers are advised to consult the references dedicated to XML given in this paper. Some of them are free electronic documents, available online, so that readers could get them easily.

4.1 XML technology

XML (Extensible Markup Language) (Morrison, 2005) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined in the XML Specification produced by the W3C, and several other related specifications, all free open standards. An XML file is written using tags as is the case for HTML tags. However, unlike HTML, tags are user-defined. the figure 4 is issued from an example provided by the W3C Corporation [1] and shows a very simple example of a XML file.

```
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Do not forget me this weekend!</body>
</note>
```

Figure 4: Simple example of an XML file

The note above is quite self descriptive. It has sender and receiver information, it also has a heading and a message body. Basically, a XML file is just information wrapped in tags. The XML language has no predefined tags and the tags in the example above (like `<to>` and `<from>`) are not defined in any XML standard. These tags are defined by the author of the XML document. A XML Document is composed of XML elements, surrounded by two tags called start and end tags. For the XML element `note`, the start tag is written `<note>`, the end tag is written `</note>` and everything between is considered as being a part of the XML element. Moreover, an element can contain attributes, text content and other elements. In our example, the element `note` is composed of the four sub-elements `to`,`from`,`heading` and `body`. As a result, a XML document forms a tree structure as depicted in figure 5. The tree starts at the root and branches to the lowest level of the tree.

XSLT (Extensible Stylesheet Language Transformations) is a declarative, XML-based language used for the transformation of XML documents. The original document is not changed; rather, a new document is created based on the content of an existing one. The new document may be serialized (output) by the processor in standard XML syntax or in another format, such as HTML or plain text. XSLT is most often

[1] http://www.w3schools.com/xml/xml_whatis.asp

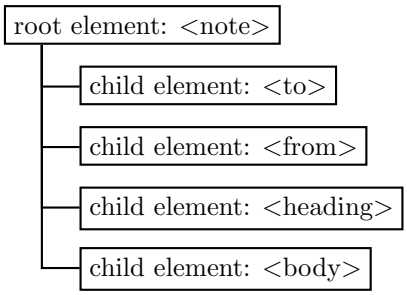

Figure 5: tree structure of our example

used to convert data between different XML schemas or to convert XML data into web pages or PDF documents. An XSLT file is composed of easy-to-write transformation rules denominated **templates**.A template is a transformation rule applied for a type of element. The example given figure 6 gives an example of data transformation using XML. The original XML file is composed of elements `person`,which childs are `name`,`age` and `weight` elements. In this file, two persons among twelve are described, John and Paul, each one with different age and weight. By using a XSLT file with appropriate rules, the final XML file is obtained. This one is diametrally different as it presents a summary of the previous file. The element **summary** is the root element and contains:

- the **number** element, which value is equal to the number of persons describe in the file,
- the **ppl** element, which value is an condensed version of the `person` element
- the **heavy** element, which value is equal to the heaviest weight in the file,
- the **light** element, which value is equal to the lightest weight in the file.

This file is thus an aggregated version of the previous one. Because all the complicated work has been done before, no difficult technological implementation is required. From our point of view, these technologies provide a framework perfectly suited for data aggregation. However, to be done, this process requires a appropriate data modelling (which structures the XML file) combined with aggregation rules (which compose the XSLT file). These two key points are detailed in the next subsection.

4.2 Modelling and rules used for data aggregation

In our vision, all important produt information is stored on the product, in each of its components and

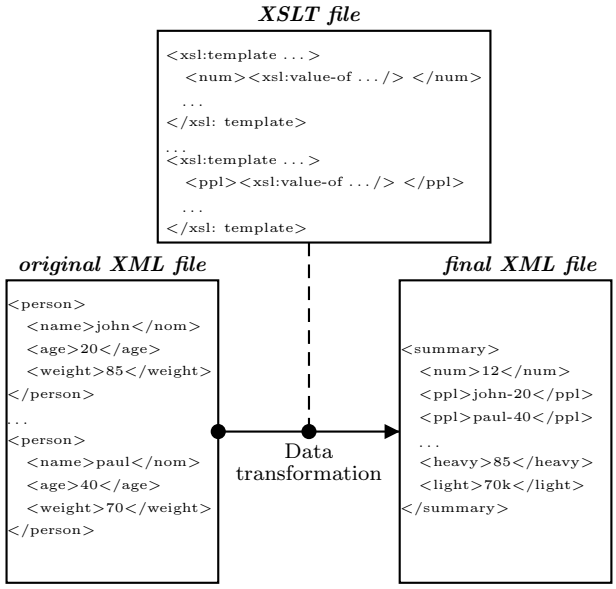

Figure 6: data transformation process

on itself. When reading the product, a lot of information is obtained. Each product sends its own information in an XML form, denoted Elementary Product Information (EPI), which are then grouped in a unique file (see figure 3) to be aggregated. Our aggregation process is divided into two steps:

- "on-the-fly"reconstruction of the complete product bill-of-materials, from the different EPIs obtained,
- creation of new product "aggregated" attributes.

4.2.1 First part:"on-the-fly" bill-of-materials

Each EPI is represented in the unique file by a `product` element.This element is composed, at least, of two elements which are:

- `id`, which stores the identification of the product item. As said previously, it could be an EPC Code, a ID@URI or anything else ensuring uniqueness,
- `parent`, which stores the identification of the parent product item. A product item has 0 to 1 parent (0 for is for the root product).

An EPI can have other elements. The figure 7 show simple EPI examples. On this figure, three EPIs obtained from three different product items are displayed. The first EPI describes a car identified as 449-541-415 with no parent product item(it is then the root product). The two others are a wheel identified as wheel45-12-13 and an engine identified as V12-74-24. They are direct children of 449-541-415. Both have additionals elements, describing their weights and materials

```
<product>
    <id>449-541-415</id>
    <parent>none</parent>
    <type>car</type>
</product>

<product>
    <id>wheel45-12-13</id>
    <parent>449-541-415</parent>
    <type>wheel</type>
    <weight>40</weight>
    <material>rubber</material>
</product>

<product>
    <id>V12-74-24</id>
    <parent>449-541-415</parent>
    <type>engine</type>
    <weight>120</weight>
    <material>aluminium</material>
</product>
```

Figure 7: simple EPI examples

The element `parent` is sufficient to recreate, thanks to XSLT rules, the complete hierarchy of products,with all their data (results shown figure 8). Please note that the dynamic nature of the product bill-of-materials is completely supported by this simple approach. For example, replacing a component by another one or adding components to the existing product will induce a new product hierarchy, automatically obtained if the previous EPI model has been implemented in the new components (figure 3).

4.2.2 Second part: creation of aggregated attributes

A final product is a group of components, each one having specific attributes. When performing aggregation, it should be interested to group these attributes into a number of global attributes caracterizing the product itself. In this procedure, attributes are grouped only and only if the need is expressed in the file. If it is required, attributes with the same name are grouped. Let a product be composed of two components C_1 and C_2,each one having `weight` attribute which values are respectively 20 and 40. If aggregation is demanded on this attribute, then a new `weight` attribute is created at the root product level with value equal to 60.

Two modes of grouping are defined, depending of the nature of the attribute concerned. Indeed, in this method, an attribute can have an extensive or in-

```
<product>
    <id>449-541-415</id>
    <parent>none</parent>
    <type>car</type>

    <product>
        <id>wheel45-12-13</id>
        <parent>449-541-415</parent>
        <type>wheel</type>
        <weight>40</weight>
        <material>rubber</material>
    </product>
    <product>
        <id>V12-74-24</id
        <parent>449-541-415</parent>
        <type>engine</type>
        <weight>120</weight>
        <material>aluminium</material>
    </product>
</product>
```

Figure 8: resulting Bill-of-Materials

```
<product>
    <id>449-541-415</id>
    <parent>none</parent>
    <type>car</type>
</product>

<product>
    <id>wheel45-12-13</id>
    <parent>449-541-415</parent>
    <type>wheel</type>
    <weight req="1" nat="ext">40
    </weight>
    <material req="1" nat="int">rubber
    </material>
</product>

<product>
    <id>V12-74-24</id>
    <parent>449-541-415</parent>
    <type>engine</type>
    <weight req="1" nat="ext">120
    </weight>
    <material req="1" nat="int">aluminium
    </material>
</product>
```

Figure 9: EPIs with `nature` and `required` attributes

tensive nature. Attributes with extensive nature (e.g volume,mass,...) are summed whereas attributes with intensive nature(e.g temperature, volumic mass, maintenance operation dates, ...) are grouped in lists.

To apply these two rules, some additions must be done on the previous EPI modelling. When specifying an element, two properties `req` (for "required") and `nat` (for "nature") need to be expressed. The `required` property can take two values, 0 or 1, and the `nature` property can take two values, "ext" or "int". With these modifications, the EPIs exposed in figure 7 are completed as depicted in figure 9. The two attributes appear in the definition of the `weight` and `material` elements. Both are required (`req`="1") and `weight` is declared as an extensive value whereas `material` is declared as intensive value. Indeed, weights can be added but not materials.

Expressing the previous rules using XSLT is easy and the resulting aggregation file is shown in figure 10. The product hierarchy is the same, but two new elements have appeared at the root level which are `weight` and `list-of-material`, which are respectively aggregations of the different `weight` and `material` elements extracted form the level below.

5 CONCLUSIONS AND PERSPECTIVES

The problem of product information management has been raised and a study of the different existing solutions has been done. There are currently three well-known architectures able to manage product information in an appropriate manner. However all of them are relying on a permanent web and database access, which could not be always the case (especially in EOL phase, when the product is aged and database server not maintained). Some important information should be stored on the product, thus conducting to a product embedded database. In this paper, we propose to model product information thanks to Elementary Product Information expressed in the form of XML fragments. The number of these fragments could be important and thus, to get high level data, an aggregation mechanism was needed at least for very basic services as getting the complete bill-of-materials. In a first attempt, we proposed an aggregation mechanism, based on a simple but generic data modelling and some aggregation rules. We demonstrated aggregating product embedded information was possible using XML technologies. Note that the dynamic nature of product information is taken into account by this methodology.

However, as stated in this article, the aggregation mechanism is fixed and perhaps too simple for all industrial cases. Moreover, the intensive and extensive nature of attribute , while working for simple data, might not be appropriate for complex one. As a result, users might develop their own rules to do aggregation. Nevertheless, the aggregation process as

```
<product>
    <id>449-541-415</id>
    <parent>none</parent>
    <type>car</type>
    <weight>160</weight>
    <list-of-material>
        rubber
        aluminium
    </list-of-material>

    <product>
        <id>wheel45-12-13</id>
        <parent>449-541-415</parent>
        <type>wheel</type>
        <weight ... /weight>
        <material ... /material>
    </product>
    <product>
        <id>V12-74-24</id
        <parent>449-541-415</parent>
        <type>engine</type>
        <weight>120</weight>
        <material>aluminium</material>
    </product>
</product>
```

Figure 10: new global attributes

well as the generic data model should not change and XML technologies could still be used, and this will tremendously simplify the implementation work.To conclude, this first attempt to develop an generic aggregation mechanism is stimulating and might be opening the path towards new promising solutions.

REFERENCES

Brock, D. (January 2001). The electronic product code (epc) - a naming scheme for physical objects,. *MIT Auto-ID Center White Paper*.

DIALOG (2009). Distributed information architectures for collaborative logistics. available from: /http://dialog.hut.fi , (accessed 14 november 2009). Technical report.

Främling, K., Ala-Risku, T., Kärkkäinen, M., and Holmström, J. (2006). Agent-based model for managing composite product information. *Computers in Industry*, 57(1):72 – 81.

Främling, K. and Rabe, L. (2006). Enriching product information during the product lifecycle. In *12th IFAC Symposium on Information Control Problems in Manufacturing*.

Global, E. (2009). Epc global architecture framework. available online at: http://www.epcglobalinc.org/standards/architecture (accessed 12 november 2009). Technical report.

Gouyon, D. and David, M. (2008). Implementing the concept of Product-Driven Control using Wireless Sensor Networks: some experiments and issues. *17th IFAC World Congress*, 1:5488–5493.

Jansen-Vullers, M., Wortmann, J., and Beulens, A. (2004). Application of labels to trace material flows in multi-echelon supply chains. *Production Planning & Control*, 15:303 – 312.

Kärkkaïnen, M., Ala-Risku, T., and Främling, K. (2003). The product centric approach: a solution to supply network information management problems? *Computers in Industry*, 52 (2):147 – 159.

Kärkkäinen, M. (2003). Increasing efficiency in the supply chain for short shelf life goods using rfid tagging. *International Journal of Retail & Distribution Management*, 31(10):529–536.

Kubler, S., Derigent, W., Thomas, A., and Rondeau, E. (2010). Problem definition methodology for the "Communicating Material" paradigm.

Marsh, L. and Finch, E. (1998). Using portable data files in facilities management. *Facilities*, 16(1-2):21 – 28.

Meyer, G. G., Främling, K., and Holmström, J. (2009). Intelligent products: A survey. *Computers in Industry*, 60(3):137 – 148. Intelligent Products.

Morrison, M. (2005). *Sams Teach Yourself XML in 24 Hours, Complete Starter Kit (3rd Edition) (Sams Teach Yourself in 24 Hours)*. Sams, Indianapolis, IN, USA.

PROMISE (2009). Product lifecycle management and information tracking using smart embedded systems. Technical report, Sixth Framework European project.

Ranasinghe, D., Harrison, M., and Cole, P. (2007). *Networked RFID systems and anti-counterfeiting: raising barriers to product authentication*, chapter EPC Network Architecture. Springer-Verlag.

Ranasinghe, D., Harrison, M., Främling, K., and McFarlane, D. (2011). Enabling through life product-instance management: Solutions and challenges. *Journal of Network and Computer Applications*.

Saha, D. and Mukherjee, A. (2003). Pervasive computing: a paradigm for the 21st century. *Computer*, pages 25–31.

Seitz, C., Lamparter, S., Schöler, T., and Pirker, M. (2010). Embedded rule-based reasoning for digital product memories. In *2010 AAAI Spring Symposium Series, Paolo Alto, California*.

Seitz, C., Runkler, T., and J.Neidig (2011). Autonomous product memories for industrial applications. In *9th IEEE International Conference on Industrial Informatics (INDIN)*.

Simon, M., Bee, G., Moore, P., Pu, J., and Xie, C. (2001). Modelling of the life cycle of products with data acquisition features. *Computers in Industry*, 45:111–222.

Sundmaeker, H., Guillemin, P., Friess, P., and Woelfflé, S. (2010). Cluster of european research projects on the internet of things (cerp-iot).

Tursi, A., Panetto, H., Morel, G., and Dassisti, M. (2009). Ontological approach for products-centric information system interoperability in networked manufacturing enterprises. *Annual Reviews in Control*, 33(2):238 – 245.

Weiser, M. (1993). Hot topics-ubiquitous computing. *Computer*, 26(10):71–72.

Wong, C. Y., Mcfarlane, D., Zaharudin, A. A., and Agarwal, V. (2002). The intelligent product driven supply chain. In *Proc. IEEE Int. Conf. on Systems, Man and Cybernetics*, pages 4–6.