1

Instance-informed information systems: a pre-requisite for energy-efficient and green information systems

Kary Främling[1], Jan Nyman[2], André Kaustell[3], and Jan Holmström[4]

[1] School of Science, Aalto University, PO Box 15500, FIN-00076 Aalto, Finland
Kary.Framling@aalto.fi,
[2] ControlThings Oy, Råbackavägen 7, FIN-06650 Hammars, Finland
Jan.Nyman@controlthings.fi
[3] Posintra Oy, Electrical Building Services Centre, Gnistvägen 1, FIN-06150 Borgå, Finland. Andre.Kaustell@posintra.fi
[4] School of Science, Aalto University, PO Box 15500, FIN-00076 Aalto, Finland
Jan.Holmstrom@aalto.fi

Summary. In order to create energy-efficient, ubiquitous and green information systems, it will be necessary to exchange information and co-ordinate action between systems that have traditionally not been interoperable. Providing such interoperability is a major challenge unless the underlying information systems are implemented in a way that allows for the collection of relevant information, such as (near) real-time sensor readings and higher-level events, as well as and the ability to take proper actions based on that information. The paper shows how and why instance-informed information systems are a requirement for implementing energy-efficient and green information systems, as well as showing how such information systems have been implemented in several real-world applications.

Key words: Instance-informed, agent processing, Green IS, energy informatics.

1.1 Introduction

Climate action and reducing greenhouse gases has become a top topic in EU and all over the world. Transports and living, i.e. the heating and cooling of buildings, both play a major role for the CO_2 emissions in the world. In this paper, we present solutions for local and remote monitoring of the energy usage of buildings and vehicles. We also show how relevant information can be collected and presented to the users and owners of buildings and vehicles that allows them to identify the most efficient ways to reduce their energy usage if they want to do so.

Information systems (IS) tracking consumption on the level of the individual artefact, or what we in this paper term *instance* are a cornerstone

for achieving energy efficiency. In this paper we propose instance-informed information systems as a platform for the collection and analysis of energy data sets to support the continuous optimization of energy efficiency while promoting 'green' values.

A context where instance-informed IS has already resulted in considerable improvements of energy efficiency in the past, and still continuing today, are the computerized engine control systems of motor vehicles introduced since the 1980's. Initially the control systems were introduced to reduce pollution but were later also used to reduce fuel-consumption. Computerized engine control systems are instance-informed because they continuously adjust and adapt the control parameters of the engine depending on the individual tear and wear of the engine, as well as on the current atmospheric conditions. When such instance-informed vehicles become internet-connected, the possibility to gather information from fleets of vehicles will make it possible to further optimize engine design and tuning, as well as optimizing the usage of vehicle fleets, and improving the behaviour of drivers.

It is important to make a clear distinction between Green information technology (IT) and Green IS [1]. Green IT is more focused on how to make computing infrastructure more energy efficient, whereas an information system is the combination of people, processes, and technologies that enables the processing of digitized information [2]. In this paper we focus on Green IS, which in [1] is defined to be a design and implementation of information systems that contribute to sustainable business processes. In this paper, we furthermore focus on *energy informatics* in particular, which is defined as [3]:

Energy informatics is concerned with analyzing, designing, and implementing systems to increase the efficiency of energy demand and supply systems. This requires collection and analysis of energy data sets to support optimization of energy distribution and consumption networks.

Because the scope of Green IS extends far beyond a single organisation, the required IS must also be able to reach across multiple organisations. One domain that has similar requirements on multi-organisational information exchange is the Internet of Things (IoT) [4][5]. The information system architecture presented in this paper, and in use in many of the case applications, was originally developed mainly for the IoT and product lifecycle information management [6]. Due to the similarity of requirements we can introduce instance-informed IS as a platform to fulfil the requirements of energy informatics, as well as Green IS in general.

After this introduction, the paper provides the basic theory and building blocks behind instance-informed IS. Then we show how such instance-informed IS have been implemented in several Green IS-related applications and why instance-informedness is necessary for a successful implementation of such information systems, followed by conclusions.

1.2 Instance-informed Information Systems

Our work on instance-informed IS was initiated as an IoT design. Several definitions and views exist on the IoT but they all include the need to represent and handle unique instances of products and 'Things' in general. A manufactured Thing becomes a unique instance at the latest when it enters its usage phase. A product in use exits the realm of conventional product-type based product lifecycle management. The product lifecycle information management concept [6] was developed as a cradle-to-cradle approach to extend product lifecycle management from the design and manufacturing phases to the usage and end-of-life phase of a manufactured product's instance-specific lifecycle [7], not forgetting the possible refurbishing and recycling of the Thing or parts of it.

Most needs and applications considered here are related to the product usage phase. In the rest of this section we will define instance-informed IS, present the main building blocks and theoretical implications of them and describe how such a system has been implemented.

1.2.1 Building blocks of instance-informed IS

The need for instance-informed IS was initially identified when attempting to model objects in the real world that all had individual properties and possibly also different functions that can be performed on them, which lead to the creation of object-oriented languages. The first truly object-oriented language, Simula 67 [8], was developed for system description and simulation programming that corresponded to real-world objects such as industrial plants. The number of properties and relationships between different objects grew too big and complex to handle by existing procedural languages. A great breakthrough happened in the 1980's when graphical user interfaces were developed at the Rank Xerox Palo Alto Centre. The object-oriented language Smalltalk was developed at the same time [9] for making it easier to manage the properties of a multitude of visual elements on the screen as well as the relations between them. Most of the current popular programming languages such as Java, C++ and C# can be considered to be derivatives of Smalltalk.

When creating information systems for the IoT we are essentially facing similar challenges as the designers of object-oriented languages, i.e. how to manage instance-specific information. The new challenge in the IoT context is that the information systems that produce, store and process that information tend to be distributed over different computing devices that may belong to different organisations and where Things may be mobile. Mobility creates technical challenges due to intermittent network connectivity, but also creates new challenges regarding e.g. security and privacy. Furthermore, with the increasing use of embedded computing in real-world products, we are also facing challenges in managing information collected by millions of embedded computers regarding the physical product that they are mounted on. On the

minimal level, the embedded computer can be just an identification device such as a barcode, RFID tag, magnetic stripe or similar, which means that much of the information about the instance needs to be stored remotely. Product lifecycle information management is a domain where information about the usage of every product instance needs to be stored and communicated when needed and where it is needed. Design and manufacturing information are also relevant. The challenge is that all the different pieces of information tend to be distributed over company servers, service company computers as well as the embedded computers of the instances.

Instance-informed IS make the identification of instances, and the tracking and handling of instances over long time-spans and many locations the basis of the design. The design makes it possible to track changes of any property of any instance, such as location and age. This approach to managing information about product instances has been presented under different names in the past, such as product-centric information management [10], item-centric information management [11] and intelligent products [12][13][14].

An instance is unique. This means that it has a set of properties that have unique values at any moment in time. A car has a certain colour, a certain number of horse powers etc. at any given moment in time. The number of properties can easily become huge when going down to all the details of any product instance. However, different organizations and individuals may use different views that contain different sets of the instance's properties, as well as different names, units, languages etc. for them. This often becomes a practical challenge because it gives interoperability issues when exchanging the information. Standards for messaging interfaces and information representation are usually the best way of overcoming such interoperability issues. In practice, there are many domains where such standards do not exist. IoT is one of those domains, where the EPC Information Services (EPCIS) [15] and related standards attempt to provide a universal and standardized way of finding and exchanging Thing-related information. So far, EPCIS is defined and used almost exclusively for supply chain management applications. Therefore, EPCIS is not a universal information exchange standard for the IoT.

Another challenge for handling instance-specific information is when database and IS designs either explicitly or implicitly assume that all instances must belong to a class, as is often the case according to the extensive survey of systems engineering literature in [16]. Parsons and Wand claim that the assumption of inherent classification, which is due to the human cognitive way of processing, represents a major obstacle to evolution and innovation. This solution design is not a technical necessity but based on the IS designer's assumption of inherent classification of instances. The assumption means that Things can be referred to only as instances of classes. In the context of operations management the assumption of inherent classification of materials means that all product items must be assigned to a material type. The tendency to manage material flows as transfers between stocks that only modify the count of identical products in different places, rather than tracking movements of

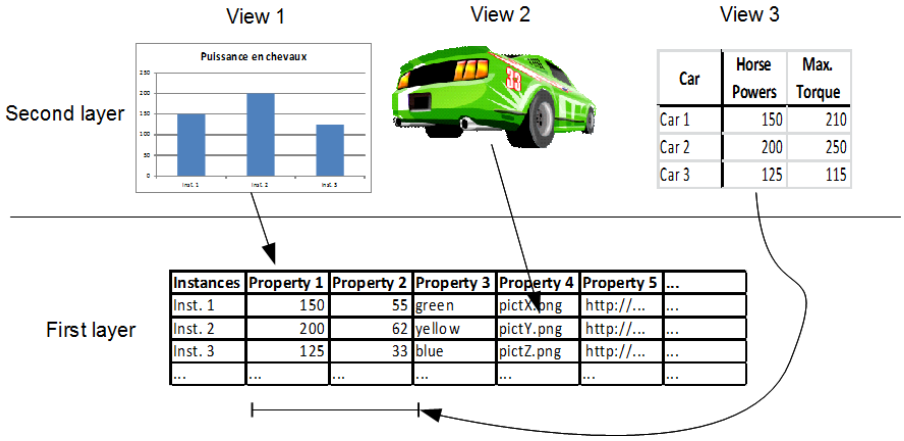

Fig. 1.1. Example of two-layered architecture, with unique instances but several possible views on them, e.g. as a bar in a graph, as a picture or as as a row in a table with selected properties.

instances, follows from this general preference for classification over identification. This type of approach makes it difficult to deal with instances that are heterogeneous and changing. For dealing with heterogeneous and changing instances there is a need for an inter-organizational tracking solution that is loosely coupled and not tightly linked to any particular locations or actors.

Parsons and Wand propose to overcome these challenges by introducing the two-layered architecture illustrated in Fig. 1.1. The first layer represents unique instances with their unique properties, independent of any classes to which the instances may belong. The second layer consists of views based on sets of properties that are relevant for a particular purpose, organisation or user. Such views may be created, removed or modified at any time without affecting the instances in the first layer.

In practice, the digital representation of instance property values, i.e. the first-layer values, are typically stored in the instance's embedded computer or in a database (or possibly even several databases). In addition to instance property values, relations between different instances and other information entities should preferably be handled as first-layer information because organisation-dependent vocabularies used for representing such relations easily become a major obstacle for interoperability. From a data consistency point of view it would be desirable that these first-layer values would be stored in one single database. Access to that database should always be done through a software component that takes care of the consistency of the first-layer instance information, such as the Product Agent approach proposed in [17]. Such a Product Agent can then provide different views to the instance information by implementing different communication protocols, messaging interfaces or even graphical user interfaces [18][19].

1.2.2 Instance-informed information processing

In the theory of information technology proposed in [20], p. 280 the representation and manipulation/processing are distinguished as the two aspects always present in an IS. In the definition of sensor networks in [3], terms used such as 'reports the status', 'can be analyzed' etc. are clearly information manipulation and processing actions. Multi-agent systems and agent processing [21] are one model for implementing instance-informed information processing.

Two-layer architectures consisting of instances and contingent aggregation are effective because the architectures create environments where interactive agents can effectively harness information provided by the environment [18]. The responses of interactive agents are not anticipated or pre-designed but formulated based on sensing the environment and interaction with other agents [22]. Actions performed by agents are typically e.g. data filtering, rule-based processing, neural net-based detection of events, event generation etc. [23].

In order to enable agents to do processing of information from different organisations, the agents must be able to access and understand the information in a uniform way. In object-oriented programming, the Design Pattern concept [24] is commonly used for defining 'good' designs that combine recommendations for information storage and for information processing. The extension of design patterns to instance-informed IS has been proposed e.g. in [25]. A major challenge in instance-informed IS compared to ordinary object-oriented programming is that universal access to information requires standardised interfaces to run the typically multi-organisational applications envisioned, of which the IoT is probably the most universal one. There is currently a lack of such universally accepted standardised interfaces, even though some standardisation initiatives exist, such as the already mentioned EPCIS standard, as well as the Quantum Lifecycle Management workgroup of The Open Group (`www.opengroup.org/qlm/`).

1.2.3 Implementation of instance-informed IS

Instance-informed IS can be implemented in many ways while respecting the principles of a two-layered view and processing based on design pattern principles. One example of such an IS is the open-source DIALOG platform (`http://dialog.hut.fi`), which was from the beginning developed for the requirements of IoT in 2001 [10].

DIALOG is generic software in the sense that it provides protocol- and interface-neutral messaging mechanisms with message persistence functionality, security mechanisms, and so forth, which are separated from the business logic itself, which is implemented by agents. Fig. 1.2 illustrates the internal architecture of a DIALOG node. We see that the components involved in sending and receiving messages are separated from agents who consume and produce messages; each has its own classes with a common interface, i.e., the receive and the send handlers. Different protocols and messaging interfaces can

be easily supported. DIALOG supports e.g. 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. Implementing a new interface or 'view' only requires adding corresponding receiver and sender classes. The DIALOG node contains a configurable mapping mechanism that defines what messages go to which agent(s) and what sender to use for which messages.

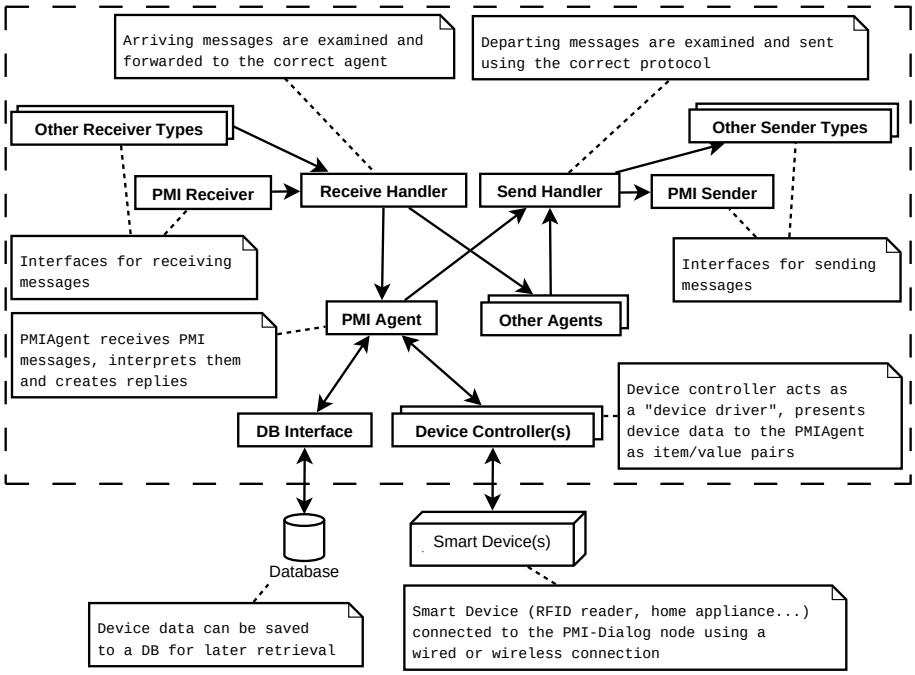

Fig. 1.2. The Dialog architecture. The dashed line shows the node boundary. The Dialog architecture can support many different interfaces for receiving and delivering messages using e.g. the PMI (PROMISE Messaging Interface) protocol.

DIALOG has been used as the platform for implementing the remote monitoring systems of vehicles described in the next section. Remote monitoring and control of energy consumption have been implemented following similar principles. However, the actual interfaces, communication protocols etc. have been selected based on the current state of the art technology in the building automation domain.

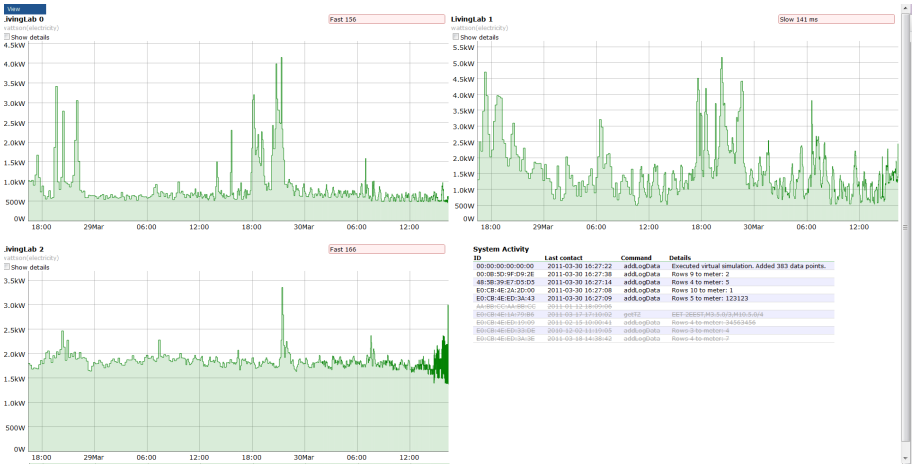

Fig. 1.3. Visualisation of energy consumption from three living lab targets, including status information.

1.3 Remote Monitoring and Control of Energy Consumption in Buildings and Vehicles

An energy monitoring system that visualizes the actual energy consumption of a house or a flat in real time has been developed at the Electrical Building Services Centre in Porvoo, Finland (`www.stok.fi/eng`). A resident may instantly 'see' his or her energy consumption as a constantly updating graph on a web page, as shown in Fig. 1.3. The graph makes it possible to visualize the effect of various energy savings measures. One instantly gets feedback on how energy consumption is affected by switching on and off electrical devices, for instance a television set. The users of the system can also write blog-type comments, and share them with other users.

The system itself is flexible and cheap. It is a stand-alone, plug-and-play type of solution, based on off-the-shelf consumer hardware supplemented by software created in the project (see Fig. 1.4). The system reads data from an electricity meter, channels it through the Internet using the residences broadband connection, and stores the data in a database.

This energy monitoring system will be used to monitor energy usage in Finland in Porvoo's new Skaftkärr housing area (`http://www.skaftkarr.fi/en`), where this system is being set-up in residences, or LivingLabs. Currently we are in the initial testing phase with half a dozen LivingLabs. The residences are inhabited by real families, so that we can get user-centered experiences on the usability of the system we are developing. In the future, energy monitoring will also include district heating and energy produced locally by solar panels.

An example of an instance-informed smart appliance has been implemented for air handling units (AHU) with heat recovery. AHUs usually in-

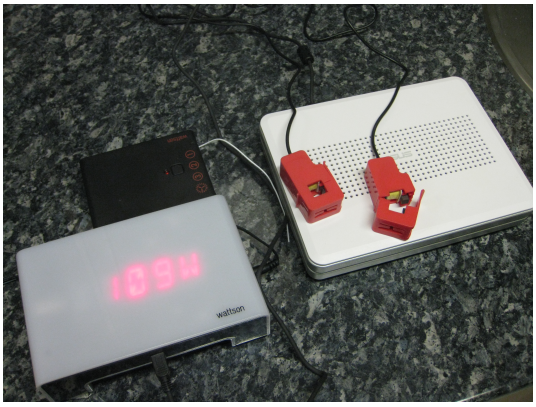

Fig. 1.4. Electricity consumption monitoring hardware.

clude heating and cooling aggregates that allow them to have total control of the base-level inside temperature. AHUs are the most significant piece of equipment regarding the energy efficiency of modern low-energy houses. Such houses are built with thick insulation and with as little air leakage as possible, which signifies that the ventilation system has to ensure that the indoor air remains at a good quality. A good air quality can only be obtained by a sufficient ventilation, which sets high requirements on the heat recovery systems used in AHUs.

Remote monitoring of AHUs provides Green IS benefits such as remote condition monitoring, optimization of maintenance operations etc. In addition to these, it is possible to continuously monitor and adapt the control of the AHU for optimal heat recovery efficiency in all conditions (Fig. 1.5). The optimal control parameters to use in the control IS depend on outdoor and indoor temperatures, humidity and other environmental factors. The dynamics of the building itself and the behavior of other systems in the building also have an influence on how to maintain the best compromise between temperature, humidity, air quality and other factors while keeping the inhabitants satisfied and optimizing the total energy-efficiency.

Instance-informed systems have also been implemented in projects performed together with industrial partners in the automotive domain. In the automotive domain, remote monitoring of vehicles can give substantial savings in maintenance and repair costs, reliability, re-sale value, improvement in driver behaviour and optimisation of fleets of vehicles, such as the airport vehicles monitored in Fig. 1.6. All these benefits are also important from a Green IS point of view.

Especially when an increasing number of vehicles will be powered by electricity, such vehicle fleet management capabilities may also turn out to be crucial for Green IS applications. Benefits can be expected from monitoring the current energy consumption, estimating the remaining kilometers to go

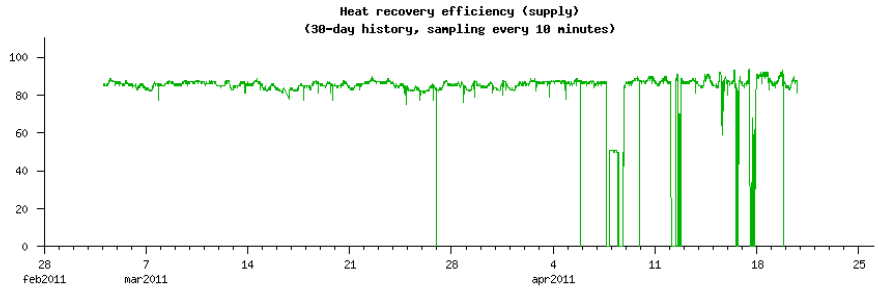

Fig. 1.5. Information collected about AHU efficiency at the end of heating period.

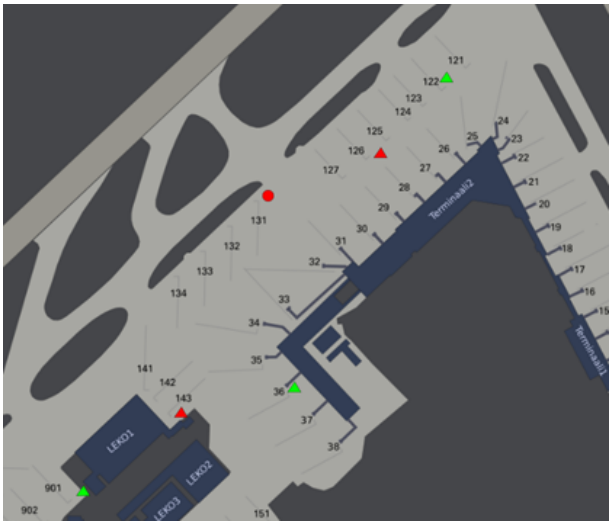

Fig. 1.6. Partial view of airport vehicles, where the shape indicates the type of vehicle and the colors indicate the status of the vehicle, e.g. 'in use', 'engine is on' etc.

with the current charge, as well as when and where to go and re-charge. However, from the Green IS point of view, the fleet management aspect may be even more important because it could give an information in real time about the charging capacity available depending on the location, as well as the total need for charging in the near future. Both the knowledge about the available energy storage capacity and the knowledge about the charging needs might be important for optimizing energy production and usage.

The presented systems show that there is a clear need for managing instance-specific information on the device (AHU and vehicle), aggregate (house) as well as fleet level (fleets of vehicles). In principle, every instance manages its own information. However, in many Smart Grid applications, it

may be necessary to collect information from many different instances. Depending on the application, the needed information may change. This is where the two-layered model provides basic principles for disconnecting the actual information and the way it is structured from the views or interfaces provided to it. When those interfaces are structured according to design pattern principles, it is possible to creat such loosely-coupled applications that are necessary in Smart Grid applications, where different organisations, individuals, products etc. have to interact and act together.

1.4 Conclusion

The requirements on information interoperability and co-ordinated action for Green IS are a challenge for many current information systems that are developed for organisation-specific needs and that are usually not inter-operable. Furthermore, such systems are not designed for storing, managing and processing information about instances such as houses, vehicles, building automation equipment etc.

In this paper, we have shown how instance-informed IS can enable the collection of information from a multitude of heterogeneous information sources that are relevant for energy informatics. We claim that any Green IS will have to be instance-informed in order to enable a holistic control and optimisation of all the elements of the system.

Acknowledgment

This work has been financed by industrial partners, the Finnish Funding Agency for Technology and Innovation (Tekes) and the AsEMo project financed by the European Regional Development Fund.

References

1. R. T. Watson, M.-C. Boudreau, A. Chen, and M. H. Huber, "Green is: Building sustainable business practices," 2008.
2. N. Melville, "Information systems innovation for environmental sustainability," *MIS Quarterly*, vol. 34, no. 1, pp. 1–21, 2010.
3. R. Watson, M. Boudreau, and A. Chen, "Information systems and environmentally sustainable development: Energy informatics and new directions for the is community," *MIS Quarterly*, vol. 34, no. 1, pp. 23–38, 2010.
4. K. Ashton, "Internet things - mit, embedded technology and the next internet revolution," in *Tag 2000, 25.5.2000, Baltic Conventions, The Commonwealth Conference and Events Centre, London.*, 2000.
5. N. Gershenfeld, *When Things Start to Think*. Owl books, 2000.

6. K. Främling, M. Harrison, J. Brusey, and J. Petrow, "Requirements on unique identifiers for managing product lifecycle information - comparison of alternative approaches," *International Journal of Computer Integrated Manufacturing*, vol. 20, no. 7, pp. 715–726, 2007.
7. H.-B. Jun, J. Shin, D. Kiritsis, and P. Xirouchakis, "System architecture for closed-loop plm," *International Journal of Computer Integrated Manufacturing*, vol. 20, no. 7, pp. 684–698, 2007.
8. K. Nygaard and O.-J. Dahl, "The development of the simula languages," *ACM SIGPLAN Notices*, vol. 13, no. 8, pp. 245–272, 1978.
9. A. Goldberg and D. Robson, *Smalltalk-80: The language and its Implementation*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.
10. M. Kärkkäinen, T. Ala-Risku, and K. Främling, "The product centric approach: a solution to supply network information management problems?" *Computers in Industry*, vol. 52, no. 2, pp. 147–159, 2003.
11. M. Rönkkö, M. Kärkkäinen, and J. Holmström, "Benefits of an item-centric enterprise-data model in logistics service: a case study," *Computers in Industry*, vol. 58, no. 8, pp. 814–822, 2007.
12. M. Kärkkäinen, J. Holmström, K. Främling, and K. Artto, "Intelligent products - a step towards a more effective project delivery chain," *Computers in Industry*, vol. 50, no. 2, pp. 141–151, 2003.
13. K. Främling and D. McFarlane, "Editorial for special issue on intelligent products," *Computers in Industry*, vol. 60, no. 3, pp. 135–136, 2009.
14. G. Meyer, K. Främling, and J. Holmström, "Intelligent products: A survey," *Computers in Industry*, vol. 60, no. 3, pp. 137–148, 2009.
15. EPCglobal, "Epc information services (epcis) version 1.0.1 specification," http://www.gs1.org/gsmp/kc/epcglobal/epcis/epcis_1_0_1-standard-20070921.pdf, 2007, [Online; accessed 29-August-2011].
16. J. Parsons and Y. Wand, "Emancipating instances from the tyranny of classes in information modeling," *ACM Transactions on Database Systems*, vol. 25, no. 2, pp. 228–268, 2000.
17. K. Främling, T. Ala-Risku, M. Kärkkäinen, and J. Holmström, "Agent-based model for managing composite product information," *Computers in Industry*, vol. 57, no. 1, 2006.
18. P. Wegner, "Why interaction is more powerful than algorithms," *Communications of the ACM*, vol. 40, no. 5, pp. 80–91, 1997.
19. C. Petrie and C. Bussler, "Service agents and virtual enterprises: A survey," *IEEE Internet Computing*, vol. 7, no. 4, pp. 68–78, 2003.
20. P. Beynon-Davies, "Formated technology and informated action: The nature of information technology," *International Journal of Information Management*, vol. 29, no. 4, pp. 272–282, 2009.
21. M. Wooldridge and N. Jennings, "Intelligent agents: Theory and practice," *Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995.
22. J. Holland, *Hidden order: how adaptation builds complexity*. Addison Wesley Longman Publishing Co., Inc., 1995.
23. D. Luckham, *The Power of Events*. Addison-Wesley, Boston, USA, 2002.
24. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison Wesley, Reading, MA, 1995.
25. K. Främling, T. Ala-Risku, M. Kärkkäinen, and J. Holmström, "Design patterns for managing product life cycle information," *Communications of the ACM*, vol. 50, no. 6, pp. 75–79, 2007.