Integrating Smart Items with Business Processes

An Experience Report

Christof Bornhövd, Tao Lin, Stephan Haller*, Joachim Schaper

SAP Research Center Palo Alto, LLC, 3475 Deer Creek Road, Palo Alto, CA 94304, USA

*SAP Research, CEC Karlsruhe, Vincenz-Priessnitz-Strasse 1, D-76131 Karlsruhe, Germany

Contact email: {christof.bornhoevd, tao.lin, stephan.haller, joachim.schaper}@sap.com

Abstract

Smart item technologies, like RFID and sensor networks, are considered to be the next big step in business process automation [1]. Through automatic and real-time data acquisition, these technologies can benefit a great variety of industries by improving the efficiency of their operations. SAP's Auto-ID infrastructure enables the integration of RFID and sensor technologies with existing business processes. In this paper we give an overview of the existing infrastructure, discuss lessons learned from successful customer pilots, and point out some of the open research issues.

1. Introduction

With RFID mandates from retailers like Wal-Mart, Metro, Tesco, and Target, manufacturers like Procter & Gamble and Kimberly Clark, and even the U.S. Department of Defense, smart item technology has received a lot of attention.

By *smart item* we mean a device that can provide some data about itself or the object it is associated with and that has the ability to communicate this information [7].

For example, a Radio Frequency IDentification (RFID) tag that contains information about the object it is attached to provide a simple form of a smart item [5]. RFID tags typically combine a modest storage capacity with a means of wirelessly communicating stored information like an electronic product code (EPC) [2] to an RFID reader. In a supply chain management context, an object to be tagged is usually a pallet, a case or even a single sales item. Passive RFID tags require no on-board battery and can be read from a distance ranging from a few centimetres to a few meters. Active tags, on the other hand, come with an on-board battery which provides larger read ranges and memory sizes but also higher unit cost and size and a limited lifespan of typically 3-5 years. Another example of a smart item is an environmental sensor, such as a temperature or humidity sensor, which can provide a more complete picture of a tracked object and its physical environment [10].

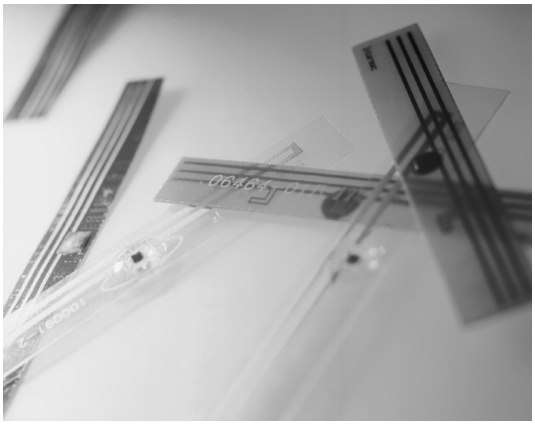

Figure 1 RFID tags

Through automatic, real-time object tracking, smart item technology can provide companies with more accurate data about their business operations in a more timely fashion, as well as help streamlining and automating the operations themselves. This leads to cost reduction and additional business benefits like increased asset visibility, improved responsiveness and even extended business opportunities. However, bridging the gap between the physical and the digital world requires a flexible and scalable system architecture to integrate automatic data acquisition with existing business processes.

Therefore, we have developed the so-called *Auto-ID Infrastructure* (AII), which integrates data from smart item devices with enterprise applications. The AII converts RFID or sensor data into business process information by associating it with specified mapping rules and metadata. These mapping rules can feed incoming observation data directly to business processes running on either SAP or non-SAP backend systems, execute predefined business logic, or simply record the data in a persistent store for later analysis.

The remainder of this paper is organized as follows. In Section 2 we will outline the system requirements that have shaped the design of our Auto-ID infrastructure. Section 3 will give an overview of the existing system architecture and discuss the key components: the Device Controller and the Auto-ID Node. A discussion of our experiences with the existing Auto-ID Infrastructure is given in Section 4. We will conclude by pointing out some of the main open issues in Sections 5, future trends in Section 6 and summarize the paper in Section 7.

2. Auto-ID System Requirements

Our initial Auto-ID Infrastructure has been architected with the following system requirements in mind.

Scalability. Companies like large retailers are assumed to require throughput rates of about 60 billion items per annum [9]. Assuming 100 distribution centers, each with an average of 5 checking points per item, the system needs to guarantee an average throughput of at least 100 messages per second per distribution center. The size of an observation message can be assumed to be around 200 bytes, and the processing of an incoming observation message usually requires multiple database updates and the execution of business procedures at the backend system.

Open System Architecture. In addition to being hardware-agnostic, the architecture should be based on existing communication protocols like TCP/IP and HTTP, as well as syntax and semantics standards like XML, PML [6] and EPC [2]. This will allow the use of sensors from a wide array of hardware providers, and will support the deployment of Auto-ID solutions across institutional or even country boundaries.

Efficient Event Filtering. The infrastructure needs to provide efficient means to filter out false or redundant readings from RFID or sensor devices. Also, it needs to provide flexible and configurable filtering of events to only pass on relevant information to the appropriate backend processes.

Event Aggregation. The infrastructure needs to support the composition of multiple related events to more complex events for further processing. For example, the system must allow the composition of individual object identification events for multiple individual cases and the corresponding pallet to only one *complete-pallet-detected* event.

Flexibility. The infrastructure needs to be adaptable to different business scenarios. Furthermore, the infrastructure needs to provide flexible means at the business logic layer to respond to abnormal situations, like the missing of expected goods or company-internal re-routing of goods. To avoid redundant implementations of the same business rules in different enterprise applications, the infrastructure needs to offer means to deploy and execute them within the Auto-ID Infrastructure.

Distribution of System Functionality. A real deployment of an Auto-ID solution can be distributed across sites, across companies, or even across countries. This naturally requires a distributed system architecture. As a first step, we require that the Auto-ID Infrastructure supports the distribution of message pre-processing functionality (for example, filtering and aggregation) and, to some degree, business logic across multiple nodes to better map to existing company and cross-company structures.

System Administration and Test Support. The infrastructure must provide support for the testing of individual custom components used in the filtering and aggregation of events, as well as the end-to-end processing of RFID and sensor data. Good administration and testing support is a prerequisite for the deployment of a distributed Auto-ID solution in large-scale applications.

3. System Overview

The architecture of our Auto-ID Infrastructure (AII) is shown in Figure 2. Conceptually, it can be divided into the following four system layers.

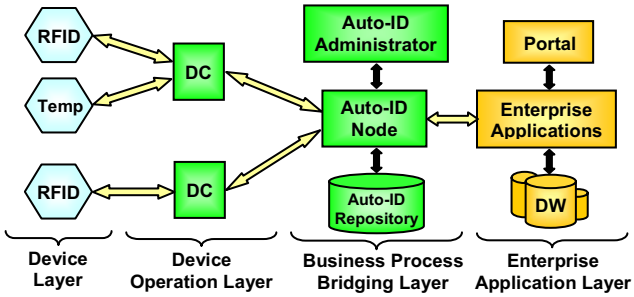

Figure 2: AII System Architecture

At the *Device Layer* different types of sensor devices can be supported via a hardware-independent low-level interface. It consists of the basic operations for reading and writing data and a publish/subscribe interface to report observation events. By implementing this API, different kinds of smart item devices can be deployed within the Auto-ID infrastructure. Besides RFID readers, these devices can include environmental sensors, or PLC devices. The *Device Operation Layer* coordinates multiple devices. It also provides functionality to filter, condense, aggregate, and adjust received sensor data before passing it on to the next layer. This layer is formed by one or more Device Controllers (DC). The *Business Process Bridging Layer* associates incoming observation messages with existing business processes. At this layer status and history information of tracked objects is maintained. This information includes object location, aggregation information, and information about the environment of a tagged object. A so-called Auto-ID Node realizes this functionality. Finally, the *Enterprise Application Layer*

supports business processes of enterprise applications such as Supply Chain Management (SCM), Customer Relationship Management (CRM), or Asset Management running on SAP or non-SAP backend systems.

Our Auto-ID Infrastructure provides an infrastructure for realizing a complete Auto-ID solution. Most existing solutions only focus on a portion of such a complete solution, for example a Savant as defined in [3] corresponds to a Device Controller in our infrastructure. Since Auto-ID solutions can span organizations or even countries, standards for the interfaces between the components are essential. Therefore, the AII is compliant with the standards proposed by the EPCglobal consortium.

As part of the infrastructure, a test and workload generator tool is provided that can simulate messages coming from one or more Device Controllers or backend systems to an Auto-ID Node. Also, a scriptable simulator is available that can simulate multiple RFID readers. These tools allow the testing of an Auto-ID deployment without the installation of physical devices.

The following two subsections will explain the two main building blocks of the AII: the Device Controller and the Auto-ID Node.

3.1 Device Controller

A Device Controller (DC) is responsible for coordinating multiple smart item devices and reporting incoming observation messages to one or more Auto-ID Nodes. A DC supports two operation modes. In the *synchronous mode*, the Device Controller receives messages from an Auto-ID Node for direct device operations, such as to read or write a specific data field from/to a tag currently in the range of an RFID reader, or to read the value from a temperature sensor at a given point in time.

In the *asynchronous listening mode*, the DC waits for incoming event messages from the sensor devices. Upon receiving such a message, additional data can be read and event messages can be filtered or aggregated according to the configuration of the DC. Note that when a DC is configured for asynchronous operations, it is still capable of synchronously receiving and executing commands.

Message processing in the DC is based on so-called *Data Processors*. We distinguish six different types of data processors. (1) *Filters* filter out certain messages according to specified criteria. For example, they can be used to filter out all event messages coming from case tags, or clean out false reads ("data smoothing"). (2) *Enrichers* read additional data from a tag's memory or other device and add this data to the event message received. (3) *Aggregators* can be used to compose multiple incoming events into one higher-level event (for example, mapping data from a temperature sensor to a *temperature-increased* event), or for batching purposes. (4) *Writers* are used to write to or change data on a tag or control an actuator. (5) *Buffers* buffer event messages for later processing and/or keep an inventory of tags currently in the reading scope of an RFID reader. (6) *Senders* transform the internal data structure of the messages to some output format and send them to registered recipients. We currently use PML Core [6] as the output format. As new standards are developed, they can be incorporated by simply implementing appropriate new Senders.

The core functions of the Device Controller, in particular the message processing described above, are independent of the hardware used. For reading and writing the data on the tags, we use logical field names to abstract from concrete tag implementations. A field map provides the mapping between memory addresses on the tag and logical data fields.

Since all Data Processors implement the same publish/subscribe interface, they can be arranged into processing chains. Powerful message processing and filtering operations can be achieved by chaining together the right, possibly customized, set of simple data processors. This results in a very flexible framework which allows for the distribution of message processing functionality close to the actual sensor devices to reduce message traffic and improve system scalability.

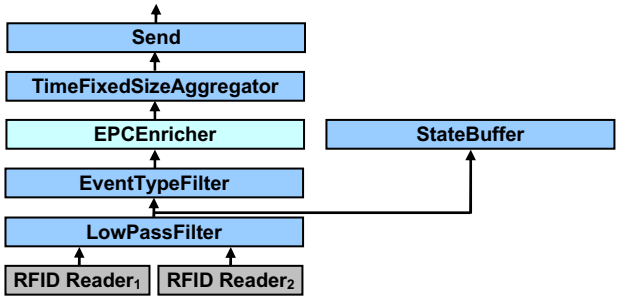

Figure 3: Typical Data Processor Chain

Figure 3 shows an example of a typical processor chain used for dock doors in a supply chain scenario. For full coverage dock doors commonly use more than one reader. This holds especially true for Europe with much stricter radio frequency regulations than in the U.S. RFID readers sometimes generate false event messages. For example, because of physical reasons a tag is not seen during a particular read cycle. To filter out these false *tag-disappeared* messages, a LowPassFilter is applied. Also, every tag that passes the radio field will issue two event messages: a *tag-appeared* and a *tag-disappeared* message. Since in the dock door scenario we are only interested in the fact that an item has passed the door, we can safely filter out *tag-disappeared* messages by using an EventTypeFilter. The EPCEnricher in the example is only needed if non-EPC tags (which are still common today) are used. These tags have a unique ID set by the manufacturer, and the EPC is actually stored in the user memory of the tag. In this case, the EPCEnricher reads the EPC and adds it to the event message. At a dock door, we want to collect all tags that are seen during a certain time window and report them in a single message to the backend system. The TimeFixedSizeAggregator and the Send processor in our example do this. In addition, a

StateBuffer keeps track of all tags currently in the reader's scope for auditing and reporting purposes.

3.2 Auto-ID Node

An Auto-ID Infrastructure can contain multiple Auto-ID Nodes. An Auto-ID Node (AIN) is responsible for integrating incoming observation messages from the Device Controllers with the business processes running at the backend systems.

For an AIN, we distinguish between the interactions with Device Controllers (reader events from and control commands to Device Controllers), and interactions with backend enterprise systems (such as receiving master data from a logistics system and returning a confirmation). These interactions with the AIN are treated as either incoming or outgoing messages.

Incoming observation messages are routed to a rule engine which, based on the message type, evaluates a specified list of conditions. The result of the evaluation step is a set of qualifying rules for which one or more actions are executed in a specified order. Such an action can, for example, update the system status of an object in the local repository, communicate with the backend system, or generate and write EPC data to a tag.

Actions of a rule can pass on parameters and can trigger other rules at the Auto-ID Node. Based on the message type, messages can be assigned different processing priorities and can be specified as being persistent in the Auto-ID Node.

An Auto-ID Node provides a local repository which contains information about the current status and history of the objects being processed. This information includes data about the operations that have been applied to an object (e.g, move, pack, or unpack), its movement and current location, and its structure (e.g, packing information). Also, the repository replicates master data from the backend system about products and business partners, or the physical location and type of the RFID readers. The Auto-ID Repository provides the basis for the execution of business logic in the Auto-ID Node.

The use of customizable rules provides a flexible mechanism to specify and execute business logic at the Auto-ID Node. This allows the pre-processing of incoming observation data and the handling of abnormal situations within the Auto-ID Infrastructure, such as discrepancies between a received advanced shipping notification (ASN) and a detected pallet. Which in turn allows the system to offload processing from the backend systems.

Our work to date has focused mainly on Supply Chain Management scenarios, for which a standard set of rules is in place. The deployment of our Auto-ID Infrastructure in a different context simply requires the adoption or extension of the existing rules.

The Auto-ID Administrator provides a graphical tool which supports the reconfiguration of existing or the definition of new rules in an AIN at run-time. In addition, it allows the central configuration, monitoring, and control of the Device Controllers and smart item devices in the system.

4. Case Studies

The following sub-sections discuss two Auto-ID pilot installations based on the research prototype described in Section 3: a real-time retail application, and an adaptive planning application. In section 4.3 we will summarize the lessons learned from these and other real-world experiences.

4.1 A Retail Application

The first pilot was conducted at a large retailer in Europe. Here, the Auto-ID Node was used as a kind of "Auto-ID data hub" to feed business event information from several processes to two backend systems: a data warehouse (SAP Business Information Warehouse) for analytical purposes, and a tracking system (SAP Event Management) to track the status of deliveries. On top of this, the SAP Enterprise Portal was used as the user interface to provide both employees and project partners with a unified view on the entire system.

From the perspective of the retailer, the goal of this project was mainly to evaluate if and how RFID technology can be used in practice. While there is a lot of hype about the technology, only by putting it in a real environment one can learn what works and what does not, and possibly how to get around technical difficulties. In addition to technical issues, another question was how customers would accept the technology.

The main process covered by RFID technology was the tracking of deliveries from the distribution center to one dedicated store, as well as the movement of goods from the store's back room to the shop floor. Tagging was done on case and pallet level. There were four read points in this business process:

1. ***Packing Station***: At the distribution center, all cases needed to be tagged and assembled into deliveries. An association between the pallet and the cases loaded onto it was recorded. Once the packing was finished, a message was sent to the Auto-ID Node with information about the pallet and its associated cases.
2. ***Goods Issue Gate***: After the deliveries were loaded onto a truck at the distribution center, they passed through a reader that registered what had passed. The reader was mounted in the dock door. The data from the reader was filtered, aggregated and then sent to the Auto-ID Node, which updated its inventory of goods.
3. ***Goods Receiving Gate***: Similar to the previous read point, incoming goods were read and recognized as they arrived at the store.
4. ***Back room / Shop Floor Gate***: This represents another automatic gate where goods were scanned

when they passed through to determine if goods were in the back room or already on the shop floor. Previously, the retailer was not able to make this distinction.

Bulk reading took place at all read points except at point 1. Depending on the nature of the products (metal cans, bottles with soda water, and so on) in the delivery, it was impossible to always read all tags of inner cases. However, since the pallets were not unpacked until they were on the shop floor, it proved to be unnecessary to achieve a read accuracy of 100%. Since the system provided the information about what was packed together from read point 1 (where the operational process guarantees a 100% read accuracy) in principle it was sufficient to only detect a single tag of the whole delivery at the other read points. The Auto-ID Node was able to deduce the other tags from the packing information.

The Auto-ID Node received observation messages from all read points to update its inventory information in its repository. Information was also sent to a data warehouse to allow for later analysis. The AIN stored information about the deliveries (actual and expected), so whenever it received a message from one of the first three read points, it inferred the delivery number from the EPCs detected. It could then tell the Event Manager System, which was responsible for tracking deliveries overall, about the status change of the delivery, for example, that it had arrived at the shop.

RFID technology was also used on the item level for some distinct goods. For example, processed cheese was tagged in order to track expiration dates using a Smart Shelf. A Smart Shelf is basically an array of RF antennas that can identify which goods are on the shelf an e..g. if goods have been misplaced. This allows to allow a very fine grained tracking of goods, which specifically through-out campaigns is important for the business operation. This pilot implementation showed that item level tagging is technically feasible, but that the cost of tags themselves, of applying the tags to products, and of the required infrastructure (readers and so on) is currently still too high for most products to make sense economically. Another reason against tagging at the item level is concerns in public regarding privacy. However for certain expensive goods and in conjunction with theft detection individual tagging is already in use.

At the time of the project, EPC tag data standards as currently defined by EPCglobal [4] have not been developed. However, user requirements required standard identifiers encoded on the tag. Cases needed to have a GTIN (Global Trade Identification Number) of the product, and pallets either a SSCC (Serial Shipping Container Code) or a GRAI (Global Returnable Asset Identifier). We therefore had to define our own mapping of these standard identifiers to EPCs, adding a serial number in the case of the GTIN mapping. Our mapping was based on [2].

The main benefit provided by RFID technology in this pilot was increased visibility of the goods, which could be used to make better decisions on when to reorder goods, leading to cost reductions because of lower inventory levels and increased sales because of increased on-shelf availability.

The software worked reliably. Because of the size of the pilot, scalability was not an issue and a single Auto-ID Node was sufficient. More daunting were the challenges regarding the hardware, like positioning and tuning the reader antennas to achieve good read accuracy while conforming to regulatory requirements in Europe, tag placement, cabling, and availability of tags, just to name a few. Workplace safety regulations added additional constraints.

4.2 A Real-time Adaptive Planning Application

The second pilot involved a large retailer and a manufacturer in North America. In this pilot, SAP provided the same components as in the pilot described in Section 4.1, plus a supply chain planning component (SAP Advanced Planning and Optimization). This pilot included three sites: a distribution center of the manufacturer, a distribution center of the retailer, and a retail shop. The main operational process consisted of three steps.

First, in the distribution center of the manufacturer, items were packed into cases and shipped to the distribution center of the retailer based on shipment orders. In the second step, the distribution center of the retailer verified the shipment on the case level and then sent a case to the retail shop. Finally, in the retail shop, the case was first placed in the backroom and then moved to the shop floor. The items contained in the case were put on a smart shelf in the shop. The following read points were defined:

1. ***Pack Station at the Manufacturer:*** After packing a case, a message with all the EPCs of the case and the contained items was sent to the Auto-ID Node. The Auto-ID Node forwarded the EPC of the case and associated shipment order to the tracking system (SAP Event Management), where an Event Handler was created with the expected shipment time, a tolerance for the shipping time and rules for exception handling — that is, what to do when a shipment has not arrived in time.
2. ***Goods Receiving Gates at the Retailer***: There were similar gates both at the distribution center and at the shop. When these gates detected a case tag, messages with the detected case tags were sent to the Auto-ID Node. After updating the status of the associated locations and the physical objects, the Auto-ID Node sent a message to the tracking system to update the status of the corresponding Event Handler.
3. ***Back Room / Shop Floor Gate***: These read points were similar to the read points at the receiving gates of the retailer.

4. ***Smart Shelves in the Shop***: When items were added or removed from the smart shelf, messages containing the EPC of the moved objects were sent to the Auto-ID Node with the logical reader ID and the timestamp of when the objects were scanned. The AIN then forwarded the observation message for the first item from a case that appeared on the smart shelf to the tracking system to indicate that the contents of a case had been put onto the shelf and that the tracking process for that case was completed.

In this pilot, a shipment was associated with a single case as only one case was sent from the manufacturer to the retailer at a time. The Auto-ID Node maintained the status and also the history of the objects including cases, items and shipments. The tracking system was used to track all shipments. Therefore, only messages on the case level were sent to the tracking system, which monitored the delivery of shipments and handled possible exceptions in almost real-time.

Through the Auto-ID Node, the manufacturer could get inventory information about its products in the retail shop. Based on the history of sale records, the Auto-ID Node maintained a local prediction model. This model could be used to trigger a request to the SAP Advanced Planning and Optimization to adjust the shipment planning.

SAP Business Information Warehouse was used for analytical operations and reporting, in a similar way as in the pilot discussed in Section 4.1.

4.3 Lessons Learned

Our experiences with the pilots described in the previous sections can be summarized by the following lessons learned.

Cross-Organizational Collaboration. The pilots contained multiple sites, and in the case of the second pilot even multiple companies. The full potential of smart items technology can only be unlocked through collaboration and data sharing across sites and organizations. The hope is that the potential business improvements offered by Auto-ID technology can bring companies to overcome their current reluctance to collaborate in the near future. This reluctance, as well as technical integration challenges, are the main reasons why EDI has not been implemented to the extent initially expected.

Standards. We found that one of the key issues is the use of common standards. To avoid integration nightmares, standards on the hardware layer (readers, tags), the communication layer (HTTP, XML), and also on the syntax and semantics layer (PML, EPC) should be used or must be developed. Deployment of components from different providers becomes feasible at a reasonable cost of ownership only with the right standards in place. We are actively involved in ongoing standardization efforts at EPCglobal and the W3C.

Automatic Identification is Not Just RFID. The main use case of smart items today is the universal unique identification of items. RFID is not the only technology that allows this, for example barcodes or data matrix codes can be used as well. Different technologies have different advantages and use cases. Thus, all of these must be easily to integrate into one system. Furthermore, in a real working environment RFID readers sometimes need to work with other devices such as conveyor belts and light beam sensors. These heterogeneities are the rule not the exception.

100% RFID Reading Accuracy Cannot be Expected. Because of physical reasons, one cannot expect to have a 100% tag reading accuracy. As described in Section 4.1, one way to work around this problem is to keep information about how objects are assembled and have the Auto-ID Infrastructure infer the missing information. For example, detecting the movement of a pallet known in the system will allow the system to infer the movement of all associated cases.

Need to Support Out of Sequence Messages. To an Auto-ID Node, the connected Device Controllers form a distributed environment. In a real-world installation, network latency, different system clocks at the readers, and message batching all can cause the order in which observation messages arrive to be different from the order in which the corresponding events took place in the physical world. Therefore, the Auto-ID Infrastructure needs to be able to reorder incoming event messages based on knowledge about the physical structure and the business processes of a given site.

Device Administration and Management. The deployment of an Auto-ID solution usually includes a large number of RFID and sensor devices. Centralized administration tools to visualize, plan (capacity planning), configure, deploy, test, monitor, and upgrade remote devices is a prerequisite for the deployment of large, highly distributed Auto-ID solutions. Our existing tools are a good a starting point but more powerful tools are needed.

Deploying an Auto-ID Solution is a Long Term Task. The deployment of an Auto-ID solution will change the IT infrastructure, the business processes and the operational processes of an organization. These fundamental changes cannot be done in a few weeks and may result in significant costs up front. It is essential for a company to have a long term migration plan addressing the required changes in the organization. Therefore, it is a good idea to start with a small pilot installation to learn about the required changes in an existing business environment before rolling out an Auto-ID solution on a large scale.

5. Open Issues

Based on our experiences with our existing prototype, we would like to point out the following open issues for future research in the area of smart items technology.

Different Qualities of Service. Different smart items applications require different qualities of service regarding event processing. For example, for high data quality an Auto-ID infrastructure may have to provide end-to-end transaction support to guarantee exactly once semantics for the processing of observation messages. That is, the system needs to guarantee that a predefined reaction to an event is executed exactly once — even in the case of a system or power failure. There is obviously a trade-off between higher degrees of reliability on the one hand and performance on the other. Accordingly, different qualities of service need to be defined and provided for different application classes.

Distributed Smart Items Infrastructure. The nature of smart items applications as well as scalability requirements may force a distributed system architecture. Although our existing Auto-ID Infrastructure allows the distribution of functionality between Device Controllers, Auto-ID Nodes, and backend systems, a full-fledged solution to the distribution problem needs to support the distribution and replication of functionality *and data*, requiring the sharing and synchronization of data across multiple nodes. The evaluation and adaptation of distribution and replication strategies developed in distributed database systems, database caching, distributed event-based systems, and peer-to-peer systems could be a good starting point.

Seamless Integration of Environmental Sensors. Currently, most work in the area of smart items has focused on RFID and Supply Chain Management. To support application scenarios like product life-cycle management (PLM) or transportation, we need to seamlessly integrate other sensors like environmental sensors with RFID technology. From the application perspective, RFID readers and environmental sensors like temperature or light sensors simply provide event sources. From the perspective of the infrastructure, however, they are different. RFID readers are discrete event sources, whereas environmental sensors provide a stream of periodic events, that is, discrete readings of the corresponding environmental conditions. Conceptually such a sensor provides a current value for each point in time. The seamless integration of RFID and environmental sensors requires means to represent and resolve this mismatch.

Networked Embedded Systems. Smart items provide small embedded systems capable of independently collecting information from their environment, processing data, and communicating over wireless networks. With advances in memory capacity and processing power, these devices allow the execution of business logic at the periphery of a smart items infrastructure rather than in the middle layers or in a central backend system. Smart items can form entire networks of collaborating devices thereby increasing reliability (through replication), efficiency, and flexibility. In addition to the question for new appropriate system architectures, efficient ways are required to model, generate, deploy, and manage business functions at the devices. Here approaches developed in the area of grid and peer-to-peer computing could be a good starting point for further research.

Privacy. The use of RFID technology, especially in retail, has raised a lot of discussion regarding privacy. The main concerns here are the possible profiling of customer behaviour and the potential to track people. Although this discussion is not a purely technical one, on the technical side mechanisms are required that enable the efficient encoding of tag and sensor information, ensure data security, and allow the disabling of tags at predefined stages in a retail chain. The resulting technology needs to be an integral part of a sophisticated smart items infrastructure.

6. Future Application Research

Beyond the current scope of CPG - Consumer Packaged Goods, retail and supply chain management, the advances of Auto-ID will trigger a variety of domains such as PLM - Product Lifecycle Management, asset tracking and spare parts management, office management as well as to support complex manufacturing processes. The advent of so called “sensor networks” as currently developed e.g. by the University of Berkeley (Motes)

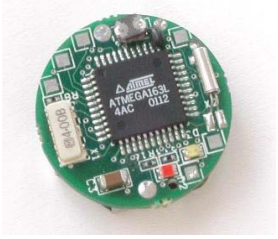

Figure 4 Mote

or the Technical University Karlsruhe (Smart-ITs) will allow more autonomous and sophisticated sensing of data from the environment. This will enable new business applications e.g. within the area of security and trust to protect the integrity of containers or goods.

7. Summary

We have described our Auto-ID Infrastructure which was architected with scalability, flexibility, and usability in mind. Device Controllers allow the processing of event messages close to the periphery of the system; Auto-ID Nodes enable the execution of business logic in the infrastructure and integrate incoming observation messages with backend business processes. We have discussed our practical experiences with different pilot

projects and summarized the main lessons learned. Smart item technology is very likely to change current business and operational processes, which will require changes in the IT infrastructure of many companies. Challenging issues remain that make this area an interesting topic for both hardware and software research.

Acknowledgements

We would like to thank Brian Mo, Uwe Kubach, Rama Gurram, Peter Ebert, and Hartmut Vogler for their valuable contributions to the Auto-ID Infrastructure project. We also benefited from fruitful discussions with other colleagues in SAP including Bernd Sieren, Bernd Lauterbach, Christoph Lessmoellmann, Ami Heitner, Alexander Renz, and Kai Morisse.

REFERENCES

[1] Alexander, K.; Gillian, T.; Gramling, K.; Kindy, M.; Moogimane, D.; Schultz, M.; Woods, M.*: "IBM Business Consulting Services – Focus on the Supply Chain: Applying Auto-ID within the Distribution Center"*, Auto-ID Center, White paper IBM-AUTOID-BC-002, Sep. 2003

[2] Brock, D.L: *"Integrating the Electronic Product Code (EPC) and the Global Trade Number (GTIN)*", Auto-ID Center, White Paper MIT-AUTOID-WH-004, Nov. 2001

[3] Clark, S.; Traub, K.; Anarkat, D.; Osinski, T.: *"Auto-ID Savant Specification 1.0"*, Auto-ID Center, White Paper MIT-AUTOID-TM-003, Sep. 2003

[4] EPCGlobal: *"EPC Tag Data Standards Version 1.1 Rev. 1.24"*, EPCGlobal, Standards Specification, Apr. 2004, http://www.epcglobalinc.org

[5] Finkenzeller, K.: *"RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification"*, John Wiley & Sons, 2nd Edition, May 2003

[6] Floerkemeier, C.; Anarkat, D.; Osinski, T.; Harrison, M.: *"PML Core Specification 1.0"*, Auto-ID Center Recommendation, Sep. 2003

[7] Haller, S.; Hodges, S.: *"The Need for a Universal Smart Sensor Network"*, Auto-ID Center, White Paper CAM-AUTOID-WH-007, Nov. 2002

[8] Kubach, U.: *"Integration von Smart Items in Enterprise Software Systeme"*, In: Praxis der Wirtschaftsinformatik, Special Issue on Ubiquitous Computing, 2003

[9] Miles, S.; Brock, D.L.; Engels, D.: *"Web Services WAN SIG: Proposals for Engineering the 'Silk Road of the Internet'"*, Auto-ID Center, White Paper MIT-AUTOID-WH-04, Apr. 2003

[10] Thede, A.; Schmidt, A.; Merz, C: "*Integration of Goods Delivery Supervision into E-Commerce Supply Chains"*, Second International Workshop on Electronic Commerce (WELCOM'01), Heidelberg, Germany, Nov. 2001