# Enabling Data Collections for Open-Loop Applications in the Internet of Things

Alexander Kröner
Georg Simon Ohm University of Applied Sciences
Nuremberg, Germany
Alexander.Kroener@th-nuernberg.de

Jens Haupert, Matthieu Deru, Simon Bergweiler, Christian Hauck
German Research Center for Artificial Intelligence
Saarbrücken, Germany
{jens.haupert, matthieu.deru, simon.bergweiler, christian.hauck}@dfki.de

*Abstract*—**Using label technology, a physical object may be employed to build a continuously growing data collection, which can, for instance, be exploited for product quality monitoring and supply chain management. Along the object's life-cycle, queries to such a collection may stay quite similar, e.g., "get unusual observations". However, expectations to a "good" answer may change, as with time different entities will come into contact with the object. This article reports on work in progress concerning a framework for collecting data about things, which aims at decoupling logic employed for interpreting such a collection from processing hardware and using the collection itself for transporting such logic. Main contributions include an approach to hardware-abstraction of processing logic at the object or remote, an app store for retrieving interpretation and presentation logic, and interaction forms with such memories.**

***Keywords-Ubiquitous computing; RFID tags; Distributed information systems; Supply chain management.***

## I. Introduction

Within the Internet of Things, physical objects may function as a focus for digital data and services concerning the artifact itself as well as associated things, people and processes as presented in [1] at UBICOMM'13. This function enables an object to take a new role as a data collector and provider in a broad range of scenarios, e.g., users may manually associate data with an object in order to socialize and foster discussion in a community [2], tools may automatically collect usage data in support of pay-by-use accounting [3], and products may steer and document their production [4] and transport rules that support reasoning of healthcare applications [5]. Existing applications of such technology are typically deployed in "closed" scenarios, i.e., requirements of users and applications are known before the collection process starts.

This reflects only to some extent a supply chain with continuously changing users and requirements. In order to facilitate communication between stakeholders in such an "open" scenario, a uniform interaction behavior of the collection would be advantageous, e.g., a uniform way to "check integrity" of an object, i.e., compliance to criteria for objects or kinds of objects specified by a third party on an individual base.

In the following, Section II provides an example scenario, where one stakeholder has to employ logics provided by another stakeholder. Section III summarizes requirements that arise from this scenario. Then, Section IV wraps up work accomplished so far concerning so-called Active Digital Object Memories (ADOMe), a framework for processing logic in a way that allows for embracing a broad range of infrastructure approaches common to Internet of Things applications. Section V extends this approach with a concept of an app store supporting distribution of the processing logic. Section VI deals with approaches to support user interaction with access to object memories - by the framework itself as well as by mobile devices and smart objects. Finally, the article concludes with a summary of results and a discussion of future work in Section VII.

## II. Scenario

The following logistics scenario deals with integrity control during transportation of a heterogeneous set of goods (see Figure 1). Each good is packaged in a way matching its nature (e.g., fragility) and value. All packages are tagged with some kind of label technology, which allows for automatically identifying the object. Depending on the respective kind of package, this technology may range from passive RFID (Radio Frequency IDentification) to embedded systems with integrated sensing and processing capabilities. At the same time an object memory was created and filled with static product and manufacturer data, as well as criteria to be monitored in the following. Additionally, the memory can contain information what sensor values are interesting to the object and orders how to treat and transport the object.

A retail chain advertises the quality of products sold in its stores, which is subject of the company's own, particular strong quality guideline. In order to leverage compliance to this guideline, the company provides business partners along the supply chain with constraints on parameters that need to be monitored. A supplier uses these parameters in order to configure an integrity test for each package destined for this particular retailer; for accuracy, input parameters should be sensed and processed by the package itself or by IT infrastructure near the object. Performing this configuration task is supported by a hardware-abstraction layer, which allows for assigning tests to packages independent from the kind of label technology provided by the respective package.

During loading a truck (by means of this layer), a dialog between package, truck, and an app store, hosting implementations of tests matching the retailer's parameters, is performed. Result of this dialog is an assignment determining which technical component (truck or package) has to conduct the monitoring task, an assignment, which may differ for each package.

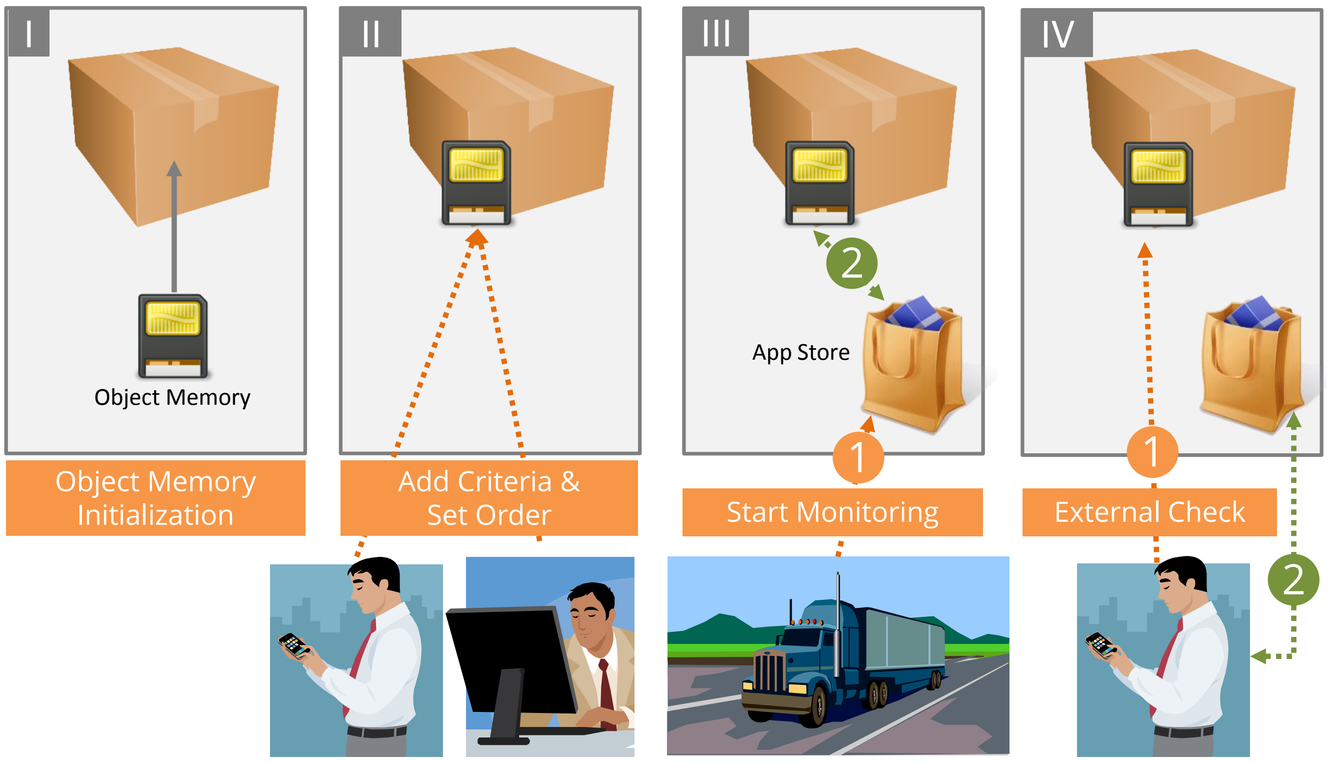

Figure 1. Logistics scenario with memory initialization [I], on-product criteria storage and monitoring orders [II], active logic processing [III], and external checks [IV].

Finally, the packages arrive at the retailer's receiving area. There, an employee uses a mobile device in order to identify the respective object and access its digital records - values sensed during transport as well as testing methods and their results. The access is performed using an app, which downloads from the app store a method suited to visualize the test chosen by the logistics expert. This visual adaptation is performed automatically in the background; even for very different kinds of packages the employee experiences always the same way of interacting with the respective object.

At the retailer's store some products and their new capabilities can be utilized for direct product-to-customer interaction. E.g., a milk carton (called "Milky") instrumented with an embedded controller, sensors and a display is present on a shop shelf (see Figure 3). The milk carton tries to catch the attention of customers with blinking eyes and an acoustic feedback. The proximity sensor and accelerometer sensor detect the fact that the customer is going to take the product from the shelf. Once bought, milky activates a new mode as the product now belongs to the customer. At this moment the sensor data tracking module is activated and all parameters are tracked and stored in the object's memory. In the same manner the customer will be also informed when the best-before day is up. The consumer can also switch between two views, the first one being the face of Milky and the second one showing a more factual view by displaying the sensor's raw data values. Depending on the content of the milk carton a personalization module can be started; strawberry milk could be presented as a pink face on the display. Once the product is no more consumable, e.g., due to an expiry of the best-before day or because the customer has completely used it, Milky goes into the "dead"-state. In this mode three options are displayed on the screen: the first one displays a map with the nearest recycling stations to ensure correct recycling for optimizing the product's carbon footprint. The second option allows the customer to share his "product-experience" over a social network. With the third and last option the product asks the consumer if he would like to buy the same product again.

Extending the mentioned fixed stakeholder chain, a more flexible approach is currently emerging. The so called "open-loop" life-cycle chain is determined by the idea of supporting different successors in each life-cycle step. This approach allows for a flexible "routing" of products by incorporating different stakeholders and delaying the process of choosing which stakeholder is next from design time to runtime (see Figure 2). This approach demands flexible systems that support various hardware platforms, diverse device capabilities and different data content.

Considering such open-loop supply chains, our scenario can be enhanced. Let's assume our known logistics partner equips each individual object with a dedicated embedded system to perform the monitoring tasks. By adding additional providers we also have to support their approach (see Figure 4). E.g., some providers equip the products only with RFID tags with server-based storage and perform the monitoring only on pallet- or container-level, whereas others attach an embedded controller to each product. Such controllers provide storage, processing and sensor capabilities. The complete data set is created and stored locally. And other providers even do not support direct monitoring of individual products during transport at all. Depending on the providers involved, a retailer might receive memories equipped with different hardware

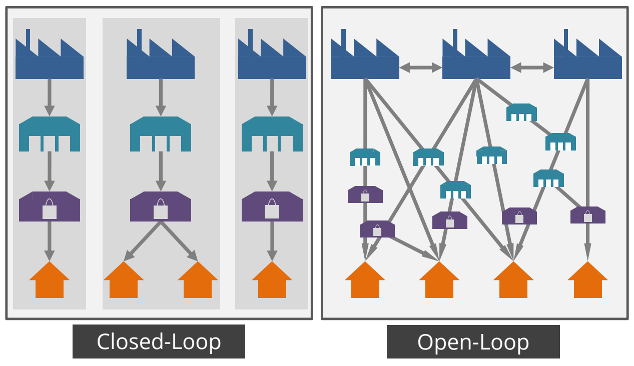

Figure 2. Supply chain: ”closed-loop” (left) and ”open-loop” (right) paradigm.

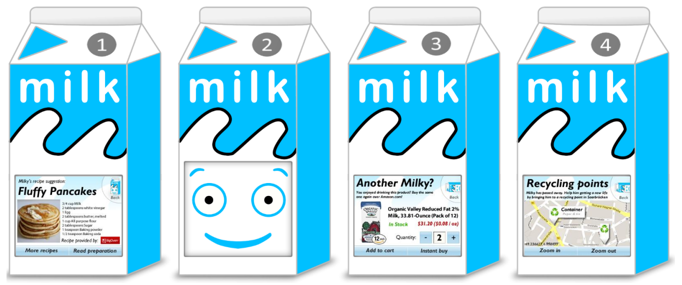

Figure 3. Smart milk carton ’Milky’ showing object-related recipes (1), in anthropomorphic mode ’happy’ (2), advertising on-product re-buy task (3), and presenting recycling information (4).

platforms and filled with data of different quality and quantity.

Summarizing, the described scenario is characterized by the following key features:

- Varying stakeholders
- Varying capturing technology
- Varying interaction devices
- Varying interaction processes
- Reconfiguration at any time throughout the process

## III. RELATED WORK

This work is related to research and development concerning frameworks that leverage collecting and processing data related to physical objects, and as such related to the Internet of Things. Related research comprises embedded systems as well as web-based data stores. So-called Collaborative Business Items (CoBIs) illustrate the benefits of delegating small parts of a well-defined business process - e.g., monitoring and self-assessment tasks - to objects with embedded sensing and processing capabilities [6].

In order to decouple such a service from the employed hardware, SmartProducts [7] seek to dynamically integrate resources - including web-based structures - in the object’s environment into the service realization. Complementary to our proposal, this work puts particular emphasis on semantic device and data descriptions for products with embedded technology.

Other projects also cover innovative solutions for the logistics domain. European projects EURIDICE [8] and iCargo

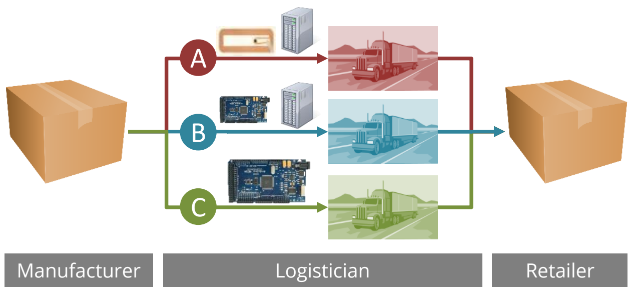

Figure 4. Extended supply chain based with three difference logisticans.

[9] build information services platforms centered around the individual object to increase the interaction, the exchange and the organization of data to increase efficency and to reduce the carbon footprint. The approach introduced in this paper targets partially similar goals, but will utilize techniques that can cover the entire life-cycle chain.

An example of collecting object-related data in a web-based data store is the Tales of Things electronic Memories (TOTeM) system. It seeks to foster communication between humans via personal stories digitally linked with things [2]. Its infrastructure shares aspects of an ADOMe, in particular a unified approach for structuring data concerning a thing, and open web-based information storage. The human-computer-interaction is performed by a web-based application on mobile devices.

Similar applications focus on connecting people with objects (e.g., like Anythinx [10], or creating object-centric data collections for personal use (e.g., like Qipp [11]). They all share a user interface, which allows for accessing collections from a desktop as well as on-the-way from a smartphone; data creation is left to the user.

Going beyond, EVRYTHNG [12] extends this general approach with Active Digital Identities for objects, where services linked with an object employ information collections (concerning the object, or objects of the same kind) in order to adapt to the user. The web-based system can be accessed either by a desktop computer or a mobile device.

The question of how implementation and provision of such services can be supported is addressed by Xively [13]. The web-based service supports not only hosting and sharing object data, but also software products and descriptions concerning devices, which we propose to extend in a way that supports exchanging such components across devices using unified data structures and semantic descriptions.

A data structure for representing object memories in open loop scenarios similar to the one mentioned in this article has to meet particular requirements. These are addressed by the so-called *Object Memory Model* (OMM), created by the W3C Object Memory Model Incubator Group (OMM-XG) and co-developed by the authors. The model partitions the memory content into several blocks, each with content of the same origin or nature (see Figure 5) [14]. Each block consists of two parts: the payload representing the content itself and a set of corresponding meta data defining and describing the payload. This set of machine-readable annotations eases the process of searching data inside object memories that

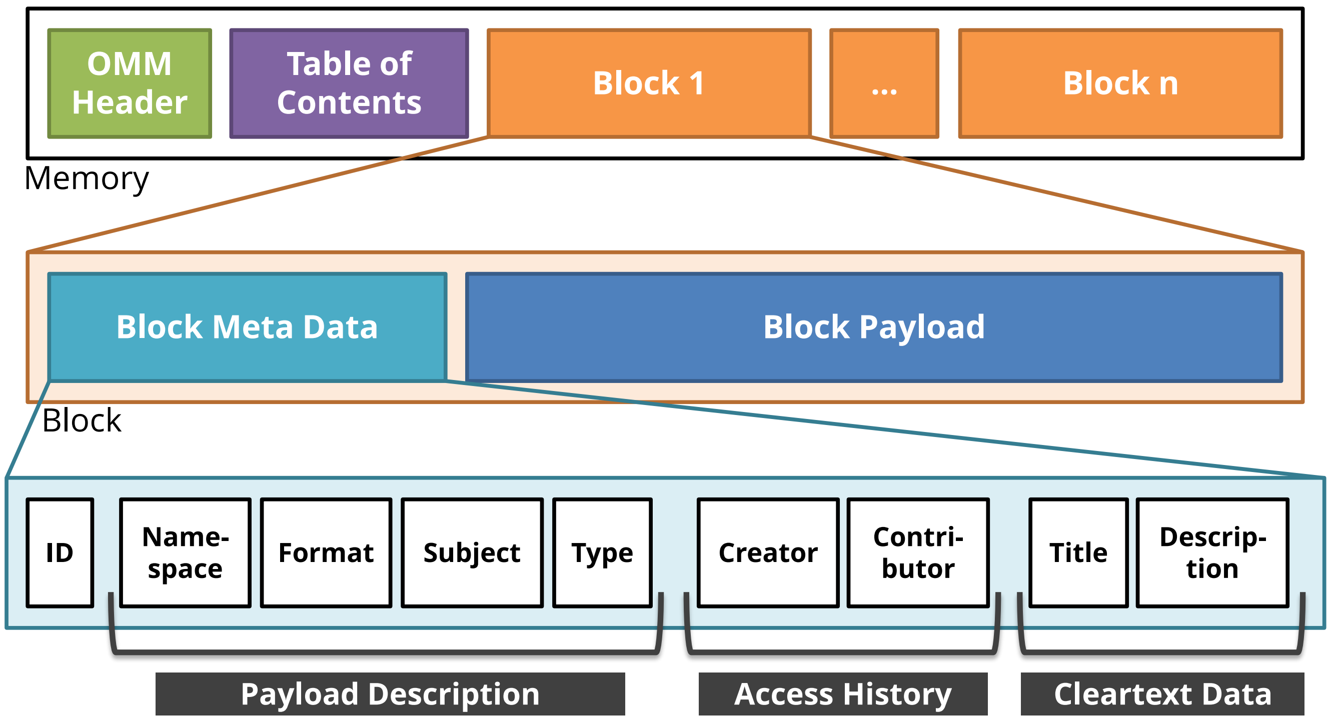

Figure 5. Sample memory based on the Object Memory Model (OMM) with detailed metadata.

contain heterogeneous data and are often not known in advance. Additionally, such memories demand new interaction forms considering machine-to-machine communication (M2M) and human-computer-interaction (HCI) [15].

## IV. The ADOMe Framework

The envisioned framework has to balance between the need for flexibility (to support different hardware platforms, open processes, new requirements in the future), for standardized structures that ease data retrieval and communication among different and cross-domain content providers and for normative description of object-related logic. Corresponding goals to the architecture model can be divided in three parts that reflect a 3-tier approach to the realization of such a model, namely the collection model for data storage (1), support for active analysis functionality (2), and a common access architecture and infrastructure support for hardware abstraction with an app-store-like approach (3).

Data storage is the basic functionality of any ADOMe. In order to enable a stakeholder to analyze data added to the memory by another one, a common storage model is achieved, which is shared by all parties along the object's life-cycle. In addition, the model should be flexible enough to cover a large variety of data formats (including encodings and further data types) and should ease the process of data retrieval. Due to the cross-domain usage of such memories involving several partners, a common infrastructure that provides an abstract memory access (including protocol and data exchange specifications) independent from any memory implementation or hardware platform is necessary to ease the task of memory access for existing and newly created applications. This hardware abstraction layer enables a transparent access for clients, which includes a compensation of missing object features by the environment. Setting up activity and analysis support on top of the data storage can extend the functionalities of object memories, by allowing the memory to process data autonomously based on given algorithms. Based on this functionality we want to enable applications to ask common (but pre-defined) semantic questions, rather than processing the entire memory data, which might be a complicated process due to possibly very large and capacious memories and in contrast a slow connection speed. In addition, the memory should pro-actively process data with rules based on expert knowledge, and deploy results to memory storage or return the result on queries. Finally, a mechanism to retrieve machine and platform-compatible logic code for the given use case is needed to support the mentioned hardware abstraction.

### A. *Object Identification and Data Model*

To identify each physical object a unique ID is necessary that goes along with the object during the entire life-cycle chain and represents the corresponding object memory. Our framework uses a unique *Uniform Resource Locator* (URL) [16], [17]. This approach has the advantage that URLs can be used to easily create unique identifiers and to directly indicate the type of memory access (web-based via a http-connection). This URL can be attached to physical objects in different ways. The options range from simple machine-readable 1D- or 2D-codes (e.g., barcodes, DataMatrix Codes or QR Codes) [18]–[20] to more sophisticated wireless solutions like Radio Frequency Identification (RFID) [21] and Near Field Communication (NFC) [22], or the well-known standards like Bluetooth [23] and Wi-Fi [24]. Both approaches can be easily accessed with the help of a smartphone or tablet.

The foundation of the framework is the storage of object-related data. To structure this data a data model covering the requirements of open-loop scenarios is necessary. In our approach we use the aforementioned Object Memory Model (OMM) for structuring the memory content in blocks composed of the payload and additional meta data (see Figure 5). In the following, we will describe the set of meta data in detail.

The first attribute, called *ID*, is a unique identification for each block represented as string. The next four attributes are intended for machine-to-machine (M2M) communication purposes. The *Namespace* is represented as URN and can be used to indicate that the payload has a defined content and a standardized type (e.g., ”`urn:objectdata:pml`”). This allows for direct access to the payload, if the reader supports the namespace. *Format* is just the MIME-Type of the payload. *Subject* contains a tag cloud like structure to annotate the block payload with free text tags (e.g., ”manual”), hierarchical text tags with a point as delimiter (e.g., ”norms.din.a4”) and ontology concepts (e.g., ”http://s.org/o.owl#Color”). The *Type* attribute is a Dublin Core DCMI Type Vocabulary Type [25]. The following two attributes create a modification history for this block. *Creator* is a tuple of the entity that created the block and a corresponding timestamp. In addition, *Contributor* is a list of tuples with entities that change the block and the corresponding timestamps. Finally, the last two attributes are rather intended for human-computer-interactions (HCI). *Title* contains a human readable short title for this block and *Description* contains a longer textual description, both with support for multiple languages.

In case that the payload has a very large size and does not fit into the memory (e.g., located in an embedded system) or the data is redundant and used in many similar memories, the payload can be out-sourced. This is done by an additional *Link* attribute that indicated the source of the payload (e.g., in the World Wide Web).

Furthermore, each memory contains a header that includes the unique ID of this memory and an optional list of links to additional blocks (e.g., that are out-sourced due to space restrictions) and a table of contents (ToC) that provides the meta data information from all blocks to enable applications to get an overview of the memory content without the need to download and access all blocks.

Generally, the OMM does not provide a set of regulations for the block payload, so the users are free to store the information in the way they want or in the way they have defined with other partners. However, the model includes three pre-defined blocks useful in several scenarios. The *OMM-ID Block* carries a list of identification codes combined with the corresponding string type and a validity timestamp or time-span. The *OMM-Structure Block* indicates relations of the physical object to other objects, without the need of additional formats. The user can use one or several of the following fixed relations: isConnectedWith, isPartOf, hasPart and isStoredIn. Each relation can also be combined with a validity time span. The *OMM-Key-Value Template* provides a container for an arbitrary amount of key-value-pairs that can be used in many use cases.

In addition, we added a proposal for additional blocks (called OMM+) extending the OMM meta model. The *OMM+ Semantic Block* is an extension of the OMM-Structure Block and allows the definition of arbitrary relations similar to ontology relations (e.g., provided by RDF and OWL) relative to the physical object. Each relation consists of a triple (subject, predicate, and object) represented by uniform resource identifiers (URIs) combined with a validity statement. To indicate the physical object itself the URI `urn:omm:this` is used. The predicate can be use case specific or use common RDF/OWL object relations. It is also possible to include the OMM-Structure Block relations, e.g., the isConnectedWith relation by using the URI `urn:omm:structure:isConnectedWith`. This block allows the definition of simple semantic statements that can be processed semantically (e.g., with a graph reasoner) or without a reasoner just compare the strings of the relation triple.

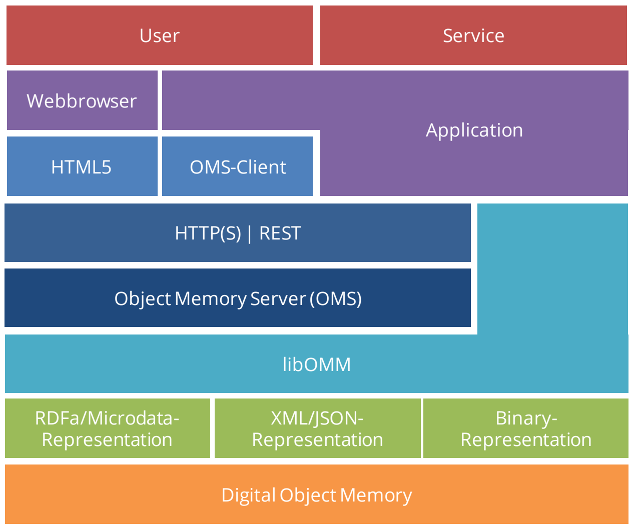

Figure 6. Hierarchical view of OMM-related components.

The *OMM+-Embedded Block* is meant to integrate an entire OMM-based memory into a specific block. In use cases where objects are physically combined to a compound object, or if access is physically hindered by arrangement of objects, it might be the case that the attached ID or the embedded system cannot be reached any longer. The problem can be solved by copying the object's memory, e.g., to the memory of the factory, so its memory can be accessed even if the object's label can no longer be reached. A specific subject meta data attribute `primaryID.<ID of integrated object>` is used to indicate the embedded memory's ID without the need of extracting the memory itself.

## B. Storage Infrastructure and Communication Interfaces

The storage infrastructure built on top of this data model is divided in two components. Firstly, a generic software library (libOMM) was developed to integrate the object memory model into Java- and C#-applications and can handle local XML-based OMM-representations. Secondly, we extended this library to a dedicated object server system called *Object Memory Server* (OMS) [26]. Figure 6 shows a hierarchical view of the OMM-related components. This server is divided into several modules that can be compiled depending on the intended application (see Figure 7).

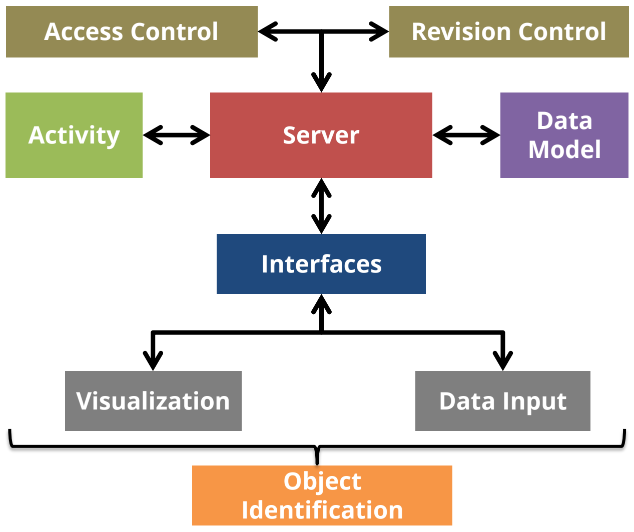

Figure 7. Server-based OMM Architecture.

To foster the usage of digital object memories, the access to memories is not restricted in general. However, the OMS is equipped with a role-based security module. The owner of a memory can restrict read and write operations for specific blocks or the entire memory. Three approaches are available to grant access to memories: passwords, certificates, and electronic ID cards. The simple approach uses a username and a password stored in a white-list containing all entities with access permissions. The more sophisticated approach utilizes digital certificates based on the ITU standard X.509 [27]. With certificates an additional way of restriction is possible: the certificate chain mode. This mode demands that an accessing entity uses a valid certificate and this certificate must provide a valid certificate chain with respect to the root certificate of the memory. The owner can select between two options. First the certificate must be directly signed by the owner or secondly a valid chain to the owner is sufficient. The latter option allows for certificate users to create their own child certificates and deploy them to other users of the memory. This approach lacks the risk that the user pool can be extended in an uncontrolled manner, but allows to distinguish between different ”subcontractors” (each using individual certificates inherited from a accepted authority) without any administrative costs by the memory owner. Finally, we created a prototype application based on the new German ID card (nPA). This card is issued to German citizens by the local registration offices and provides an electronic identification (eID) mechanism to create a unique but anonymous and application dependent ID for each card. This can be utilized to restrict memory access to such eIDs by using an DOMe infrastructure with access control [28], [29].

To increase the level of security the system also prevents the possibility of creating plagiarism by duplicating memories. If a server-based approach is used, the given API allows only block-based memory access, so each block has to be copied to another fake memory, but these ”new” blocks contain different creator information than the origin. If a solution based on RFID-tags is used, the framework can add an additional hash value incorporating the entire memory and the tag ID. A simple copy of the tag data breaks this hash value, due to a different tag ID. For more complex scenarios a more sophisticated solution providing a secure provenance change is described in the next section.

For memory interaction tasks two different interfaces are available (see Figure 6). Applications can use a RESTful HTTP-interface to access, to supplement and to modify object memories. End users can alter memories with the built-in web-based HTML5 user interface that can be utilized within a standard browser on multiple different devices. As mentioned before each memory can be accessed with a unique URL that serves simultaneously as access point and as the object’s primary ID. This URL begins with the DNS name or IP address of the OMS and ends with the name of the memory. In between the caller indicates which module of the OMS should be triggered. This module can be set to ’st’ for the RESTful storage interface or to ’web’ for the HTML5 user interface. Further actions and commands deployed to this module are added to this generic URL part. In the following, we use an exemplary memory that is stored on an OMS and accessible at the domain ’sampleoms.org’ and the sample memory named ’s_memory’:

```
http://sampleoms.org/st/s_mem
http://sampleoms.org/web/s_mem
```

The RESTful interface, represented throughout the ’st’ path, maps the functions and operations of the mentioned libOMM. The URL path `.../st/s_mem/block/_ids/` retrieves the list of all blocks with their IDs that allows applications to access the entire memory. For data retrieval located in a block with unknown ID the function `.../st/s_mem/toc/` can be utilized to get the table of contents of this memory that is a compressed set of meta data from all blocks (e.g., large meta data like clear text description and tags are excluded). Finally, a direct access to block meta data is possible, e.g., to access the creator of a block an application can use URL part `.../st/s_mem/<blockID>/meta/creator/` or `.../st/s_mem/<blockID>/payload/` to access or to change the block payload.

## C. *Smart Binary Encoding and Secure Provenance*

The proposed Object Memory Model and its XML representation can be easily used on a server-based infrastructure with virtually unlimited storage space. Introducing technologies with smaller storage spaces like RFID-tags [21], [30], [31] or cheap embedded controllers do not provide enough capacity to store all memory related metadata, not even by using compressed binary representations. To foster the usage of OMM-based memories even on such technologies, we introduce a context-aware dynamic schema-based mapping approach (see Figure 8) to reduce the overhead of OMM metadata [32]. The mapping schema is utilized to define the mapping process between OMS-interfaces and the corresponding byte stream. The advantage of this approach is to be format-independent by concentrating the mapping logic to the schema rather than using inflexible format-dependent code. The schema is created by an author, delivered to each stakeholder of this object memory and combined with the corresponding OMM access code. The storage space contains only the raw binary memory content.

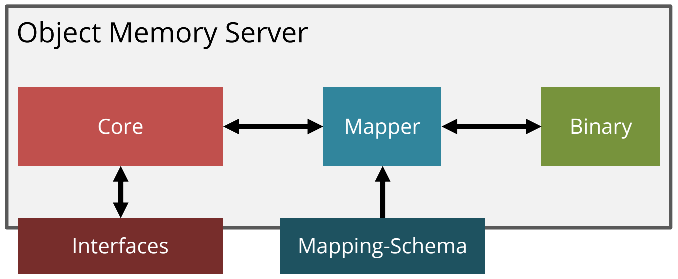

Figure 8. Binary mapping for Object Memory Models.

The schema itself contains rules to map the binary content to code interfaces. In this sample schema, the namespace metadata of a block is defined. The `<composition>`-section defines the structure of the corresponding binary part. Due to the variable length of a namespace, the binary representation consists of a length indicator (one byte) with a restriction to a maximum of 32 bytes and a value (the namespace string itself) as UTF-8 string. The `<interface>`-part defines a code interface including methods that enable the mapper to retrieve the data to be written and to transfer the loaded binary data to the application logic. The definition allows for specifying methods (e.g., for Java or C++ implementations) as well as properties (e.g., for C# or PHP implementations).

The mapping process supports different modes. First, it simply streams the given object memory data to a byte array and vice versa based on the schema definitions. The more advanced modes allow the schema author to limit specific information, e.g., the length of the *title* block metadata or the number of blocks of a specific type. In addition, priorities can be defined for each part of the metadata model to allow the mapper to dynamically drop metadata and blocks with low priority. This process can be applied in two different ways: only to newly written or changed blocks or retroactively to all existing blocks in the memory. The quality and quantity of this loss of information is transparently retrieved by the respective application.

The framework presented in the previous chapters is meant to leverage the process of free access to object memories, to encourage different stakeholders to contribute available data to such memories. However, other use-cases demand a more controlled and secured memory structure. E.g., in a pharma setting where drugs are filled in boxes, then are transported to a pharmacy and finally are sold to customers a consistently and verifiable documentation for these life-cycle steps is needed to ensure the necessary quality requested by the customer. For such purposes the object memory model can be extended, to guarantee the integrity and the authenticity of the memory and to prevent subsequent modifications and amendments of the stored data, as well as the illegal erasing of unwanted information similar to EPC Pedigree approach [33].

This goal is achieved with the help of an extended model by adding the following information to each block (see Figure 9). First, a hash-value of the block content (the metadata and the payload) is generated by concatenating the content and using a SHA-512 [34] hash function. Second, the ID of the predecessor block to connect each block with its predecessor is added. Third, a cryptographic signature is created by using standard

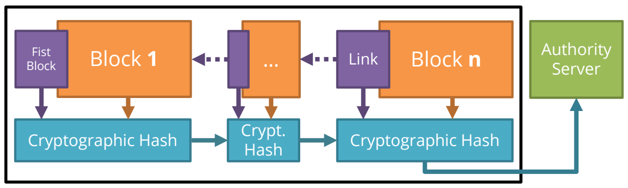

Figure 9. Secure Object Memory with linked Blocks.

X.509-certificates [27] and a private key. The aforementioned two values are integrated as certificate extensions. Finally, the signature of the last (meaning the most recent block) is stored on an additional *authority server*. This server contains the certificate of the most recent block for several memories.

These modifications can prevent an attacker to add additional blocks or to change existing blocks because he needs a private key to generate valid certificates. In addition, an attacker can no longer remove unwanted blocks from a memory because the integrity chain (linking each block with its predecessor) would be broken in this case. Finally, the authority server shows the certificate of the most recent block for each memory such that it is not possible to remove this block without leaving a broken integrity chain. However, since such techniques cannot be used directly by users, e.g., within a web browser, applications have to be extended (by using the mentioned libOMM or the RESTful interfaces) to benefit from this approach.

## V. Hardware Abstraction

The aforementioned scenario involves a heterogeneous set of goods equipped with different techniques, ranging from simple barcodes to embedded systems with storage and processing capabilities. In our sample scenario, a user does not have to care about processing power and storage capabilities of the hardware used to implement the object memory. In order to achieve this goal, the ADOMe framework includes a data access interface, which provides abstraction that allows accessing users and the objects themselves to complement their missing functionalities on their own, or at least to inform the outer environment about requested but not available functionalities.

This concept utilizes the aforementioned ADOMe framework that is built on modular software components. It allows the framework to run on a large variety of platforms raging from high-end servers to small embedded systems (see Figure 10). The framework's RESTful interface decouples communication with a memory from its concrete hardware and software implementation. In addition, it enables systems lacking embedded ADOMe functions (e.g., for logic processing) to outsource these to server-based ADOMe solutions by passing-through incoming requests.

In fact, we distinguish three kinds of instrumentation types and interaction approaches, respectively (see Figure 10). These solutions range from a packaging instrumented minimalistically with a barcode or an RFID-tag that can be read during the production and by any mobile compatible device, or instrumented with a fully integrated intelligent embedded system, incorporated into the packaging during the production process.

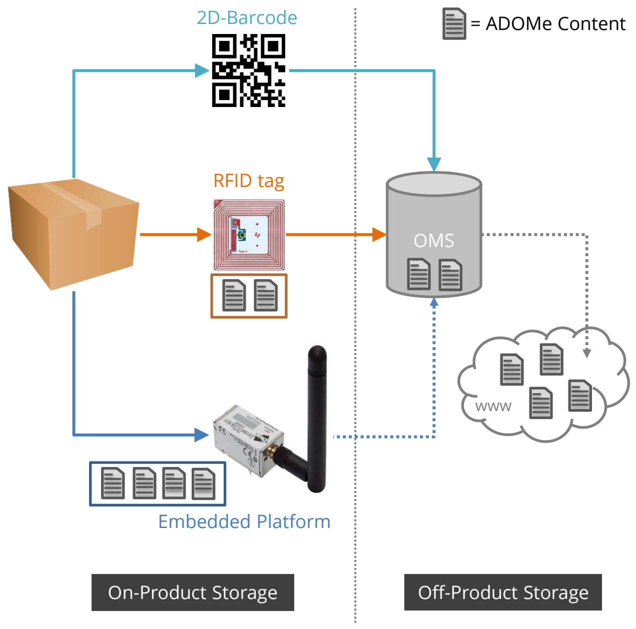

Figure 10. ADOMe hardware abstraction ranging from ”off-product” memories linked via barcodes to ”on-product” embedded systems.

The first approach, called *off-product*, is focused on a lightweight object instrumentation with a barcode or RFID-tag. An external device with internet connection is necessary to access the data stored on server-based solution (e.g., Object Memory Server). The reading and visualization of the data is done with the consumer's own internet compatible device. The drawback of this approach is that it heavily relies on the instrumented environments providing services like sensor measurements and data connections. An access to memory data is only possible with available internet connection. On the other hand, the centralized storage allows an access to memory data without access to the physical object.

Whereas the off-product scenario relies heavily on a dedicated infrastructure, the *on-product* scenario uses the full power of the inbuilt embedded system. Such systems (like Gadgeteer [35], Arduino [36] or other embedded controllers) are directly included within the packaging and are constantly tracking the physical properties of the product like temperature, humidity, air pressure, geo-localization over GPS position, or even shock detection, and thus enabling a precise autonomous tracking. Integrated visualization (ranging from simple multicolor LEDs to touch displays) allow users to get a full overview of the status of the intelligent product and to inspect the object's memory (see Figure 11) without the need of an external device (e.g., like a smartphone or tablet). The drawback of this solution is that the costs are much higher than simple barcodes or RFID stickers, due to the electronic components that need to be embedded into the packaging to realize the sensor value tracking. In addition, the memory data is only accessible as long as the physical device is in range if there is no distributed backup server kept in sync.

A hybrid solution set between the on-product and off-product approach is the so called *incycling* approach [37], [38] that might be a solution for the following cases: the cost-benefit ratio of the product concerning the price of an on-

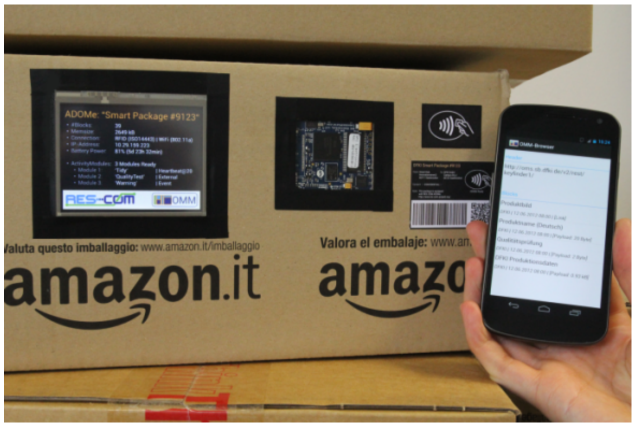

Figure 11. Performing built-in and on-product integrity checks on a smart package equipped with ADOMe and accessed by a mobile handheld device.

product instrumentation and the given surplus value of a local object memory is below 1, or the on-product instrumentation is only meaningful for a short term or a short life-cycle phase. Hence, incycling proposes the following scenario: a product is instrumented at a specific point of time (e.g., a new lify-cycle phase). As soon as the next phase is applied (e.g., the product is consumed or local sensors are not needed anymore) the intelligent board is removed from the package and is attached to a new object that is just entering this phase. Due to the fact that such an on-product memory can be reused in a closed-loop and the costs are distributed over several carriers of the memory, the value of benefit is increased significantly. Sample incycling applications range from attaching an embedded controller to a solid product during transport to workpiece carriers that provide memory functionalities for each workpiece that is carried through the production line.

The next evolution is the integration and processing of logic code directly inside such active object memories. This approach has the advantage that there is no need for an external device to execute the logic code, no need to download the entire memory content to this device and no need to keep code ready for each platform and for every object type. The code fragments (called *snippets*) are stored in blocks inside the memory. This allows manufacturers to deliver their products with build-in quality check or monitoring scripts. The snippets can be addressed with a unique ID or a semantic concept (e.g., ”#QualityCheck”) stored in the block meta data. The latter one can be deemed as definition of semantic questions and accessed from external sources that trigger such concepts as asking a question (trigger of ”#QualityCheck” equals asking the question ”Are you ok?”). To complement the activity module, snippets can also be triggered event-based or by a so called heartbeat timer. The event-based mechanism allows users to define a set of monitored blocks for each snippet and the activity module triggers the snippet each time these blocks are changed. The heartbeat represents a timer, that allows for snippet execution in a defined period (e.g., every 30 minutes).

The processing and execution location (either in embedded systems ”on-product” or on server-based solutions ”off-product”) is transparent for the access application. In case of no fitting software module available in memory, an access to a so called app store is possible based on a semantically

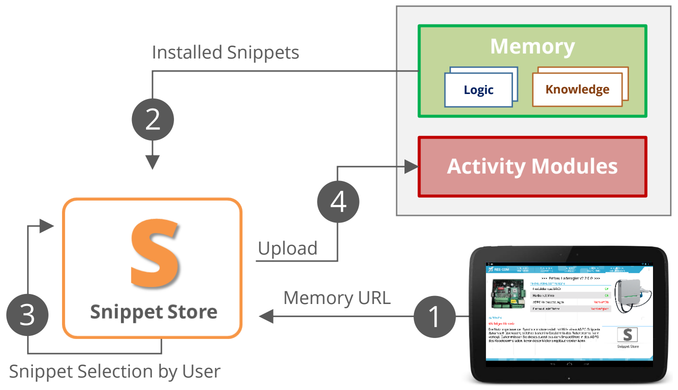

Figure 12. Dataflow of Activity Update with external Snippetstore.

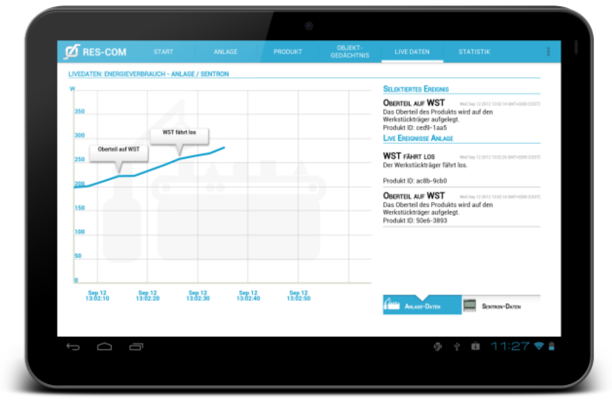

Figure 13. Client application displaying ADOMe-based measurements created by a software module downloaded from an external appstore.

defined logic description. This store contains a large set of logic modules for different applications and platforms, ranging from fixed domain-specific code to generic modules capable of parametrization. If available the framework downloads, installs and executes the downloaded module from the store (see Figure 12). Client applications performing operations based on memory data can be extended with in-memory or app store modules the same way.

## VI. User Interaction with Object Memories

In the following section, we describe the interaction forms with smart objects within the already mentioned logistic process starting at the warehouse and ending at the customer's house. The kind of interaction used within these scenarios is closely linked to the hardware type and chosen packaging type.

During the entire logistics chain, an intelligent product will need to go through several checks and validation processes until reaching the end consumer. In the first stage, the product leaves the production phase, in which an object memory was created and filled with static product and manufacturer data. This is done by machine-to-machine (M2M) communication without any user involvement. The special monitoring or processing capabilities of some products can be achieved with the already mentioned snippets that can perform such tasks. The manufacturer adds tailored snippets to the object memory and starts execution. If the product is equipped with an on-board memory the snippets are inside the embedded controller otherwise (in the off-product case) the execution is done within the object memory server.

During transport sensor values are generated, e.g., by monitoring temperature, humidity, acceleration, and position either by the object itself (on-product) or by the environment (off-product). Since memories may contain hierarchical information about their environment, sensor data measured at higher levels (e.g., on container or palette level) can be passed to and stored in lower levels (e.g., a transport box or the object itself). The continuously updated memory can be accessed (pulled) with a simple web-browser or control station directly (off-product, via OMS) or by relaying information to higher levels (on-product), e.g., palettes relay data to containers and these containers sync their data to web-based OMS. Additional important data can also be pushed by active memories with the help of output channels triggered by snippets (e.g., a product can send an email as soon as a temperature threshold is exceeded).

Some products may require special treatment or have to comply with special customer demands. In our scenario, compliance is verified by a worker by means of a mobile device (smartphone or tablet). Once the worker has scanned the barcode or RFID-tag attached to the object, an app running on the mobile device connects to the on-product memory or the off-product object memory server. In case of an on-product memory the app can directly access the memory, e.g., with Wi-Fi Direct [39] or Bluetooth [23]. The app retrieves memories nearby with the UPnP [40] standard (see also Figure 11). The snippet store is available as smartphone or tablet app and is structured and handled the same way as known app stores on such devices. A check for available snippets regarding the specific object is performed and displayed to the user. He or she can select the favored one and automatic upload and activation process is initiated (see Figure 12). Such snippets ease the process of adding and adjusting additional monitoring and processing logic.

The identical concepts and hardware extended with a display can be integrated in products of everyday life. An example is the already mentioned milk carton ”Milky” that supports direct product-to-consumer interaction within three life-cycle steps [41]. Due to the highly instrumented product no further infrastructure or tools are necessary to interact with the carton. In the advertising phase (intended to raise the customer's attention) product-related data is presented via the built-in display. A generic framework gets all related data from the object memory. Information is provided by the manufacturer and can be extended by the retailer. Once bought, milky continues monitoring conditions like temperature and humidity without any user interaction if the customer allows such tasks in his or her preferences. Otherwise, milky asks the customer. Any condition violations are displayed. At home milky informs about best-before dates coming closer based on a memory data set by the manufacturer and possibly adjusted by sensor readings (e.g., higher temperatures are measured and the best-before date is set to an earlier date) Once the content is consumed recycling information are displayed regarding the current position. In addition to these presentations based on raw data, the object can also display the 'face' of milky (see Figure 14). This anthropomorphic style presents the 'mood' of the milk carton. The system provides four different moods: happiness, sadness, amorousness and annoyance reflecting the

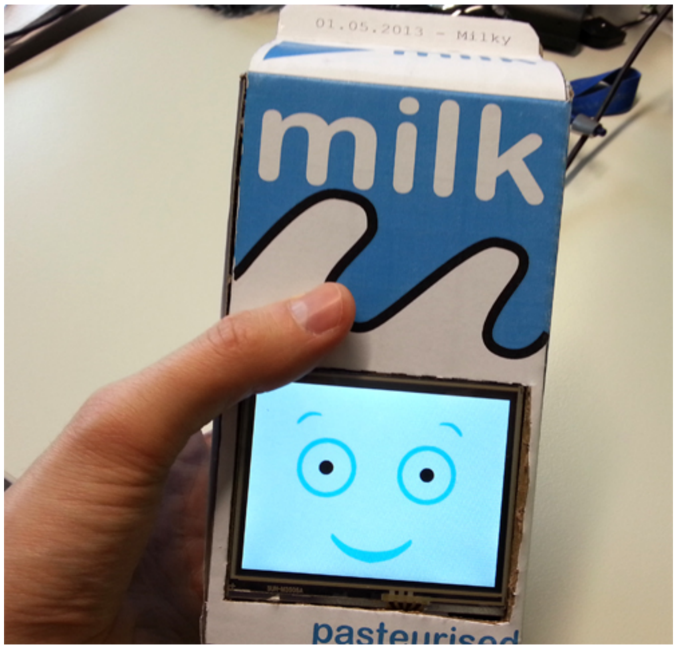

Figure 14. Smart milk carton 'Milky' in anthropomorphic mode 'happy'.

current state of the milk inside the carton by addressing the customer's feelings. Happiness is the default mood representing the milk to be in a good condition. Sadness occurs shortly before the carton is empty. Amorousness is used to display that other products or customers are detected in range. And finally annoyance (addressing the fact that in German language the states annoyance and sour are expressed by the same word 'sauer') is displayed once the milk has turned sour.

## VII. Conclusion and Outlook

This article summarized work in progress concerning a framework for setting up so-called Active Digital Object Memories. Its contribution is twofold: In order to leverage the application of such data collections in open scenarios, this framework seeks to 1) embrace different ways of deploying answering logic in a collection, and 2) provide abstraction from the technical diversity of existing infrastructures for collecting object-related data, which employ technology embedded into physical objects, virtual data stores located in the Web, and combinations of both approaches. Future work will address in the very first place the proposed method of distributing processing logic within this framework: the app store implementation. A first prototype illustrates the feasibility of this approach in a manufacturing scenario involving passive RFID, Android tablets, and embedded devices; however, more efforts are needed to verify that concept for broader range of embedded system platforms as well as logic hosted for deployment.

## Acknowledgment

This research was funded in part by the German Federal Ministry of Education and Research under grant number 01IA11001 (project RES-COM) and 01IS12050 (project OMM++). The responsibility for this publication lies with the authors.

## References

[1] A. Kröner, J. Haupert, C. Hauck, M. Deru, and S. Bergweiler, "Fostering access to data collections in the internet of things," in UBICOMM 2013, The Seventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies located at NexTech 2013, September 29 - October 3, Porto, Portugal, W. Narzt and A. Gordon-Ross, Eds., IARIA. IARIA, 9 2013, pp. 65–68.

[2] R. Barthel, K. Leder Mackley, A. Hudson-Smith, A. Karpovich, M. de Jode, and C. Speed, "An internet of old things as an augmented memory system," in Personal and Ubiquitous Computing, vol. 17. Springer London, 2011, pp. 321–333.

[3] D. Fitton, F. Kawsar, and G. Kortuem, "Exploring the design of a memory model for smart objects," in Ambient Intelligence and Smart Environments, vol. 4: Workshops Proceedings of the 5th International Conference on Intelligent Environments. IOS Press, 2009, pp. 33–38.

[4] P. Stephan, G. Meixner, H. Kling, F. Flörchinger, and L. Ollinger, "Product-mediated communication through digital object memories in heterogeneous value chains," in Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications. IEEE, 2010, pp. 199–207.

[5] M. Schneider, M. Velten, and J. Haupert, "The objectrules framework - providing ad hoc context-dependent assistance in dynamic environments," in Proceedings of the Sixth International Conference on Intelligent Environments. July 19-21, Kuala Lumpur, Malaysia. IEEE Computer Society CPS, 2010, pp. 122–127.

[6] C. Decker, T. Riedel, M. Beigl, L. Moreira, S. Souza, P. Spiess, and S. Haller, "Collaborative business items," in Proceedings of IE 07: 3rd International Conference on Intelligent Environments, Ulm, Germany, 2007, pp. 40–47.

[7] M. Mühlhäuser, "Smart Products: An introduction," in Constructing Ambient Intelligence: AmI 2007 Workshops. Springer Berlin / Heidelberg, 2007, pp. 158–164.

[8] EURIDICE, "EURopean Inter-Disciplinary research on Intelligent Cargo for Efficient, safe and environment-friendly logistics," 2011, [last accessed: 05-16-14]. [Online]. Available: http://www.euridice-project.eu/index.php/web/pubdocs/58

[9] iCargo, "Intelligent Cargo in Efficient and Suitable Global Logistics Operations," 2014, [last accessed: 05-16-14]. [Online]. Available: http://i-cargo.eu/type/publications

[10] stuffl UG, "Anythinx," 2014, [last accessed: 05-16-14]. [Online]. Available: http://www.anythinx.de/

[11] qipp AG, "qipp," 2014, [last accessed: 05-16-14]. [Online]. Available: https://www.qipp.com/en

[12] EVRYTHNG Ltd., "EVRYTHNG Every Thing Connected," http://evrythng.com/, [last accessed: 05-16-2014].

[13] Xively (by LogMein Inc.), "Xively - Internet of Things Platform Connecting Devices and Apps for Real-Time Control and Data Storage," http://xively.com/, [last accessed: 05-16-2014].

[14] A. Kröner, J. Haupert, M. Seißler, B. Kiesel, B. Schennerlein, S. Horn, D. Schreiber, and R. Barthel, "Object memory modeling - W3C incubator group report," 2011, [last accessed: 05-16-2014]. [Online]. Available: http://www.w3.org/2005/Incubator/omm/XGR-omm-20111026/

[15] R. Barthel, A. Kröner, and J. Haupert, "Mobile interactions with digital object memories," Pervasive and Mobile Computing, vol. 9, Issue 2, 2013, pp. 281–294.

[16] M. Mealling and R. Denenberg, "RFC 3305: Uniform resource identifiers (uris), urls, and uniform resource names (urns): Clarifications and recommendations," 2005.

[17] T. Berners-Lee, R. Fielding, and L. Masinter, "RFC 3986: Uniform resource identifier (uri): Generic syntax," 2005.

[18] O. Rosenbaum, Das Barcode-Lexikon, ser. Edition advanced. BHV-Verlag, 1997.

[19] International Organization for Standardization, "Information technology automatic identification and data capture techniques data matrix bar code symbology specification," Geneva, Switzerland, 2006.

[20] ——, "Information technology — automatic identification and data capture techniques — qr code 2005 bar code symbology specification," ISO/IEC 18004:2006, 2006.

[21] ——, “Identification cards - contactless integrated circuit(s) cards - proximity cards,” ISO/IEC 14443:2000, 2000.

[22] ——, “Near field communication interface and protocol-2,” ISO/IEC 21481 / ECMA-352, 2010.

[23] Bluetooth Special Interest Group, “Bluetooth Specification ,” 2013, [last accessed: 05-16-14]. [Online]. Available: https://www.bluetooth.org/en-us/specification/adopted-specifications

[24] IEEE, “802.11: Wireless LANs,” 2014, [last accessed: 05-16-14]. [Online]. Available: http://standards.ieee.org/about/get/802/802.11.html

[25] DublinCore, “DCMI Type Vocabulary,” 2014, [last accessed: 05-16-14]. [Online]. Available: http://dublincore.org/documents/dcmi-type-vocabulary/#H7

[26] J. Haupert, “DOMeMan: A framework for representation, management, and utilization of digital object memories,” in 9th International Conference on Intelligent Environments (IE) 2013, July 18-19, Athens, Greece. IEEE, 7 2013, pp. 84–91.

[27] ITU-T Recommendation X.509 Version 3, “Information technology - open systems interconnection - the directory: Authentication framework,” 1997.

[28] B. Brandherm, J. Haupert, A. Kröner, M. Schmitz, and F. Lehmann, “Demo: Authorized access on and interaction with digital product memories,” in 8th Annual IEEE International Conference on Pervasive Computing and Communications, March 29 - April 2, Mannheim, Germany. IEEE Computer Society, 2010, pp. 838–840.

[29] ——, “Roles and rights management concept with identification by electronic identity card,” in 8th Annual IEEE International Conference on Pervasive Computing and Communications, March 29 - April 2, Mannheim, Germany. IEEE Computer Society, 2010, pp. 768–771.

[30] G. Reinhard, P. Engelhardt, E. Genc, T. Irrenhauser, M. Niehues, M. Ostgathe, and K. Reisen, “Einsatz von RFID in der Wertschöpfungskette - Konsortium entwickelt RFID-basierte hybride Steuerungsarchitektur und Bewertungsmethode für Wertschöpfungsketten,” RFID im Blick - Sonderausgabe RFID in der Region München, 3 2011.

[31] N. Schlitter, F. Kähne, S. T. Schilz, and H. Mattke, Operations and Technology Management, ser. Innovative Logistics Management. Erich Schmidt Verlag, 2007, vol. 4, ch. Potential and Problems of RFID-Based Cooperation in a Supply Chain, pp. 147–164.

[32] A. Höh, “Smart Binary Representation For Digital Object Memories,” Bachelorthesis, Saarland University, 2014.

[33] EPCglobal, “The Pedigree Ratified Standard Version 1.0,” 2007, [last accessed: 05-16-14]. [Online]. Available: http://www.epcglobalinc.org

[34] D. Eastlake 3rd and T. Hansen, “RFC 6234: US secure hash algorithms (SHA and SHA-based HMAC and HKDF),” 2011.

[35] Microsoft Corporation, “.NET Gadgeteer,” 2014, [last accessed: 05-16-14]. [Online]. Available: http://www.netmf.com/gadgeteer/

[36] Arduino, “Arduino Project,” 2014, [last accessed: 05-16-14]. [Online]. Available: http://arduino.cc/

[37] B. Brandherm, A. Kröner, and J. Haupert, “Incycling - sustainable concept for instrumenting everyday commodities,” in Proceedings of the International Workshop on Networking and Object Memories for the Internet of Things. Workshop on Digital Object Memories (DOME-11), located at UbiComp 2011, September 17-21, Peking, China. ACM, 9 2011, pp. 27–28.

[38] B. Brandherm, A. Kröner, J. Haupert, M. Schmitz, F. Lehmann, and R. Gampfer, “Sustainable instrumentation of everyday commodities - concepts and tools,” in 10th Annual IEEE International Conference on Pervasive Computing and Communications, 10th, March 19-23, Lugano, Switzerland. IEEE Computer Society, 2012, pp. 467 – 470.

[39] WiFi Alliance, “WiFi Direct,” 2010, [last accessed: 05-16-14]. [Online]. Available: http://www.wi-fi.org/discover-wi-fi/wi-fi-direct

[40] International Organization for Standardization, “Information technology - UPnP device architecture - part 1: UPnP device architecture version 1.0,” ISO/IEC 29341-1:2011, 2011.

[41] M. Deru and S. Bergweiler, “Milky: On-product app for emotional product to human interactions,” in Proceedings of the 15th International Conference on Human Computer Interaction with Mobile Devices and Services, 15th, August 27-30, Munich, Germany. ACM, 2013.