From Smart Objects to Smarter Workflows – An Architectural Approach

Steffen Kunz, Benjamin Fabian, Holger Ziekow
Institute of Information Systems
Humboldt-Universität zu Berlin
10178 Berlin, Germany
steffen.kunz, bfabian, ziekow@wiwi.hu-berlin.de

Dirk Bade
Distributed Systems and Information Systems Group
University of Hamburg
22527 Hamburg, Germany
dirk.bade@informatik.uni-hamburg.de

***Abstract*—To live up to its promised business impact, the emerging Internet of Things must be accompanied by new ways to decentralize and adapt business logic for smart objects. Smart object lifecycle management refers to the decentralized, event-based, context-sensitive execution and adaptation of processes by smart physical objects during their path through different domains and their interaction with other entities. These entities include the end-user or consumer, other objects, or web-based information services. In this paper, we propose an approach how events and reaction to those events can be virtually attached to certain situations in the lifecycle of physical smart objects. In addition, we present an smart object lifecycle architecture for enabling multiple stakeholders to provide the necessary event and processing information for specific domains.**

***Keywords*-Internet of Things, Smart Objects, Event-Driven Architecture**

I. Introduction

Research and practical approaches to establish an Internet of Things are currently mostly data-focused, not process-focused. An example is the proposed EPCglobal architecture [1] where current standards exclusively emphasize data flow between smart objects like RFID-tagged items and background infrastructures for data discovery and storage. To increase its business impact, the Internet of Things needs to be extended by process-oriented business logic. Especially, it must be accompanied by new ways to decentralize and adapt business processes and workflows for smart objects. The management and control of distributed dynamic processes for individual objects poses several challenges to supporting information systems [2], [3]. This applies to an even higher degree to processes spanning across several organizations and different physical locations. The challenges include flexibility in sharing object data and business logic for supporting dynamic value chain processes [4]. Other challenges concern achieving system scalability and reactivity in order to timely handle events in the processes [3]. In this paper, we present an infrastructure that facilitates event processing for smart objects and aids developers in coping with these emerging challenges.

The main contributions of this paper are: (i) to provide an approach that allows objects to influence their processes across different domains, (ii) the development of the Smart Objects Lifecycle Architecture (SOLA), including mechanisms to decentralize data control as well as to accelerate data access and (iii) testing the architecture in a global setting.

The rest of the paper is structured as follows: First, we present related work in the following Section II, which also serves as an introduction to the multiple and heterogeneous research areas important for our approach. In Section III, we give a more detailed problem definition based on an example, followed by an initial solution approach in Section IV. Section V presents SOLA. In particular, we show how to conduct smart object lifecycle management, including the necessary global discovery, retrieval, and processing of the corresponding data. Then, in Section VI, we present experiment results of SOLA. In Section VII we discuss our research and outline areas for future work. Section VIII reviews the impact of SOLA on interorganizational IT infrastructures and Section IX concludes the paper.

II. Background and Related Work

Related work in the research area of this paper can be found in four major disciplines: Business Process Management (BPM), Complex Event Processing (CEP), IOT or EPCglobal discovery service design, and content delivery systems.

The discipline of BPM [5] focuses on deadline-based escalation in process-aware information systems. In order to handle escalations, the authors propose the so-called 3D approach, which means *Detect* late processes, *Decide* what to do, and *Do* the escalation which was decided on. In [6], the authors apply *worklets*, an extensible repertoire of self-contained sub-processes and associated selection rules. Their work is similar to the escalation approach presented by [5], since each task of a process instance may be linked to an extensible repertoire of actions that can continuously be developed, especially through experiences gained during run-time. In [7] the authors developed the ADEPT flex system for supporting dynamic changes of running workflow instances while preserving their correctness and consistency. They show that complex structural changes can be applied to a workflow instance during its execution. So far, little attention has been devoted to research concerning usability

aspects of managing processes involving complex events. [8] provides a mapping from Web Service Business Process Execution Language (WS-BPEL) statements to Business Process Modeling Notation (BPMN) artifacts. They integrate their mapping into the open-source modeling framework Oryx.[1]. In contrast, our approach focuses on events, which makes it harder to provide a complete integration to a process modeling language such as BPMN.

The second discipline where related work can be found is CEP. The usage of CEP techniques in the context of RFID and sensor applications has been proposed by [9], [10], [11], among others, but existing approaches incorporating CEP are primarily designed for a specific purpose and do not consider the integration of superordinated business processes with event data. Especially the latter aspect also holds for more general data stream processing architectures, e.g., GSN [12]. In [13], a commercial patient monitoring system that applies above mentioned technologies is presented. The respective company states that their product allows for workflow automation in the healthcare domain, but only few details are given. [14] recently proposed to extend the Sensor Alert Service of the Sensor Web Enablement [15] with CEP capabilities and described how this is achieved in their SAPHE project. All projects make use of CEP techniques in order to detect complex patterns in streams of primitive RFID or sensor events to derive higher-level information about perceived conditions or to trigger certain actions. The overall lifecycle of the related objects and their respective processes are not considered.

[16] motivated the use of software agents as building blocks for CEP middlewares and applications while [17] is (among others) using an agent-based middleware incorporating CEP techniques for asset management.

The recently coined term Event-Driven BPM (ED-BPM) is the combination of the two research disciplines mentioned before. In [18] and [19], the authors show the potential of the combined usage of CEP and BPM in the logistics and finance sector. [20] proposes an architecture for event processing of RFID data in enterprise information systems applying workflow models to extract complex event patterns.

The third research area directly relevant to our work is the design of IOT (or EPCglobal Network [1]) discovery services to globally locate relevant information sources for objects, where, contrary to the EPC Information Services (EPCIS) repositories for actual object information itself [21], no common standards are finalized yet. Notable discovery service designs have been created by the BRIDGE project [22], [23], the company Afilias, and an IETF working group on Extensible Supply chain Discovery Services (ESDS) [24], [25]. P2P-based approaches for ONS and discovery services have also been proposed by [26] and [27], among others. To our knowledge, none of those proposals for discovery service design specifically considers cross-domain data of smart objects, where not only historic general object information is discovered and exchanged, but also rather future-oriented complex event patterns and event-processing workflows can be discovered to influence processes directly.

The fourth technological area related to our work are content distribution networks (CDN). The solution of *Akamai* [28] supports scalability for data retrieval as well as reduction of network latency. Scalability is achieved trough hosting multiple copies of the data. Reduction of network latency is addressed by distributing the data to edge networks, i.e., networks of local Internet providers. Users are directed to the closest copy of the requested data through specialized name servers in the DNS system.

While the design of Akamai is suitable for Web data, it does not fit well for the IoT application domain because with Web data, few pieces of data are relevant for a potentially very large number of users. In contrast, RFID events result in many pieces of data that are only relevant to a very small group of users. In addition, Web data is relatively static. This is fundamentally different to RFID events, which are continuously captured throughout the supply chain. Such dynamics also impact the required caching and redirection mechanisms.

III. Application Scenario

For illustration of the real world challenges, we especially focus on a reference example, the life cycle of fish fillet. Figure 1 depicts the overall lifecycle using BPMN [29]. Notation artifacts colored in white represent the different stages in the lifecycle of an fish fillet, while the artifacts and the annotations colored in grey are part of the solution approach presented in Section IV.

The different domains in the lifecycle of an fish fillet are: manufacturer, logistics service provider, wholesaler distribution hub, wholesaler store, and the customer. The normal lifecycle of the fish fillet would be: load the fish fillet on a truck and ship it to the wholesaler distribution hub. Next, the fish fillet is unloaded and stored in a cold store. The fish fillet is then again loaded on a truck and shipped to the wholesaler store, where it is offered in in a cold shelf on the shop floor. Finally, the customer buys and eats the fish fillet.

During the lifecycle of an fish fillet, there are many situations and events that could influence the planned workflow, e.g., a cold store breakdown or a late or missing shipment. Ideally, a reaction to such events is immediately initiated right on spot, by the object itself. Our solution approach to this issue is presented in the next section.

IV. Solution Outline

We distinguish between escalation and exception events, adopting the terminology from [5] who focus on the problem of escalation handling in insurance processes. While

[1] http://www.oryx-editor.org/

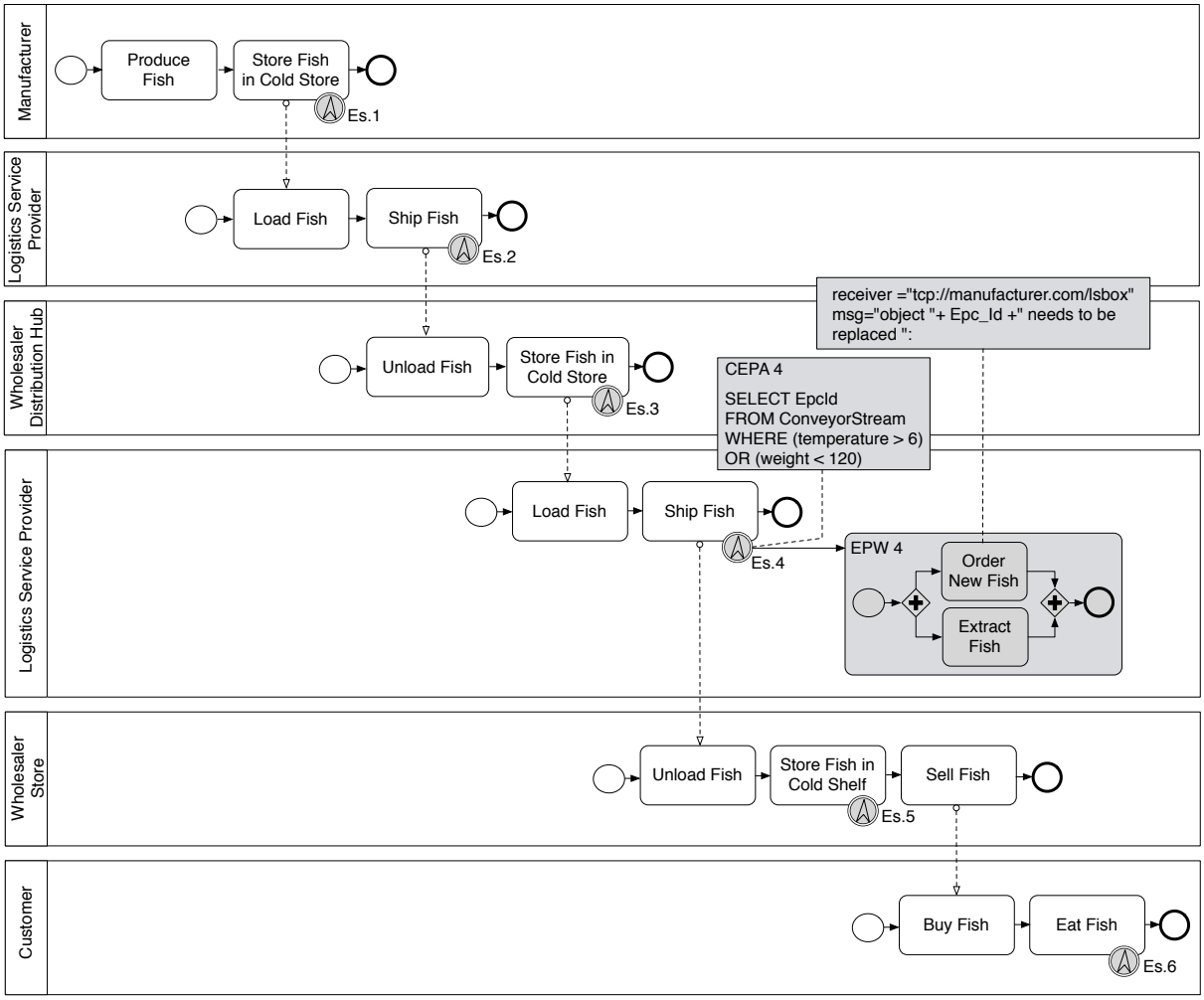

Figure 1. Process Diagram of Fish Fillet Lifecycle

escalation corresponds to events that are expected to happen regularly and can be anticipated in advance, an *exception* can be caused by any change of the environment, is difficult to identify, and even more difficult to handle. In our current research and in this paper we focus on escalation.

During the object lifecycle of a fish fillet displayed in Figure 1, there are many escalation events, presented as BPMN escalation event artifacts, which have to be taken care of. These escalation events need to be identified and reacted on, as the fish fillet needs to be treated very carefully in order not to spoil it. We assume that the required context information is provided through RFID readers or sensor networks located along the way of the object. Due to space limitations, we only focus in detail on one escalation event within the overall object lifecycle. That is, we consider the case where the fish fillet is shipped to the wholesaler store (see Figure 1 ES.4).

In our approach, we identify certain escalation events in the lifecycle of an object, e.g., if the temperature of an object exceeds a specified threshold, then a certain *Event-Processing Workflow* (EPW) needs to be executed. The escalation events are defined as *Complex Event Patterns* (CEPA) in an Event Pattern Language (EPL) [30, pp.33]. An example of a complex event pattern – attached as annotation to the BPMN escalation event artifact – is depicted in Figure 1, Escalation 4 (*Es. 4*). Similar to a database, this statement is continuously evaluated upon arrival of new sensor events, but only generates a result in case its condition – temperature above 6°C or weight below 120 g – is fulfilled. Once the system detects sensor events coming from the sensor network, which correspond to (or match) the complex event pattern statement (here *ES. 4*), it shall execute the corresponding event-processing workflow (here *EPW 4*) – that is to say, alert the manufacturer that a new fish fillet has to be shipped and extract the object. The event-processing workflows, are modeled as BPMN workflows, which are directly executed by the middleware (processing service) of our system; they are used to orchestrate service agents

handling the pre- and post-processing of (complex) events (see next Section). For this purpose, the event-processing workflows are annotated with additional information used to specify the sequence of task execution as well as to parameterize the tasks (see Figure 1, *EPW 4*).

In order to realize such kind of scenario, we developed SOLA which is detailed in the following section.

V. Smart Object Lifecycle Architecture

Now, we turn the discussion to the details of SOLA. The pyramid in Figure 2 gives an overview about all building blocks of SOLA, i.e., software components, languages, etc. The building blocks in italic bold letters have been developed and implemented by ourselves, whereas those in normal letters have been specified by standardization organizations, such as EPCglobal. Based on the event producers in the sensor layer, the data flow continues with the data discovery and retrieval layer and the processing layer, finally reaching the top in the form of an event consumer, i.e., the application layer. Some of the building blocks have already been introduced; the others are introduced in the next paragraphs where an overview of the core system elements, especially the event-processing middleware, the P2P discovery service, and the cloud EPCIS is given.

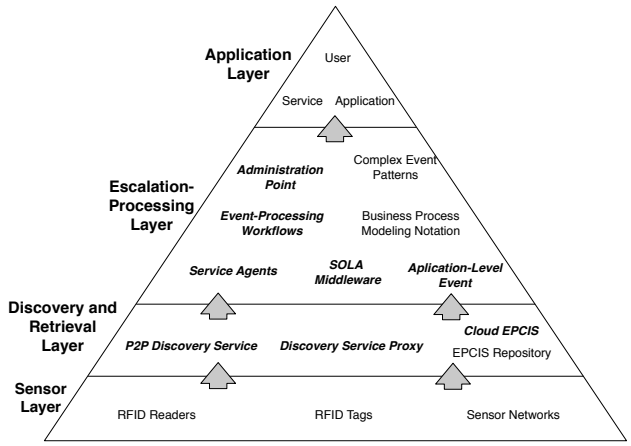

Figure 2. SOLA Pyramid

For the near-future IOT, we anticipate that most of the data will be stored in distributed repositories rather than on the object itself or on attached tags because on-tag memory is still expensive and therefore limited. In addition, information about objects must also be available for retrieval or update, even if an object is temporarily not connected to the IOT. This assumption is shared by the large industry consortium EPCglobal that is responsible for standardizing a global networked information architecture for smart objects. This EPCglobal Network [1] promises to enhance global information flow for simple as well as smart objects [31]. Parallel to general object information, we expect a high prevalence of storing data *on-network* (vs. *on-tag*) also for complex event patterns and event-processing workflows.

The backbone of our system architecture (cf. Figure 3) is compliant with the EPCglobal standards [1] where RFID tags are associated with unique *Electronic Product Codes* (EPC). Information about the EPC and the corresponding object is stored in distributed information sources on the Internet, such as the *EPC Information Service* (EPCIS) repositories, which are located by looking up their addresses – based on the EPC as a search key – from a discovery service. As an extension to this basic architecture, we introduce *cloud EPCIS* that mirror EPCIS repositories and reduce network delay in data access.

Similar to the EPCglobal proposal, we use the EPCIS repositories of different domains to store the complex event patterns and event-processing workflows for individual objects (depicted in Figure 3 step (1)). Those are defined at an *administration point* (step (0)), according to the method described in Section IV and then stored at the corresponding EPCIS repository.

Before the middleware is able to interact with our discovery service for EPCIS repositories, it initially has to be bootstrapped and configured by registering a special event-type filter and an initialization workflow for orchestrating the *service agents* of the *middleware*, each of them is responsible for processing exactly one workflow activity. The filter is used to trigger the execution of the initialization workflow upon every RFID-related event. The initialization workflow itself specifies the steps for looking up the addresses of the corresponding EPCIS repositories, and prepares the subsequent retrieval and registration of the complex event pattern and event-processing workflow for the individual EPCs.

Once an RFID tag is read, the RFID reader dispatches the EPC to the middleware. The event passes the registered filter and thereby triggers the orchestration of the initialization workflow of the middleware (cf. Figure 3, step 2). The service agent is responsible for executing the first step within this orchestration and establishes a connection to the *proxy* (3), which in turn connects to the *discovery service gateway* (4). The *discovery service* takes arbitrary object identifiers, in our case the EPC, as well as a domain identifier as input (see Section V-B for details) and resolves them to a list of corresponding information sources, e.g., EPCIS repositories and associated cloud EPCIS, where public or domain-specific information about the object is stored. The list is then filtered for the most suitable information sources and returned to the middleware (see Section V-C for details). With the help of the filtered list, the middleware connects to the corresponding EPCIS repositories and requests the complex event patterns and event-processing workflows for the domain in which the object is currently located (5).

The complex event patterns and event-processing workflows are then registered at the complex event processing

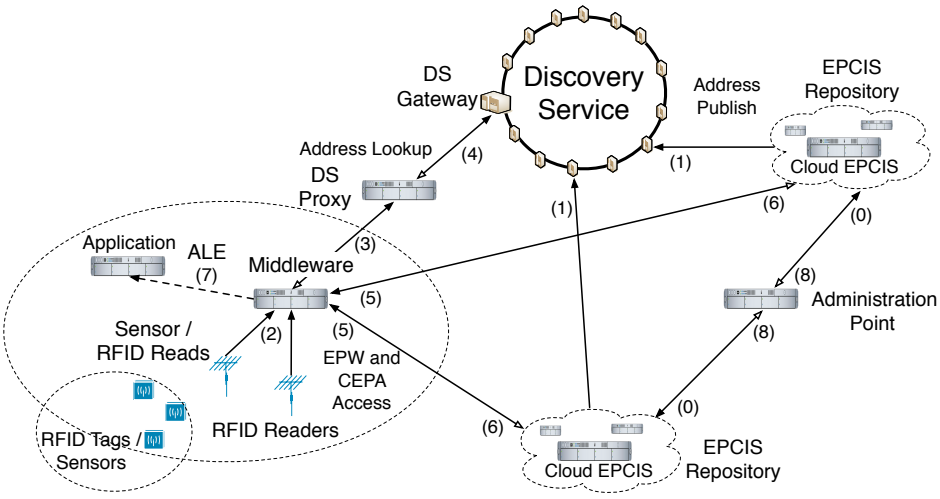

Figure 3. Smart Object Lifecycle Architecture

engine of the middleware. From that point onwards, the sensor data and event streams are analyzed and compared with the registered complex event patterns designed for the current domain of an object. Once a complex event pattern is matched, a complex event is generated and the corresponding event-processing workflow, handling the escalation, is executed. Finally, in step (6), the escalation event that occurred and the event-processing workflow that was executed, are updated in the object history at the EPCIS repository, enabling an information exchange among all stakeholders, which have legitimate access to the object's lifecycle. In some cases, the event-processing workflow may additionally include the creation of an *Application-Level Event* (ALE), transmitted to and interpreted by a service or an application (7). In addition, an user is also able to view the event and process history for individual objects at the administration point (8).

A. SOLA Agent-Based Middleware

As part of SOLA, we created an agent-based middleware architecture for the detection and processing of complex event patterns within streams of primitive events [32]. Such primitive events may originate from RFID systems or wireless sensor networks, and usually contain raw sensor measurements, such as an EPC read by an RFID reader. In order to detect complex relationships between multiple primitive events, possibly spanning different event streams, complex event-processing techniques (cf. [33]) are applied. For this purpose, our middleware incorporates a high-performance, open source event-processing engine (*Esper* [30]), which allows to define complex event patterns using an SQL-like event-processing language (see Figure 1, *Es. 4*).

Moreover, our middleware allows to individually preprocess the incoming events before they are forwarded to the complex event-processing engine, and to post-process the resulting generated complex events. During pre- and post-processing, arbitrary, user-defined actions may be carried out. These actions are defined using annotated BPMN-based processing workflows, which represent the orchestration of service agents and are used for escalation handling. An example of such post-processing is the initialization of the middleware as described previously. A simplified event-processing workflow, which is enacted once the complex event pattern is fulfilled, is also depicted in Figure 1 (*EPW 4*) and was further discussed in Section IV.

In addition to executing arbitrary actions (e.g., informing a third party about a specific condition), the post-processing stage may also be used to create individually tailored application-level events, which only contain relevant information in a desired format, by invoking conversion, translation, and encoding services. The results can then directly be further processed by enterprise systems, such as supply chain management and workflow-management systems, other smart objects, or web-based services. This way, event producers (RFID/sensor readers) and event consumers (like enterprise systems) do not need to adapt their business logic to the middleware's functionality and data formats, but may abstract from how events are generated and how actions are carried out, benefiting from the middleware's ability to conduct the overall processing.

Figure 4 depicts the subscription process, the post-processing of sensor data once a complex pattern has been detected (for brevity, the pre-processing stage as well as components related to security, event-logging, and the tasking of sensors are omitted), and the generation of an application-level event. In step (1a-b) the complex event pattern and event-processing workflows are subscribed at the middleware. From that point onwards, incoming streams of primitive events (2) are continuously analyzed by the event-processing component (Esper) in order to detect the

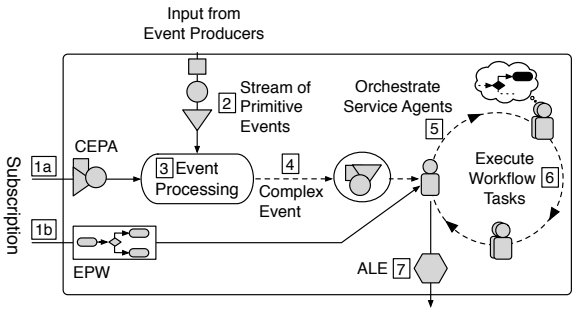

Figure 4. SOLA Agent-Based Middleware

registered patterns within the incoming streams of primitive events (3).

Once a match has been detected, a complex event is generated (4) and handed over to an orchestration agent, which subsequently executes the corresponding event-processing workflow (5) by activating service agents capable of executing the specified workflow tasks (6). Besides the effects of the workflow tasks themselves (e.g., a message sent to a third party), the result of this process may also be an application-level event transmitted to applications (7).

Due to its flexible, agent-based design, the middleware can easily be deployed in different scales. An industry application may require a distributed setting in order to process thousands of reads per minute, whereas reading events in smart home environments only occur once in a while and hence lightweight, localized middleware deployments may be more appropriate. Moreover, the middleware can easily be extended to incorporate new functionalities by simply adding new types of service agents and referencing them within the event-processing workflows. This way, the middleware is (runtime) adaptable to changing conditions once requirements, standards, or best-practices evolve in the future.

B. SOLA Discovery Service

According to the EPCglobal architecture standards, object information will be stored in globally distributed EPCIS repositories [21], which themselves will be discovered by using a DNS-based Object Naming Service (ONS) [34] and – potentially multiple – discovery services [22]. The ONS is responsible for class-level resolution of EPCs into EPCIS repository addresses of the object manufacturer, whereas discovery services will also provide serial-level resolution of multiple information providers. We emphasize that for a flexible, cross-domain discovery of complex event patterns and event-processing workflows for individual objects, a discovery service is needed. Due to the fact that currently no EPCglobal specification or implementation of a discovery service exists, we developed a P2P-based discovery service (see [26]), which offers secure and robust serial-level resolution. This SOLA discovery service can easily function as a lightweight discovery service for multiple information providers per object, as will be described later, and involves the following main ideas: Each interested company deploys dedicated nodes, in addition or instead of ONS servers. Those nodes form an *overlay network* based on a *Distributed Hash Table (DHT)* [35]. A cryptographic hash function maps EPCs and nodes to overlay IDs. This pseudorandom mapping of identifiers to storage nodes balances load more evenly, allows for easy replication, avoids single points of failure, and reduces the feasibility of targeted attacks against specific information providers or clients.

The DHT provides the routing to the responsible nodes, as well as joining, leaving, repair, and optimization procedures, without a central entity managing those operations. Nodes store deterministically assigned, but from an outside perspective apparently random, encrypted, and digitally signed documents belonging to EPCs mapped to certain hash value ranges. Those documents may contain either EPCIS repository addresses or even object data. Hence, following the data-on-network strategy, the complex event patterns and event-processing workflows may be stored on the nodes, too, and can directly be retrieved without additionally querying an EPCIS repository. In our current approach, however, we comply with the EPCglobal model [1] of using a separate EPCIS repository, in our case a cloud infrastructure, to retrieve the actual data.

C. SOLA Cloud EPCIS

In order to facilitate high speed access and scalability, we developed a solution for physically distributing data from EPCIS repositories [36]. It is inspired by content distribution networks but tailored to event data and the IOT. Experiments reported in [36] show the impact of application level network delay in fetching globally distributed EPCIS data. In our solution, we minimize network delay through physically distributed EPCIS mirrors called *cloud EPCIS*. Central to our solution are mechanism for event data distribution and request redirection.

The mechanism for data distribution is in charge of proactively pushing RFID events from the corresponding EPCIS repositories (*origin EPCIS*) to cloud EPCIS. Each origin EPCIS can be associated with cloud EPCIS at various physically distributed locations. The goal is to have RFID events available in close proximity to the origin of future requests. The mapping between origin EPCIS and cloud EPCIS can be optimized with different strategies. A default strategy is to choose one cloud EPCIS per continent. However, background knowledge about the application domain allows the user to limit cloud EPCIS to certain regions of interest in the given value chain. The distribution mechanism follows the observer design pattern and waits for new RFID events at origin EPCIS. On the arrival of new events the distribution mechanism pushes the RFID events into all associated cloud EPCIS.

The service for request redirection directs user requests for RFID data to the closest corresponding cloud EPCIS. It thereby ensures that the requested RFID data are recalled with low network delay. The redirection mechanism sets up on our implementation of the discovery service and the discovery service proxy. For redirection, we register all cloud EPCIS along with the origin EPCIS in the DHT. In addition, we store information about the location of each cloud EPCIS. When the discovery service proxy requests the EPCISs for a given EPC it filters the responses in accordance to its own physical location. The client receives a list of relevant EPCIS repositories where each repository is represented by the closest corresponding cloud EPCIS thereby minimizing network delay in communication with the EPCIS repository.

VI. Experiment

In this section, a proof of concept for SOLA as well as performance results for the individual components is presented. The scenario described in Section III serves as underlying application scenario.

In order to get realistic event data, an OMNet [37] event simulation was conducted assigning EPCs with read event timestamps. The simulation considers two interacting partners: the manufacturer located in Asia (Singapore) and the wholesaler distribution hub located in EU (Berlin). An overview of the individual components is given in Table I. As far as the individual machines are concerned, the Amazon EC2 mirco instances had 613 MB of RAM, up to 2 EC2 Compute Units (1 EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor), and ran a Ubuntu 10.4 64 bit operating system. The notebook had 3 GB of RAM, a Core 2 Duo 1.6 GHz CPU, and ran Ubuntu 10.4, 32 bit operating system. The virtual machines at HU Berlin had 512 MB of RAM, 1 virtual core (equivalent to a Xeon X5460 core with 3,166 GHz), and ran Ubuntu 10.4, 32 bit operating system.

As reference measuring point, the wholesaler distribution hub was used where the performance of (i) the wholesaler distribution hub middleware, (ii) the performance of the cloud EPCIS and origin EPCIS of the manufacturer, when queried from the wholesaler distribution hub middleware, and (iii) the performance of the discovery service, were measured. Figure 5 shows the system load at the online pharmacy distribution hub, i.e., when objects are scanned by the reader. The figure indicates that 480 events were simulated to arrive in 550 s in batches of about 20 objects. The initial bootstrapping procedures were excluded. Accordingly, the results are an 550 seconds slice from a system in operation.

Figure 6 shows the performance results of the discovery service when queried from the wholesaler distribution hub. The median of the response time was 933 ms and the average response time was 920 ms. The fastest response took 135 ms and the slowest response 2164 ms.

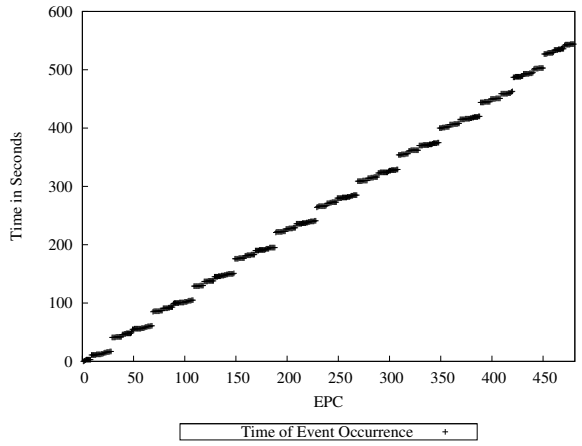

Figure 5. Event Schedule

Table I
Experiment Overview of IoT Services

Component	Location	Machine
Manufacturer origin EPCIS	Asia (Singapore)	Amazon EC2 micro instance
Wholesaler distribution hub middleware	EU (Berlin)	Notebook
Cloud EPCIS Europe	EU (Ireland)	Amazon EC2 micro instance
Discovery service node 1	EU (Berlin)	VM HU-Berlin
Discovery service node 2	EU (Berlin)	VM HU-Berlin

As far as the performance of the wholesaler distribution hub middleware is concerned (see also Figure 6), the median of the processing time was 52 ms, the average 57 ms; the longest processing step toke 383 ms and the shortest 28 ms.

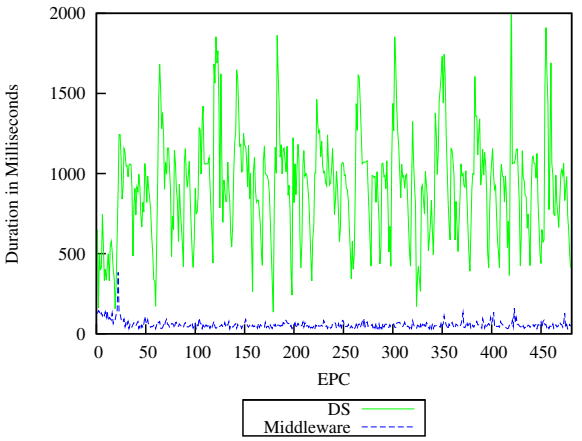

Figure 6. Discovery Service and Processing Service Performance

In Figure 7 the performance of the processing service request to the cloud EPCIS in EU and to the manufacturer's

origin EPCIS in Asia is displayed. The median response time of the cloud EPCIS was 169 ms, the average response time 267 ms. The slowest response took 2051 ms and the fastest 95 ms. In contrast, the response time for the origin EPCIS in Asia was on average 893 ms, the median was 848 ms. The slowest response took 2969 ms and the fastest response 520 ms. The performance difference between the cloud EPCIS and the origin EPCIS was 625 ms on average.

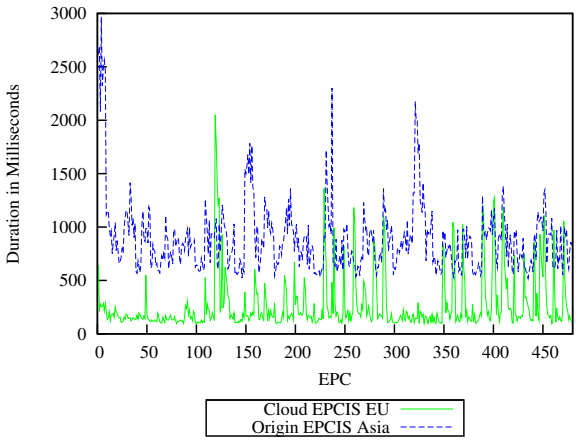

Figure 7. EPCIS and Cloud EPCIS Performance

All in all (see Figure 8), it can be stated that the average total round trip time (discovery, retrieval, and processing) for one EPC in SOLA was 1869 ms for the origin EPCIS in Asia and 1245 ms for the cloud EPCIS in EU, while the median was 1849 ms and 1196 ms, respectively. The fastest round trip with the origin EPCIS in Asia was 736 ms; the slowest 3486 ms. For the round trip with the cloud EPCIS in EU, the results were 315 ms and 3812 ms, respectively.

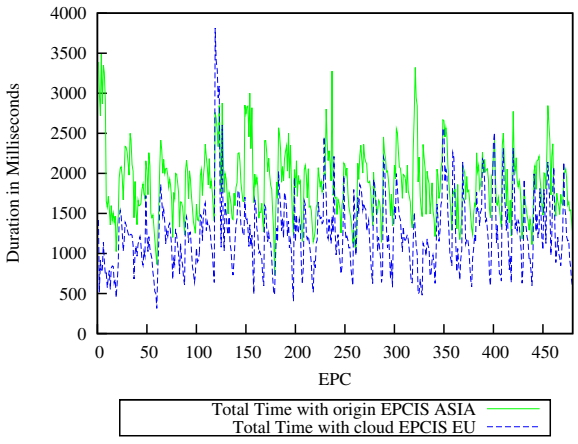

Figure 8. Total Response Time

VII. Discussion and Future Work

The SOLA middleware has been designed for processing massive amounts of events when deployed in a distributed setting [32]. The main building blocks of the middleware, the underlying agent system *Jadex v2* [38], and the complex event processing engine *Esper* [30], have proven to be high-performing in our experiment.

As far as the discovery service is concerned, special emphasis was placed on designing a robust and flexible solution that avoids bottlenecks, can handle XML-based web service and ONS requests on the inside, and can actually connect to several different DHTs on the outside, if necessary. We investigated latency as part of overall efficiency for our discovery service. Experiments indicate that the P2P discovery service will be able to provide query results with an average latency of 920 ms.

The implementation of cloud EPCIS draws on concepts of content delivery networks and reduces application level network delay in globally distributed value chains [36]. The proposed approach is in line with existing EPCglobal standards and transparent to the end user. We showed that the usage of the cloud EPCIS leads to a performance improvement of 625 ms on average compared to the origin EPCIS.

In future work, we plan to extent the performance test for domain specific scenarios and include more stakeholders. Further, a deployment of the middleware on android mobile phone platfrom is planned in order to include end users in the SOLA experiments. Providing comprehensive and decentralized security mechanisms will as well be part of our future research.

VIII. Impact on Interorganizational IT Infrastructures

SOLA facilitates the use of smart objects in interorganizational settings, where technological choices interrelate with organizational aspects. The provided functionality and architectural properties of SOLA open up new options for collaboration and business models. In particular, SOLA impacts (i) the provisioning of object related data, (ii) the discovery of object related data sources, and (iii) the management of object related processing logic.

Provisioning of object related data: SOLA proposes cloud EPCIS for data provisioning. This opens up opportunities for intermediaries who offer cloud EPCIS as third party service. Our experiments in previous work have shown that cloud EPCIS can benefit from scale effects and statistical multiplexing [39]. These effects enable cloud providers to run cloud EPCIS at lower cost than supply chain participants could do on their own. Hosting cloud EPCIS is therefore an appealing business model that SOLA enables for cloud providers.

Discovery of object related data sources: Discovery of object related data sources plays a key role in the

flexible exchange of object data. EPCglobal has proposed the ONS standard for this purpose. However, this standard is greatly debated for its centralized design and the resulting implications on control over this critical part of the system [40]. The SOLA discovery service uses a P2P-based design that is completely decentralized. It thereby mitigates the problem of placing a key component under control of a single player.

Management of object related processing logic: SOLA uses an agent-based middleware and processing rules in EPCIS to handle escalations in the object lifecycle. This design presents a fundamental shift in managing processing logic at the point of operation. SOLA decouples the control over processing logic from the physical location of the object by storing rules in the EPCIS. Its design can enable different players in the value chain to influence the object throughout the whole object lifecycle and enable the development and execution of fine-grained business rules, e.g., for supply chain escalation management.

IX. Conclusion

In this paper, we presented SOLA, an architecture that enables smart object lifecycle management across different domains and includes mechanisms for decentralizing data control and for accelerating data access.

We believe that the presented concepts will enhance the business impact of the emerging Internet of Things and Services, specifically by adding process logic to the flow of objects and data. In particular, our completely decentralized architecture allows objects to influence the processes they are taking part in, e.g., their own lifecycle. In addition, our solution includes novel components and architectural concepts that enhance different aspects of communication in the Internet of Things. That is, we provide (i) a scalable discovery service with a decentralized control structure, (ii) cloud EPCIS which facilitate fast and standard conform access to RFID data around the globe, and (iii) a middleware that enables flexible adaption of object specific event-processing rules. We strongly believe that the presented solutions and underlying concepts can aid development of a broad range of applications in the Internet of Things.

References

[1] EPCglobal, "The EPCglobal Architecture Framework – Version 1.3," March 2009. [Online]. Available: http://www.epcglobalinc.org

[2] N. Minar, M. Gray, O. Roup, R. Krikorian, and P. Maes, "Hive: Distributed Agents for Networking Things," in *ASA/MA*, 1999, p. 118.

[3] G. Li, V. Muthusamy, and H. Jacobsen, "A Distributed Service-oriented Architecture for Business Process Execution," *ACM Transactions on the Web (TWEB)*, vol. 4, no. 1, pp. 1–33, 2010.

[4] S. Haller, S. Karnouskos, and C. Schroth, "The Internet of Things in an Enterprise Context," in *First Future Internet Symposium, FIS 2008*, ser. LNCS, vol. 5468. Springer, 2009, pp. 14–28.

[5] W. van der Aalst, M. Rosemann, and M. Dumas, "Deadline-based Escalation in Process-aware Information Systems," *Decision Support Systems*, vol. 43, no. 2, pp. 492–511, 2007.

[6] M. Adams, A. Hofstede, D. Edmond, and W. van der Aalst, *Worklets: A Service-oriented Implementation of Dynamic Flexibility in Workflows*, ser. LNCS. Springer, 2006, vol. 4275, pp. 291–308.

[7] M. Reichert and P. Dadam, "ADEPT flex – Supporting Dynamic Changes of Workflows Without Loosing Control," *Journal of Intelligent Information Systems*, vol. 10, no. 2, pp. 93–129, 1998.

[8] M. Weidlich, G. Decker, A. Großkopf, and M. Weske, *BPEL to BPMN: the Myth of a Straight-Forward Mapping*, ser. LNCS. Springer, 2008, vol. 5331, pp. 265–282.

[9] W. Wang, J. Sung, and D. Kim, "Complex Event Processing in EPC Sensor Network Middleware for both RFID and WSN," in *11th IEEE ISORC*, 2008, pp. 165–169.

[10] L. Dong, D. Wang, and H. Sheng, "Design of RFID Middleware Based on Complex Event Processing," in *IEEE CIS*, 2006, pp. 1–6.

[11] N. Dindar, Ç. Balkesen, K. Kromwijk, and N. Tatbul, "Event Processing Support for Cross-Reality Environments," *IEEE Pervasive Computing*, vol. 8, no. 3, pp. 34–41, 2009.

[12] K. Aberer, M. Hauswirth, and A. Salehi, "Global Sensor Networks," EPFL, Lausanne, Tech. Rep., 2006.

[13] J. Morrell, "Complex Event Processing and RFID," *RFID Journal*, February 2007. [Online]. Available: http://www.rfidjournal.com/article/articleview/2991/1/82/

[14] J. Foley and G. Churcher, "Applying Complex Event Processing and Extending Sensor Web Enablement to a Health Care Sensor Network Architecture," in *London Communications Symposium*, 2009.

[15] M. Botts, G. Percivall, C. Reed, and J. Davidson, *OGC® Sensor Web Enablement: Overview and High Level Architecture*, ser. LNCS. Springer, 2008, vol. 4540, pp. 175–190.

[16] J. Odell, "Why Agent Technology for Event Processing?" http://tibcoblogs.com/cep/2010/03/16/why-agent-technology-for-event-processing-by-james-odell-csc/, 2010.

[17] L. V. Massawe, F. Aghdasi, and J. Kinyua, "The Development of a Multi-Agent Based Middleware for RFID Asset Management System Using the PASSI Methodology," in *ITNG*, 2009, pp. 1042–1048.

[18] C. Emmersberger, F. Springer, and C. Wolff, "Location Based Logistics Services and Event Driven Business Process Management," in *Intelligent Interactive Assistance and Mobile Multimedia Computing (IMC)*, 2009.

[19] R. von Ammon, C. Emmersberger, T. Ertlmaier, O. Etzion, T. Paulus, and F. Springer, "Existing and Future Standards for Event-Driven Business Process Management," in *3rd ACM DEBS*, 2009, p. 24.

[20] C. Zang and Y. Fan, "Complex Event Processing in Enterprise Information Systems Based on RFID," *Enterprise Information Systems*, vol. 1, no. 1, pp. 3–23, 2007.

[21] EPCglobal, "EPC Information Services (EPCIS) Version 1.01 Specification," September 2007. [Online]. Available: http://www.epcglobalinc.org

[22] BRIDGE, "High Level Design for Discovery Services," August 2007. [Online]. Available: http://www.bridge-project.eu/

[23] C. Kürschner, C. Condea, O. Kasten, and F. Thiesse, "Discovery Service Design in the EPCglobal Network – Towards Full Supply Chain Visibility," in *Internet of Things (IOT 2008), Zurich*, 2008, pp. 19–34.

[24] A. Rezafard, "Extensible Supply-chain Discovery Service Problem Statement," 2008. [Online]. Available: http://tools.ietf.org/html/draft-rezafard-esds-problem-statement-03

[25] M. Young, "Extensible Supply-chain Discovery Service Concepts," 2008. [Online]. Available: http://tools.ietf.org/html/draft-young-esds-concepts-04

[26] B. Fabian, "Implementing Secure P2P-ONS," in *Proceedings IEEE International Conference on Communications (IEEE ICC 2009), Dresden*, 2009.

[27] N. Schönemann, K. Fischbach, and D. Schoder, "P2P Architecture for Ubiquitous Supply Chains," in *ECIS*, 2009.

[28] Akamai, "Akamai CDN." [Online]. Available: http://www.akamai.com

[29] Object Management Group, "Business Process Model and Notation (BPMN) 1.2," 2009. [Online]. Available: http://www.omg.org

[30] EsperTech, "Esper – Reference Documentation," 2009. [Online]. Available: http://esper.codehaus.org

[31] S. F. Wamba and H. Boeck, "Enhancing Information Flow in a Retail Supply Chain Using RFID and the EPC Network," *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 3, no. 1, pp. 92–105, 2008.

[32] D. Bade, "Towards an Extensible Agent-based Middleware for Sensor Networks and RFID Systems," in *Proceedings of the 3rd Int. Workshop on Agent Technology for Sensor Networks (ATSN-09)*, 2009.

[33] D. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley, 2002.

[34] EPCglobal, "Object Naming Service (ONS) 1.0.1," EPCglobal, May 2008. [Online]. Available: http://www.epcglobalinc.org

[35] H. Balakrishnan, M. F. Kaashoek, D. R. Karger, R. Morris, and I. Stoica, "Looking up Data in P2P Systems," *Communications of the ACM*, vol. 46, no. 2, pp. 43–48, 2003.

[36] H. Ziekow, B. Fabian, and C. Müller, "High-Speed Access to RFID Data: Meeting Real-Time Requirements in Distributed Value Chains," in *Proceedings of the Confederated International Workshops OTM 2009*, ser. LNCS, vol. 5872. Springer, 2009, pp. 142–151.

[37] OMNeT. (2011) OMNeT++ Simulation Library and Framework Web Site. OMNeT. [Online]. Available: http://www.omnetpp.org

[38] A. Pokahr, L. Braubach, and W. Lamersdorf, "Jadex: A BDI Reasoning Engine," *Multiagent Systems Artificial Societies and Simulated Organizations*, vol. 15, pp. 149–174, 2005.

[39] H. Ziekow, B. Fabian, O. Günther, and C. Müller, "RFID in the Cloud: A Service for High-Speed Data Access in Distributed Value Chains," in *Proceedings of the Sixteenth Americas Conference on Information Systems (AMCIS 2010)*, 2010.

[40] S. Evdokimov, B. Fabian, and O. Günther, "Multipolarity for the Object Naming Service," in *Proc. Internet of Things (IOT 2008), Zurich, Switzerland, 2008*, ser. LNCS 4952. Springer-Verlag, Berlin-Heidelberg, 2008, pp. 1–18.

ACKNOWLEDGEMENTS

This research was funded by the German Federal Ministry of Education and Research under grant number 01IA08001E as part of the Aletheia project. The responsibility for this publication lies with the authors.