Information architecture for Intelligent Products in the Internet of Things

Kary Främling [*]

Jan Nyman [**]

*) BIT Research Centre, Helsinki University of Technology, FI-02015, TKK, Finland
E-mail: Kary.Framling@tkk.fi, Tel: +358 50 5980451; Fax: +358 9 451 3665

**) BIT Research Centre, Helsinki University of Technology, FI-02015, TKK, Finland
E-mail: Jan.Nyman@tkk.fi, Tel: +358 9 451 6017; Fax: +358 9 451 3665

ABSTRACT

Purpose of this paper

The paper presents an information architecture for managing product information during the whole lifecycle about "dumb" products as well as products that have their own processing capabilities. Real-life implementations presented show the technical feasibility of the architecture and attempt to illustrate the potential commercial impact.

Design/methodology/approach

As defined in Weiser (1993): “standard experimental computer science”, meaning construction of working prototypes in sufficient quantity and showing that it works.

Findings

Presentation of two real-world implementations for Product Lifecycle Information Management, generalisation of these to other kinds of products (appliances, machines, shipments etc.).

Practical implications (if applicable)

Disseminate the existence of an information architecture with ongoing standardization efforts that could address increasingly urgent issues of product lifecycle information management.

What is original/value of paper

No alternative information architecture for the same purpose is known yet. Comparisons are made with e.g. the EPC Global architecture in order to explain why they are not sufficient. The implementations described illustrate how consumers, maintenance providers and manufacturers can be integrated with the architecture in a scalable way.

Keywords: Intelligent Products, Internet of Things, Ubiquitous Computing, Middleware, Product Lifecycle Information Management

1. INTRODUCTION

At least in the industrialized parts of our planet, most people have grown up with some vision of intelligent robots, buildings etc. in films such as Star Wars and Star Trek, as well as in a multitude of Science Fiction books. Despite the enthusiasm for such an imaginary world, our current world is still relatively far from that vision. The scientific domain called Ubiquitous Computing (alternatively, Pervasive Computing, Ambient Intelligence etc.) may be the one that most clearly envisages developing such environments. A new candidate concept, the Internet of Things, could expand the rather local view of Ubiquitous Computing in a way that would integrate backend systems of companies and other organizations with the more embedded processing model of Ubiquitous Computing. The resulting communication infrastructure could enable any computing device to communicate with any other computing device (not to forget the human users), no matter what is their physical size or computational capacity. In this paper, we present an information architecture that enables this vision. The information architecture has been validated by industrial pilots performed at Helsinki University of Technology (TKK) in different application areas. In this paper, we focus on two implementations related to Product Lifecycle Information Management (PLIM) (Harrison et al., 2004; Främling et al., 2007b).

In order to make such an information architecture commercially interesting, it also needs to handle security and copyright issues. These aspects are not covered in this paper but are covered in a security white paper of the PROMISE project (PROMISE, 2004) and will be the subject of a separate research paper under preparation. However, as the proposed information architecture is based mainly on existing Internet technology and protocols, the existing Internet security models are also applicable here.

The research methodology used in this paper is called “standard experimental computer science” (Weiser, 1993), who defines it as follows in the context of ubiquitous computing:

The research method for ubiquitous computing is standard experimental computer science: the construction of working prototypes of the necessary infrastructure in sufficient quantity to debug the viability of the systems in everyday use; ourselves and a few colleagues serving as guinea pigs.

There was originally 11 real-world demonstrators from different domains defined in PROMISE (vehicles, household equipment, telecom equipment etc.). Ten of these demonstrators have been implemented by the PROMISE consortium (one of the PROMISE demonstration owners had to withdraw due to organisational changes). These demonstrators are the main benchmark against which the success of the new concepts, information architecture and implementations are measured. One of these demonstrators has been implemented at TKK, as well as a generic vehicle demonstrator that corresponds to scenarios of other PROMISE demonstrators. The success of the information architecture and the implementations of it are in this paper evaluated mainly based on the following criteria:

1. Degree of fulfilment of the requirements set up by the demonstrator owners.
2. Scalability.
3. Flexibility of the system when organisational, hardware or other changes occurs.
4. To what extent it satisfies the requirements of an Internet of Things.
5. To what extent it satisfies the requirements of PLIM.

The structure of the paper is the following: Section 2 explains the background of Ubiquitous computing and the Internet of Things. Section 2 also attempts to identify how these concepts relate to each other, if and how they could be combined and what remains to be done. Section 3 gives a retrospective view of the work performed in these domains at the BIT Research Centre of TKK and in the PROMISE project, the resulting information architecture and why the information architecture looks like it does. Section 4 presents two implementations of the information architecture in real-life applications made at TKK, followed by conclusions.

2. FROM UBIQUITOUS COMPUTING TO THE INTERNET OF THINGS

The term *Ubiquitous Computing* (Ubicomp) was apparently launched by Mark Weiser in 1988 (Weiser, 1991). The idea of Ubicomp is that computing will eventually be embedded into most everyday objects, which will sense, communicate and possibly act in a way that is invisible for the human user. Invisibility may not be complete; it could also signify that the user interface is conceived in a way that makes user interaction so natural that the human user doesn't realize that she or he is interacting with a potentially complex computing environment "behind the scenes". Simple, everyday examples of such user interfaces are changing the desired temperature setting for a room, which in modern heating systems may lead to a computation that takes into account the desired temperature, the current room temperature, the outside temperature, a thermodynamic model of the room and possibly even the weather forecast – or at least the current outside temperature gradient. Another example is the car, where the main user interface components have remained essentially the same for decades, even though they have now become indicators of a "desired state" for a digital control system rather than controlling devices directly through a mechanical link.

Ubicomp is inherently *distributed* because of the amount of embedded computing in devices that need to interact in order to accomplish the tasks they are supposed to. The temperature control example mentioned could involve the temperature sensing and control module in the room, the controller of the central heating, an outdoor temperature sensing unit and a weather forecasting service available over Internet. When Weiser wrote his article in 1991, the World Wide Web did not yet exist so such weather forecasting services did not exist or at least were not easily integrated with. This is where the concept of *Internet of Things* fits into Ubicomp, i.e. as an extension of mainly locally networked smart devices into globally networked computer systems of all sizes, from the smallest to the largest. The smallest could be mere object identifiers stored as barcodes, Radio Frequency Identification (RFID) tags etc. The largest could be servers, company-

wide Enterprise Resource Planning (ERP) systems, Product Data Management (PDM) systems etc., which may not fit into the traditional Ubicomp view.

The Internet of Things concept is explicitly mentioned e.g. in (Brock, 2001; Huvio et al., 2002; Gershenfeld et al., 2004) but the name Internet of Things seems to have been used in different contexts already before these papers. Unfortunately, the Internet of Things concept is often interpreted in a very RFID- and Supply Chain Management (SCM) centric way as in (Brock, 2001) and as promoted by the EPCglobal organization (http://www.epcglobalinc.org/). This limited view of the Internet of Things concept is rather focused on product identification technologies, tracking of product locations and stock levels and industrial systems than on everyday objects. Because of the focus on one Auto-ID technology of many (RFID) and one specific application area (SCM), the information architecture and the interface standards created by EPCglobal tend to have limitations that make the name Internet of Things inappropriate for them. Such limitations are for instance:

- Hierarchical and uni-directional: data flows "upwards" only from RFID readers towards backend systems (as opposed to the device-to-device communication that is typical in Ubicomp systems).
- The identifier space is "closed" by the support for Electronic Product Codes (EPC) only, which are centrally managed by GS1 (as opposed to Ubicomp, where locally unique identifiers can be negotiated when needed if existing serial numbers or similar are not sufficient). Other alternatives to the EPC are presented e.g. in (Främling et al., 2007b).
- The focus on RFID tags and their price means that devices with embedded computing power are hardly considered in the architecture.

In the next section we will describe one alternative definition and implementation of the Internet of Things to the EPCglobal one that enables both the Ubicomp and the EPCglobal views and would be as universal as the Internet itself. This work was started at TKK in 2001, first in the Tekes-funded projects Dialog and EloCore and then in the PROMISE (PROMISE, 2004) project of the EU 6th Framework Program.

3. FROM INTERNET-BASED SHIPMENT TRACKING TO PRODUCT LIFECYCLE INFORMATION MANAGEMENT

This section gives a retrospective overview of how a system that was initially developed for global, Internet-based tracking of shipments turned out to also provide much of the functionality needed by the Internet of Things, including Ubicomp functionality. The reason for providing this overview is that it shows how the same architecture can provide a universal solution both to the needs of the RFID/SCM-based Internet of Things and the Ubicomp-enabled Internet of Things. Since the beginning of this storyline in 2001, industrial pilots and real-life demonstrations have been implemented that demonstrate that the architecture is a valid platform for the Ubicomp-enabled Internet of Things.

3.1. Shipment tracking with the Dialog software

The Dialog research project (DIALOG, 2001) was defined based on experience gained from earlier e-commerce projects where computer programs based on the peer-to-peer paradigm had been developed mainly for exchanging sales forecasts between different organisations. The initial application area of Dialog was to develop a forwarder independent tracking-and-tracing system for worldwide project deliveries. Since the Dialog project ended, the name "Dialog" has been used to refer to the software initially written during the Dialog project. The development of the Dialog software has continued since the Dialog project ended and is still ongoing.

In order to create a globally unique product identifier that would also indicate where information updates about shipments should be sent, a solution labelled "ID@URI" was chosen, where URI is a computer address (e.g. 'www.some_company.com') and ID is a serial number or any other unique number at the URI indicated. A system using this notation was installed in 2002 for forwarder-independent tracking of project deliveries (Kärkkäinen et al., 2004). In this pilot, the ID was the unique serial number of the RFID tags used, while the URI part was written into the RFID tag's memory. When a shipment was ready for transportation, a shipment label with the RFID tag and other labelling information was attached to it. Due to the great number of sub-contractors involved, this labelling of shipments took place in many different geographical places and was performed by many different organisations but the URI always pointed to the projecting company's "tracking agent". Whenever a shipment was observed at a tracking point (e.g. loaded on truck, handled in a harbour or at a building site), a location update was sent to the projecting company, as illustrated by Figure 1. The same principle was used in another pilot performed in 2003 but with both ID and URI written with barcodes (Kärkkäinen et al., 2005). An extensive comparison between ID@URI and other alternatives can be found in (Främling et al., 2007b).

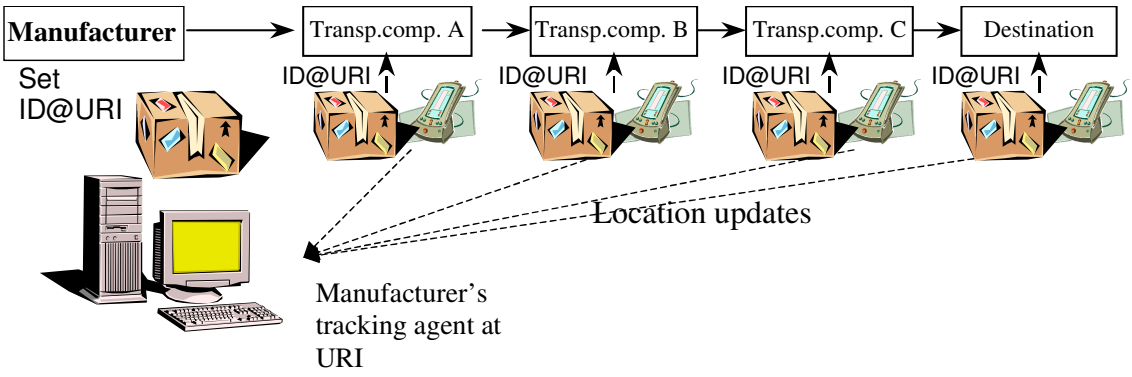

Figure 1. ID@URI based tracking, reproduced from (Främling et al., 2003).

Nearly from the beginning it became clear that shipments are just a more transient variant of products or physical objects in general, while the location of an object is simply a property among others of the object. Therefore any property of any object could be updated or retrieved (if permitted by the security settings) using the same architecture. The *product agent* concept (Främling et al., 2003; 2006) was introduced as the virtual counterpart of the physical object that would enable the creation of *Intelligent Products*

(Kärkkäinen et al., 2003a). In the SCM domain, these intelligent products were the cornerstone behind the product-centric information management concept described in (Kärkkäinen et al., 2003b). Identifying the parallel between object-oriented programming and product agents made it possible to apply Design Patterns (Gamma et al., 1995) also to managing data about product items even when it is spread over organizational borders (Främling et al, 2004; Främling et al., 2007a).

3.2. Product Lifecycle Information Management and the PROMISE project

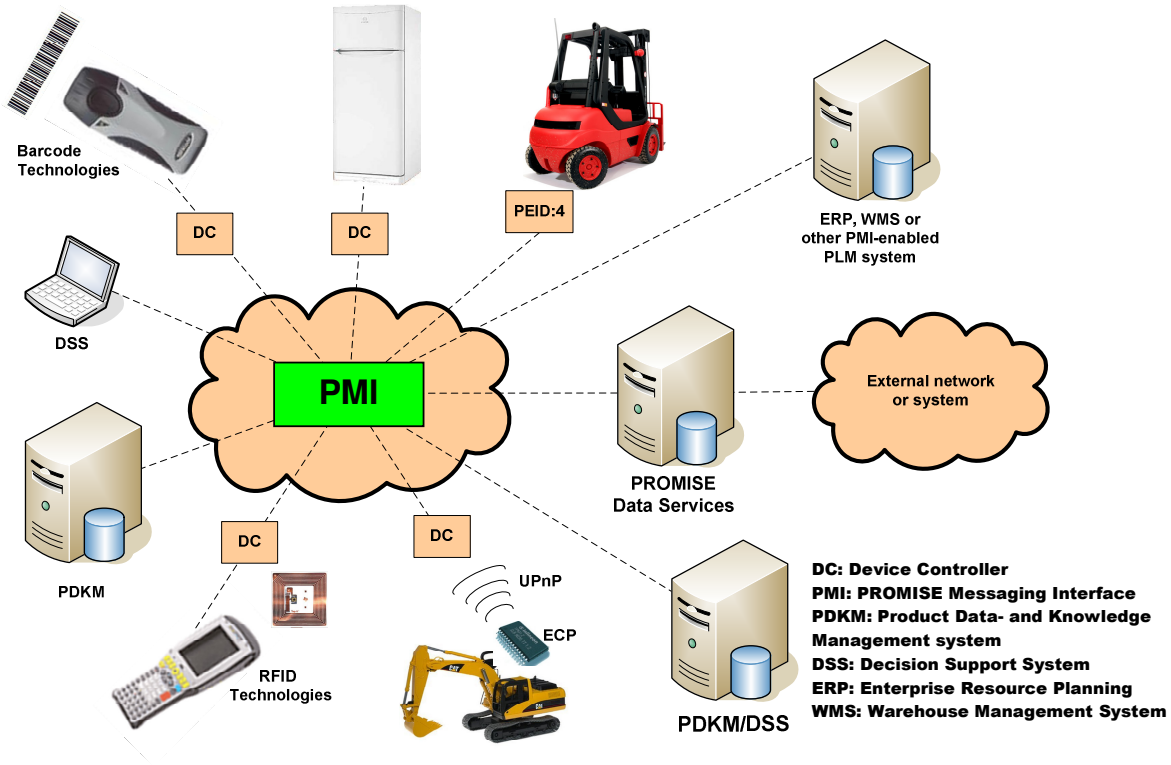

Figure 2. Illustration of PROMISE architecture and connectivity (PROMISE, 2008).

As a partner of the PROMISE project (Kiritsis et al., 2003; PROMISE, 2004) that started in 2004, it was a challenge to see to what extent the Dialog architecture would also be suitable for Product Lifecycle Information Management in a context where data needed to be collected from many kinds of product items during their whole lifetime. This could even imply the collection of data when a product item was used by consumers as in the refrigerator and car applications of PROMISE. Since the initial system architecture ideas described e.g. in (Anke & Främling, 2005), the PROMISE system architecture has gradually evolved into the one illustrated in Figure 2. This architecture uses a peer-to-peer information exchange model, where any device that implements the Web Service-based PROMISE Messaging Interface (PMI) can communicate with any other device that supports PMI, no matter the size of the device. If the Product Embedded Information Device (PEID) does not have enough computational power or communication capabilities for implementing PMI, then it connects either through a device-specific Device Controller or using the UPnP-based CorePAC interface defined in PROMISE. Otherwise

it is called a PEID:4 according to a classification based on computation and communication capabilities that is documented in PROMISE deliverable "DR5.4: Generic PEID roadmap for each group".

The PMI is a key interface which enables a web-services based approach, permitting any PMI-enabled user to exchange data with another. Depending on the complexity of any specific application, this can be achieved on a simple peer-to-peer basis if the two users are known to each other, or on a more complex wide-area basis using advanced PROMISE Data Services (middleware).

The PROMISE connectivity model is similar to that of the Internet itself. Where the Internet uses the *HTTP* protocol for transmitting *HTML*-coded information mainly intended for *human users*, PROMISE uses the *PROMISE Messaging Interface (PMI)* for transmitting *XML*-coded information mainly intended for automatic processing by *information systems*. It is important to understand these relationships because PROMISE in effect proposes an extension to the Internet itself.

The next section shows how the Dialog system has been used as an implementation platform for PMI and the PROMISE architecture in general. As the original Dialog architecture and the PROMISE architecture are nearly identical, this implementation task mainly consisted in adding a new networking component that uses PMI instead of using one of the existing Dialog networking methods. Because Dialog was already from the beginning designed to support different protocols and interfaces, adding a new one for PMI was rather straightforward.

4. INTERNET OF THINGS IMPLEMENTED – TWO REAL-LIFE EXAMPLES

Today, many products have an embedded control computer that controls various functions of the product. A good example is the computer system embedded in modern cars, which monitors the various subsystems, provides the user with reminders about scheduled maintenance and notifies the owner of possible error conditions by the "Malfunction Indicator Light" that is usually labelled "Check Engine" on the car dashboard. Such PEIDs are also starting to appear in ordinary household appliances. In this section we look at how such an appliance is integrated with the Dialog platform, and how an installation in a real building or home would be configured. We also take a look at how vehicle diagnostics can be transmitted using PMI.

Dialog is a "generic" software in the sense that it provides protocol- and interface-neutral message passing mechanisms with message persistence functionality, security mechanisms etc. that are abstracted away from the "business logic" itself, implemented by "agents". Figure 3 illustrates the internal architecture of a Dialog node. We see that the components involved in sending and receiving messages, and agents, which consume and produce messages, are separated as their own classes with a common interface, i.e. the receive and send handlers. This signifies that different protocols and messaging interfaces can be easily supported. Already before PROMISE, Dialog supported a Java remote method invocation (RMI) interface, a Web Service interface using the Simple Object Access Protocol (SOAP) and an interface using HTTP POST messages. For implementing the PMI, it was sufficient to add new SOAP-based PMI receiver and

sender classes. A simple and configurable mapping mechanism that is internal to the Dialog node defines what messages should go to which agent(s) and what sender should be used for which messages.

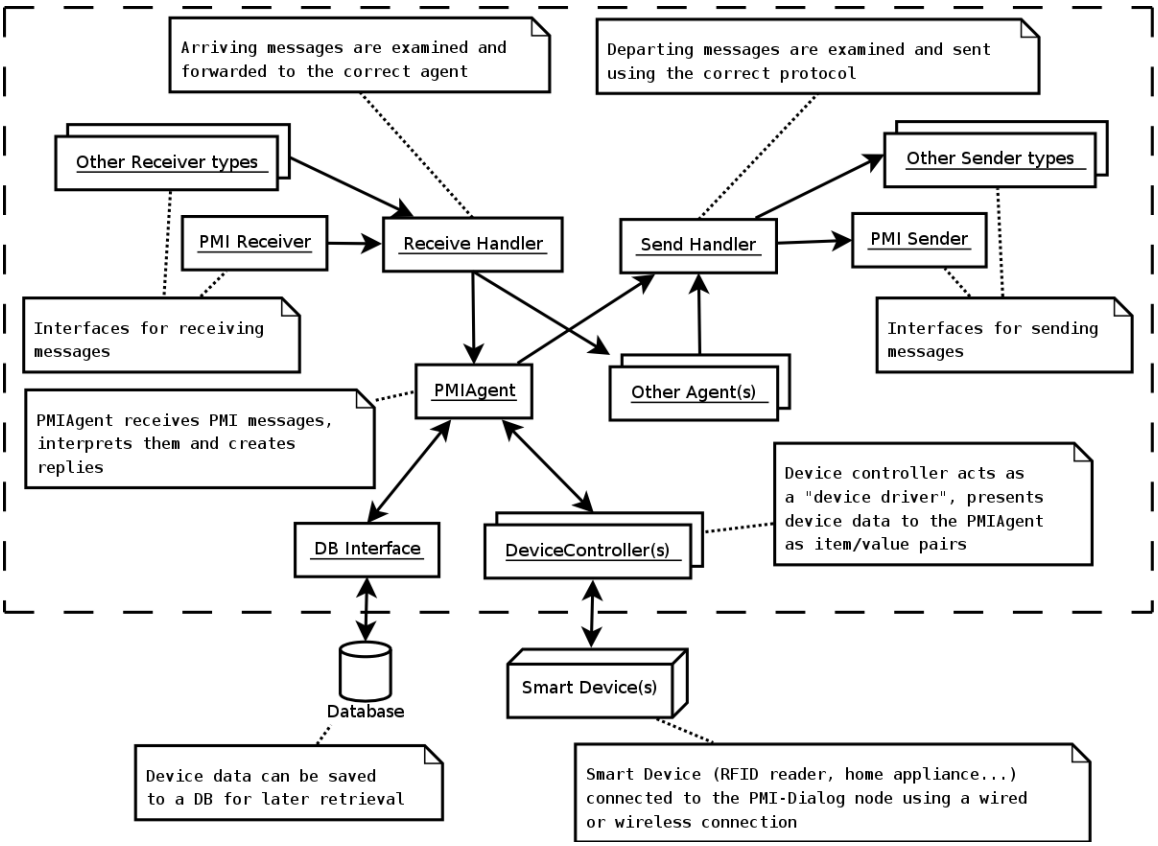

Figure 3. Internal architecture of a DIALOG node.

Dialog agents are free to process the messages as they like, and may send messages at will. Dialog was adapted to support PMI-specific functionality by the addition of a PMI-specific agent, which mainly means implementing Device Controller (DC) functionality. As one of the PROMISE demonstrators, support for an intelligent refrigerator control system was added to the Dialog system by adding a corresponding DC that enables information, alarms and other events to be obtained from the refrigerator and sent to a remote location using PMI.

The statistical data obtained from the appliances installed at the customer's premises can be used to detect service needs in advance, before a failure occurs (condition-based maintenance), thus offering improved service for the customer (Cassina et al., 2007). It could also be possible to determine in advance which spare parts are needed for the job and improve the scheduling of service personnel. This is an example of how a refrigerator product-selling activity could gradually change into a service-selling activity, i.e. selling "refrigeration services" rather than selling the physical refrigerator itself.

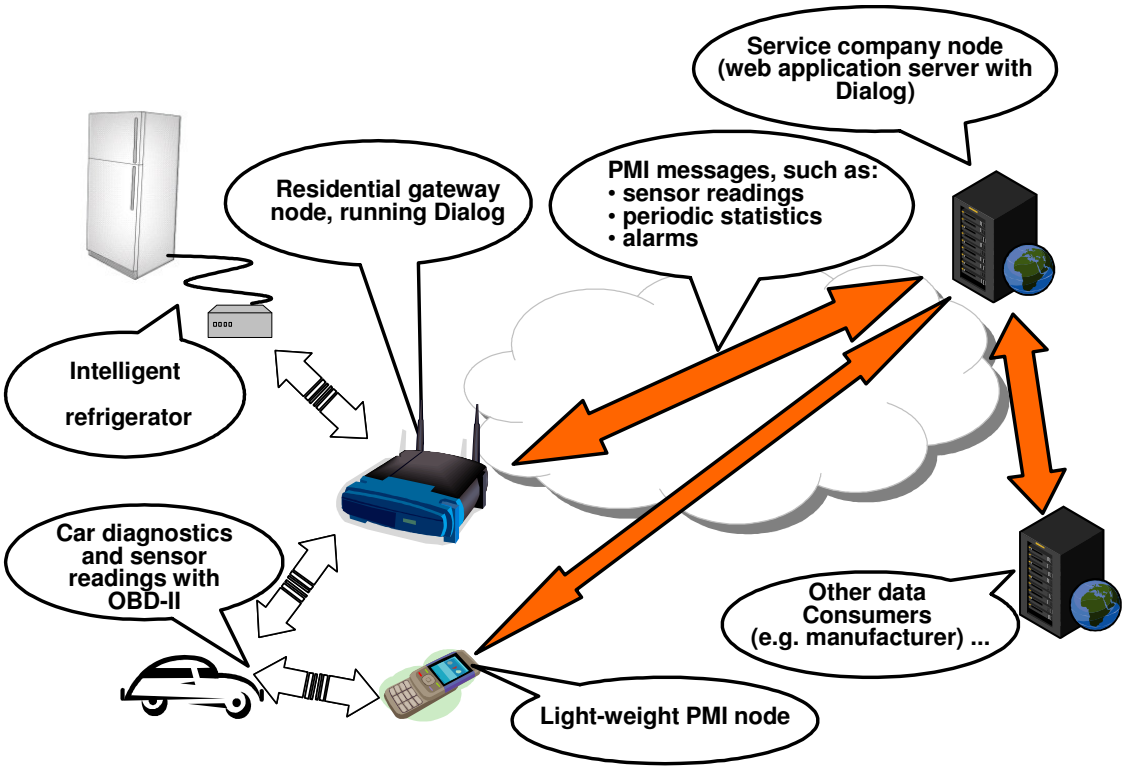

Figure 4. A residential gateway acts as a message interface that enables simple product-embedded information devices to participate in PMI communications over the Internet.

A specific DC is needed for interfacing with the refrigerator due to the proprietary protocol used by the refrigerator. Other possibilities could have been to implement the PMI on the refrigerator itself or to implement the more light-weight UPnP-based CorePAC protocol on it. However, both options were excluded because they would have increased the cost of the system too much to remain commercially defendable. Therefore, at least in the near future, such household appliances' PEIDs are likely to implement a simpler protocol for communicating data to a PMI node that functions as the endpoint for PMI communications. The PMI node could take the form of a residential gateway, similar to the broadband routers on the market today, but equipped with PMI- and DC-implementing software (Figure 4).

The PEID is usually involved in controlling functional aspects of the product, such as engine control in a car, or climate control of a refrigerator. Another important functionality that PEIDs provide is diagnostics. In a state-of-the-art car, the diagnostics notifications are only for the driver to notice and act upon. The effectiveness of on-board diagnostics could be enhanced by enabling the notifications to pass to the car owner and even to the service company directly, for instance via an ordinary mobile phone with suitable software. We have implemented such a system on a Nokia Series 60 mobile phone by a Java MIDP program capable of acting as a node that sends PMI messages over the mobile network with data downloaded from the car's Engine Control Unit (ECU). The connection between the mobile phone and the car's ECU was implemented using a commercially available OBD-II protocol converter connected to the mobile phone via Bluetooth. The setup enables diagnostics notifications from the car to be sent to a

remote node in real time. The remote monitoring node can in turn place a request for the PMI node in the mobile phone to send specific sensor values periodically to the remote monitoring node. This information can then be used to aid in further problem determination, scheduling a time for service, ordering needed spare parts or taking some other proactive actions. The collected information could also potentially be transmitted to the car manufacturer. If the car manufacturer could collect such real-use information from a sufficient amount of cars, it could lead to improved maintenance scheduling, product design and manufacturing procedures. This is true also for most other manufacturing companies (such as the refrigerator manufacturer), i.e. product design, manufacturing and possibly also recycling could be improved if a sufficient amount of in-use information can be collected.

5. CONCLUSIONS

In the introduction, five criteria were proposed for the analysis and assessment of the degree of success of the information architecture and implementation presented in this paper. Based on the current results, we can conclude the following:

1. *Degree of fulfilment of the requirements set up by the demonstrator owners*. What comes to the PROMISE information architecture, it has allowed all PROMISE demonstrators to be successfully implemented. It has also been admitted both by demonstrator-owners and technical solution providers that the current architecture is more adequate than an initial information architecture that was rather hierarchical and rigid. The "owner" of the refrigerator demonstrator has also been satisfied with the system implementation.
2. *Scalability*. The peer-to-peer based information exchange mechanism makes it easier to avoid centralised information storage and other potential bottlenecks. Because PROMISE can be seen as an extension of the Internet, the PROMISE information architecture is in general as scalable as the Internet itself.
3. *Flexibility of the system when organisational, hardware or other changes occurs*. By the use of PMI for nearly all information exchange, it is easy to split, merge or move functionality of the software to new hardware or even new organisations because it can be done in a way that is more or less transparent to other parties.
4. *To what extent it satisfies the requirements of an Internet of Things*. Any "thing" can be connected using the proposed architecture, with support for read and write operations, subscriptions to information updates, alarms and similar events. "Meta-data" operations are also supported, such as querying for the list of properties that can be read, their units etc., as well as updating (through a "write" operation) them if needed. Therefore, we consider that most requirements for an Internet of Things are fulfilled.
5. *To what extent it satisfies the requirements of PLIM*. The architecture was designed for the specific needs of PLIM and does allow information retrievals and updates about products during their whole lifetime.

We also claim that the proposed information architecture fulfills the needs of what we call the Ubicomp-enabled Internet of Things, as shown by earlier industrial pilots in

shipment tracking and tracing, the real-life implementations explained in Section 4 and other real-life implementations performed in the PROMISE project. Even though we are not currently aware of other information architectures with the same scope, partially similar but more domain-specific approaches exist such as oBIX (Open Building Information Xchange) for "intelligent buildings" or the EPC Network for SCM. Time will show what the final architecture will be called and what standards will become predominant but at least the building blocks now exist for implementing real-life Ubicomp-enabled Internet of Things applications. Such applications would open new "service supply chain" market opportunities for manufacturing companies, service providers and many other commercial actors so the economical impact on society will be significant.

ACKNOWLEDGEMENTS

The work reported here has mainly been funded by Tekes (Finnish Funding Agency for Technology and Innovation) through the Dialog and EloCore research projects and by the EU 6th Framework Program through the PROMISE project. The industrial partners of the Dialog project have also contributed financially, while the industrial partners of both the Dialog and PROMISE projects have provided useful insight into real-world applications, as well as the possibility to perform pilot installations with access to their equipment, infrastructure and professional experience.

The PROMISE architecture and the PMI are a joint result of PROMISE partners, notably TKK, Trackway (especially Björn Forss and Jouni Petrow), Indyon (David Potter), SAP (several participants), InMediasP (notably Michael Marquard) and BIBA (notably Robertino Solanas). The authors hope they have not forgotten any key contributors but if we have done so, we present our excuses in advance.

In addition to the authors, especially Vincent Michel from Ecole Nationale Supérieure des Mines de Saint-Etienne, France, has done a major programming effort on the PMI implementation in Dialog. The authors would like to thank him and earlier contributors to the Dialog software for their effort.

The authors are also grateful to the reviewers who have shown a good understanding of the paper and proposed useful improvements to the initially submitted version.

REFERENCES

Anke, J., Främling, K. (2005), "Distributed Decision Support in a PLM scenario", in: *Proceedings of Product Data Technology Europe 14th Symposium*, 26-28 September 2005, Amsterdam, Netherlands, pp 129-137.

Brock, D. L. (2001), *The electronic product code (EPC)-a naming scheme*, Technical Report MIT-AUTOIDWH-002, MIT Auto-Id Center, Massachusetts Institute of Technology, Available from http://www.autoidlabs.org/uploads/media/MIT-AUTOID-WH-002.pdf, accessed March 26, 2008.

Cassina, T., Tomasella, M., Matta, A., Taisch, M., Fclicctti, G. (2007), "Closed-loop PLM of Household Appliances: An Industrial Approach", in: Olhager, J., Persson,

K. (Ed.), *IFIP International Federation for Information Processing, Volume 246, Advances in Production Management Systems*, Boston: Springer, pp. 153–160.

DIALOG (2001), *Distributed information architectures for collaborative logistics*, Available from http://dialog.hut.fi/, accessed March 20, 2008.

Främling, K., Holmström, J., Ala-Risku, T., Kärkkäinen, M. (2003), *Product agents for handling information about physical objects*, Technical Report of Laboratory of Information Processing Science series B, TKO-B 153/03, Helsinki University of Technology, 2003.

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. (Ed.), *Proceedings of IMS International Forum*, Cernobbio, Italy, 17-19 May 2004, pp. 1424-1431.

Främling, K., Kärkkäinen, M., Ala-Risku, T., Holmström, J. (2006), "Agent-based Model for Managing Composite Product Information", *Computers in Industry*, Vol. 57 No. 1, pp. 72-81.

Främling, K., Ala-Risku, T., Kärkkäinen, M., Holmström, J. (2007a) "Design Patterns for Managing Product Life Cycle Information", *Communications of the ACM*, Vol. 50 No. 6, pp. 75-79.

Främling, K., Harrison, M., Brusey, J., Petrow, J. (2007b), "Requirements on unique identifiers for managing product lifecycle information - comparison of alternative approaches", *International Journal of Computer Integrated Manufacturing* , Vol. 20 Issue 7, pp. 715-726.

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

Gershenfeld, N., Krikorian, R., Cohen, D. (2004), "The Internet of Things", *Scientific American*, Vol. 291 No. 4, pp.76–81.

Harrison, M, McFarlane, D., Parlikad, A.K., Chien Y.W. (2004), "Information management in the product lifecycle - the role of networked RFID", in *Proceedings of 2nd IEEE International Conference on Industrial Informatics*, Berlin, Germany 24-26 June 2004.

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

Kiritsis, D., Bufardi, A., Xirouchakis, P.C (2003), "Research issues on product lifecycle management and information tracking using smart embedded systems", *Advanced Engineering Informatics*, Vol. 17 No. 3-4, pp. 189-202.

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

Kärkkäinen, M., Ala-Risku, T., Främling, K., (2004), "Efficient Tracking for Short-Term Multi-Company Networks", *Int. J. of Physical Distribution and Logistics Management*, Vol. 34 No. 7, pp. 545-564

Kärkkäinen, M., Ala-Risku, T., Främling, K., Collin, J. (2005), "Establishing inventory transparency to temporary storage locations", in: *Proceedings of Advances in Production Management Systems (APMS)*, 18-21 September 2005, Washington, USA.

PROMISE (2004), *Product Lifecycle Management and Information Tracking using Smart Embedded Systems*, Available from http://www.promise-plm.com/ and http://www.promise.no/, accessed March 20, 2008.

PROMISE (2008), *PROMISE Architecture Series Volume 1: Architecture Overview*, Forthcoming.

Weiser, M. (1991), "The computer for the twenty-lirst century", *Scientific American*, Vol. 265 No. 3, pp. 94-104.

Weiser, M. (1993), "Some computer science issues in ubiquitous computing", *Communications of the ACM*, Vol. 36 No. 7, pp. 75-84.