Incorporating business process management into RFID-enabled application systems

Xiaohui Zhao
Information Systems Group, Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, Eindhoven, The Netherlands

Chengfei Liu
Faculty of Information and Communication Technologies, Centre of Complex Software Systems and Services, Swinburne University of Technology, Melbourne, Australia, and Tao Lin
Amitive Inc., San Mateo, California, USA

Abstract
Purpose – The emergence of radio frequency identification (RFID) technology promises enormous opportunities to shift business process automation up to the wire level. The purpose of this paper is to explore the methodology of incorporating business logics into RFID edge systems, and thereby facilitate the business process automation in the RFID-applied environment.

Design/methodology/approach – Following the object-oriented modelling perspective, concepts of classes, instances are deployed to characterise the runtime context of RFID business scenarios; event patterns are used to aggregate RFID tag read events into business meaningful events; and business rules are established to automate business transactions according to the elicited events.

Findings – The paper has emphasised the synergy between business process automation and automatic data acquisition, and has identified the inter-relations between RFID tag read events, application-level events, business rules, and business operations. The reported research has demonstrated a feasible scheme of incorporating business process control and automation into RFID-enabled applications.

Originality/value – The paper analyses the characteristics of RFID data and event handling in relation to business rule modelling and process automation. The features of event-relied awareness, context containment and overlapping, etc. are all captured and described by the proposed object-oriented business model. The given data-driven RFID middleware architecture can serve as one reference architecture for system design and development. Hence, the paper plays an important role in connecting automatic data acquisition and existing business processes, and thereby bridges the physical world and the digital world.

Keywords Radio frequencies, Automation, Process management

Introduction
It is dramatically important to assess the strategic business value of integrating radio frequency identification (RFID), a key connective technology, into enterprise system applications (Want, 2004; Mo et al., 2009; Roussos and Kostakos, 2009). Nowadays, the
increasing adoption of RFID is evidenced in retailing, manufacturing, supply chain, military use, health care, etc. particularly in the background of business globalisation commoditisation (Koh, 2007; Kumar, 2007). An optimistic forecast from IDTechEx (2007) expects that the total RFID market value (including all hardware, systems, integrations, etc.) across all countries will rise from $4.96 billion in 2007 to $26.88 billion in 2017.

Current RFID technology has been established with emphasises on network infrastructure, data/event communications, and information sharing (Banks et al., 2007; Chalasani and Boppana, 2007; Chen et al., 2007; Lin et al., 2007). With the RFID facilitating equipments and systems, such as reading devices, RFID edge systems, federated RFID information service systems, etc. enterprises are enabled to automatically sense and react to the real world (Asif and Mandviwalla, 2005; Lehtonen et al., 2007; Wijngaert et al., 2008). Yet, how such awareness improves the business effectiveness and efficiency is subject to the extent that how the RFID systems are integrated to the existing enterprise backend application systems and how these backend systems utilise the collected RFID data. In practice, this integration can be technically interpreted into how to incorporate business process management (BPM) into RFID systems, and how they together integrate people, information, processes, and products across traditional organisation and application boundaries.

BPM and RFID

Attempts to apply RFID in facilitating real enterprises’ business running are often made by monitoring the material flows through reading events or collecting data from physical world. However, most of these solutions are lack of BPM support. This barriers RFID-enabled applications from the benefits of on-site responses according to business logics, system agility against changing requirements, seamless integration with enterprise application systems, etc. Business processes blend business logics and related resources into reusable models for efficient transaction management. Once such business logics are incorporated into RFID systems, it can provide real-time information in a network consisting of different enterprises, applications, and business partners in collaboration scenarios. Consequently, such integration can enable automatic reactions and execution based on the captured real-time information while eliminating manual inputting, processing, and checking of information (Lefebvre et al., 2006; Liu et al., 2009). As a milestone towards such integration. Boeing has already embarked on a new assembly paradigm that is focused on using RFID and collaborative processes for its 787 (formerly 7E7 Dreamliner) aircraft to reduce the assembly time (Gillette, 2004). The integration of RFID and business logics seeks methodological advances and facilitating architectures to merge the wire-level deployment and business-level applications under one umbrella.

From the technical perspective, Figure 1 shows a high-level overview of an RFID solution architecture suggested by Microsoft (2004). This reference architecture primarily classifies five logical layers at a conceptual level, which depicts an ideal integration from RFID wire level to enterprise application level. In this architecture, layer 0 consists of the RFID reading devices and some on-site functioning equipments on the “edge” of the RFID enabled systems; layer 1 consists of the basic operating environment and platform of the RFID-enabled solution architecture. At this layer, the RFID data receives the primitive processing, such as cleansing, aggregating, filtering, etc.;
layer 2 and the layers above it are essentially the enablers of the business processes and solutions that can leverage the real-time data generated by lower layers to drive business activities; layer 3 provides the encapsulated services that are implemented as the “abstraction” of the layer(s) below it, and can be implemented as application programming interface services and/or web services; layer 4 leverages the services, data and tools provided by the layers below to implement application solutions.

Our work mainly targets at incorporating business process control and automation into the RFID-enabled applications. To achieve this objective, we propose to create a middleware system lying across some components on layers 2 and 3, as highlighted by shadows in Figure 1. The middleware system is aimed to connect and mediate the BPM and the edge-level devices, such as forklift cars, hand-picking or automatic-picking devices, etc. A novel process model is proposed to describe the behaviours of the tagged objects and the inter-relation between them from an object-oriented perspective. This process model takes into account blending RFID tag read events and event patterns into business logic modelling, and thereby enables the awareness to the runtime dynamics of the environment. By customising the deployed business rules and data schema, the middleware system can easily be updated to adapt to new requirements with the same edge infrastructure. This feature considerably enhances the system agility and flexibility and reduces the cost of adopting RFID technologies to existing legacy application systems.

In the remainder of this paper, we first review the industry products and research works related to RFID data/event management and context-sensitive middlewares, as well as the object-oriented business modelling perspective and RFID’s impact on BPM. After that, we investigate a real RFID application to explore how RFID technology influences the data management in business process automation. Following that, we present an object-oriented business model and discuss its deployment to real RFID applications. A data-driven middleware architecture design is introduced to guide the implementation the proposed approach. Finally, we conclude this paper and give an indication of future work.
Literature reviews and discussions

Industry standards and products

As the pivotal advocator of RFID technologies, electronic product code (EPC) global has established a series of standards covering hardware-level identification, event capture and information exchange in the RFID-applied environment. Particularly, EPCglobal has proposed a reference architecture framework including Object Name Service, EPC Information Services, Application Level Events, etc. This framework depicts the primary constitution of RFID systems, and lists the basic interfaces for interoperability of RFID applications.

Major software vendors, like IBM (WebSphere RFID), Sun (Java RFID System), BEA (BEAWebLogic RFID), Oracle (Sensor Edge Sever), Microsoft (BizTalk Server RFID), Sybase (RFID Anywhere), etc. have been extending their application development and middleware technology stacks to handle RFID. Yet, these middleware systems mainly emphasise on the RFID data filtering, cleansing and storage, while treat the connection to business application systems as a side objective. SAP has run an Auto-ID Infrastructure project to facilitate the connection between RFID devices, middleware systems, and business application systems (Bornhövd et al., 2004; Götz et al., 2006). This infrastructure covers the edge systems and application systems and puts the devices, backend systems and human staff under one umbrella. The core services of the infrastructure contain two important components, i.e. the association data management (ADM) and the action and process management (APM). ADM is designed to manage the contextual information and the received events, and APM is to navigate the activity handling according to the pre-defined rules.

RFID data/event management and context-sensitive middlewares

A lot of research efforts have been put to tackle RFID complex event processing, yet most of them mainly focus on data cleansing and filtering. Work “Stream-based And Shared Event” (SASE) processing has defined an SQL like complex event language to aggregate RFID events (Wu et al., 2006; Gyllstrom et al., 2007). The implemented SASE system uses a persistence storage component to support querying over historical data and to allow query results from the stream processor to be joined with stored data. In addition, the extended sliding window control and indexing techniques have been adopted by Bai et al. (2007) and Park et al. (2007) to improve the performance of continuous query processing over RFID event flows. Hu et al. (2006) have addressed the query issue from the perspective of energy efficiency. Wang and Liu (2005) have investigated the temporal management of RFID data. They have adapted traditional database query techniques to the temporal relationships of RFID data, and thereafter defined a set of temporal complex event constructors in their follow-up work (Wang et al., 2006). Two partitioning mechanisms have also been proposed in their work to support efficient queries. However, none of the mentioned works have provided an explicit solution on how to handle the delayed effects in event management, or how to integrate business process automation into RFID event management.

In regard to the semantic elicitation from sensor-captured data, context-sensitive middlewares have attracted quite some research efforts. Ranganathan and Campbell (2003) have proposed a middleware to allow heterogeneous agents to acquire contextual information and uniform the knowledge using ontologies. Yau et al. (2002) have built a reconfigurable context-sensitive middleware for developing
context-sensitive pervasive computing softwares and their runtime operations. In service computing area, Gu et al. (2005) have developed a service-oriented middleware to support the acquisition, discovery, interpretation and accesses of various contexts to build context-aware services. These middleware systems are mainly designed for general sensors, and therefore they fail to fully address the various characteristics of RFID technology. For RFID, Kim et al. (2007) have reported a business aware framework for business processes in the EPC Network. This framework allocates its business aware layer between the traditional RFID data middleware and upper applications, and the framework mainly focuses on the conversion of RFID raw events to business events, and the invocations to upper-level business services.

Object-oriented business modelling
To support business process modelling in data-intensive business scenarios, the object-oriented (or artifact-oriented) perspective has been proposed recently as a new business modelling method (Nigam and Caswell, 2003; Bhattacharya et al., 2007; Küster et al., 2007; Liu et al., 2007; Hull, 2008; Wahler and Küster, 2008). This perspective uses objects to denote the information entities that capture process goals and allow for evaluating how thoroughly these goals are achieved. Business rules are used to assemble the services together, and define how the entities respond to the real-time dynamics. Compared with traditional business process modelling approaches, the object-oriented modelling approach focuses on the business contexture and behaviours, rather than the sequencing of activities. Therefore, the object-oriented modelling advocates a complete data-driven execution mechanism, and thereby enables business actors to be aware of what can be done instead of what should be done.

RFID’s impacts on BPM
As a promising data sensing/collecting mechanism, RFID brings a lot of operational benefits to business sectors, including manufacturing, healthcare, transportation, defense, retail and agriculture, where requires accurate data or more collection points, according to Christine Overby of Forrester Research, Boston, MA, USA. For the general BPM in these sectors, RFID provides the ability of tracking goods moving through the supply chain. This makes it possible to shorten the order-to-cash cycle, detect and resolve delivery exceptions, prevent out-of-stock situations, and pinpoint-affected product in a recall, while minimising inventory and safety stock levels.

This real-time visibility is expected to benefit BPM in following aspects, where new business values are highly sought after:

- Possibility to handle the business onsite. This can significantly enhance the operational efficiency.
- Improved productivity of business processes. For example, RFID might be used to simultaneously read all of the cartons on a pallet as it passes through a portal, or read all of the serial numbers virtually at once as a pallet of goods leaves a production cell.
- Improved sensitivity of business intelligence. Real-time visibility supports vendor-managed inventory programs, helps prevent shrinkage and diversion, and discourages counterfeiting by making it easier to identify fake products. End-to-end visibility also supports the record keeping needed for e-pedigree tracking for the pharmaceutical industry.
In next section, we will discuss the features of RFID applications and how it influences the data handling of BPM by investigating a distribution centre (DC) scenario from a technical view.

**Investigating RFID applications**

Let us examine an RFID-enabled enterprise application. As a typical chain of logistics processes, a DC is often selected to discuss RFID deployment (Bottani, 2008), since a distributed centre assembles a large volume of shipments every day using pallets. Here, we also use this case to illustrate the nature of an RFID-enabled application.

As shown in Figure 2, this packing process includes steps of receiving orders, picking goods, pallet packing, and shipment assembling. After receiving an order from customers, pickers begin to pick the ordered goods from the inventory, and the forklift cars begin to transfer the pallets and the picked goods to the assembly spot. Thereafter, the goods are packed onto pallets at an assembly line, and finally the packed pallets are loaded to proper trucks for shipment.

This distributed centre represents a typical RFID-"rich" environment, where products, pallets, equipments, and tools are all attached with RFID tags. The deployed readers in inventories, packing lines, transporting vehicles, etc. constitute a network that monitors the movements of products, pallets, vehicles, etc. In this scenario, using RFID with the goods-packing process in a distributed centre can bring the following benefits:

- Improve the visibility of the status of a business transaction. Given the planned picking time for a shipment, the latest time for packing process can be estimated. With RFID technology, which packing processes have started and which have not can be seen clearly. Therefore, early warnings for potential late shipments can be easily discovered.

- Improve the content accuracy for shipments.

- Record association hierarchy (association relationship of serialised cases with the corresponding pallet, pallets with the corresponding shipment). This information can be used for validating shipments and also tracking recall goods or counterfeit goods.

- Recover recall or counterfeit goods. Picking and packing are the stages in a distributed centre where cases still remain as individual. The systems in
distributed centre will check whether the goods in distributed centre are of those types. If so, picking or packing systems will hold the serialised identification of those products and inform operators to pick those products when reads them.

RFID provides the real-time object-level information for business processes to execute in a prompt and precise manner. At the same time, RFID-enabled applications also bring the following distinct characteristics from traditional applications:

- Activities are triggered by RFID data rather than human.
- RFID systems tend to generate a huge amount of tag event data, as the readers continuously report all pass-by objects.
- Movements of some RFID-tagged objects reflect swarming phenomena, as many RFID-tagged objects act with the similar behaviours, particularly in the packaging and transportation stages.
- The products of the same type and same batch may participate in different business processes, yet it is hard to pre-define the correlation between products and the involved business processes.

How RFID influences the data handling in BPM?
To align the collected object movement information with business, BPM methodologies have to adapt themselves to the aforementioned characteristics. In the RFID-applied environment, edge systems are responsible for the most on-spot business operations, while they monitor the object movements (Glover and Bhatt, 2006). Correspondingly, our deployment strategy is to enable the awareness of edge systems to be both real-time context and business logics. Therefore, the edge systems can react to the contextual dynamics with proper business operations intelligently, and as such the business efficiency and effectiveness can be improved.

In regard to facilitating business process automation with RFID technologies, Palmer (2004) has identified “digest the data close to the source”, “turn simple events into meaningful events”, and “cache context” as three principles for RFID data management in work. Basically, these principles emphasise the following aspects of data management:

- The raw RFID data should pass the cleansing, consolidation, and summarisation processes at the edge systems to ensure greater reliability and protect central IT systems from the flood of data.
- Turn simple RFID read events into meaningful events to derive actionable knowledge from discrete events.
- Further understand and process RFID event data in the specific business context with the cached reference data and related scenario context.

Our work targets at incorporating business process control and automation into RFID-enabled application systems with a novel object-oriented business model. Different from traditional business process modelling approaches, this model depicts a business scenario by means of characterising the contextual dynamics among related objects. To cater for the event-driven and data-intensive execution mechanism of RFID applications, this model defines event patterns and business rules to elicit the business
meaning from RFID tag read events. In addition, this model encapsulates the related classes, rules event patterns into deployable RFID application contexts. Once such RFID application contexts are deployed to the data-driven middleware systems, the business logics can be pushed down to RFID edge systems. Therefore, the RFID edge systems will become aware to both real-time object information and business logics, and thereby can intelligently respond to the tag read events with proper operations on spot. By changing the deployed RFID application contexts, the same edge systems and infrastructures are capable to support different applications. This feature enhances the flexibility and customisability of RFID application systems.

Object-oriented business modelling framework
To initiate the thinking on business process automation in RFID-applied environment, we pose two questions as follows:

1. How business processes help navigate edge systems to respond to the tag read events with proper operations immediately?
2. How to “inject” business logics to edge systems to help them filter tag read events intelligently?

Traditional activity-based workflow models architect business processes with the main focus on control flow dependencies, where activities execute in accordance to the pre-defined sequence rather than the dynamics of real-time business context. To adapt to the event-based communication and effectively utilise the real-time object information, we attempt to model business processes from an object-oriented perspective, and drive business process according to the contextual dynamics.

Object-oriented perspective and RFID-enabled applications
Object-oriented methodology has been widely used in modelling real-world applications. Here, we discuss the feasibility of applying the object-oriented perspective in modelling RFID-enabled applications against the classic object-oriented features.

Polymorphism. In the RFID-enabled application, an entity type may own multiple definitions and its method may own multiple implementations. Once deployed in a practical scenario, it will choose proper definition or implementation according to the actual situation. Take assembly line B in the mentioned DC scenario as an example. With the products of the same type, this assembly line will pack them from the centre of the pallet to balance the weight distribution; while with mixed products, this assembly line will pack them in proportional spacing to match different product sizes and balance the weight distribution. This reflects the function-level polymorphism.

Inheritance. In our object-oriented business model, an involved entity type can be easily extended or specialised by adding new methods or attributes. This feature is particularly useful when deploying the defined classes in a practical scenario. For example, the class of forklift car can be specialised with methods of loading and unloading pallets in the DC case; while it may be specialised with attributes of latest position, latest speed, etc. in the case of monitoring the locations of cars.

Encapsulation. Our object-oriented business model allows each class to encapsulate the implementation of its operations/methods. In addition, a composite class can encapsulate the related classes, involved event patterns and business rules into a new class. This new class hides the internal-handling details, and only provides the
necessary interfaces for interaction with other classes. This feature enhances the integrity of the entity, as well as reduces the system complexity by limiting the interdependencies between entities. The details of this feature will be discussed in the following section.

Following the object-oriented paradigm, we abstract the entities involved in a business scenario, such as products, equipments, staff, tools, related documents, etc. into classes. The following notions are used to characterise business processes in the RFID-applied environment:

- A class abstracts the attributes and behaviours of a kind of entities, while an object denotes an instance materialised with actual business data at run time.
- A set of event patterns elicit the business meaning from RFID tag read series.
- Using event patterns, a set of business rules define the behaviours of individual classes, and coordinate the orchestration between the collaborating classes.
- A run time application context consists of all the involved objects, related rules and the real-time event series into a dynamically interacting system.

In the RFID-applied environment, the continuous tag read event series reflects the movements of objects, while these movements can be interpreted into business meaningful events using event patterns. According to the business rules, the objects will response to these elicited events by updating their internal status or invoking external operations. Thereby, the business process is fulfilled in form of the interactions between these objects.

To accommodate to this object-oriented perspective, we reckon that the management over the deployed business processes should correspondingly conduct in a rule-based and data-driven manner. Technically, the behaviours of objects are subjected to the defined rule set, and therefore by setting up multiple rules sets we can support objects to serve multiple business processes at the same time. By redefining the rules, we can customise the application context, and thereby reconfigure the edge systems to adapt to new requirements.

**Object-oriented business model**

In this section, we present the details of our object-oriented business model for RFID-enabled applications. This model is based on our work on RFID event integration (Zhao et al., 2009) and follows the event-calculus formalism (Shanahan, 1999). First, we define some:

1. **Terms.** Event $e$, denotes the action of the system throwing a message. The event used in this model can be either raw RFID tag read event or system-generated event.

2. **Functions:**
   - $\text{type}(a)$ returns the data type of variable $a$; and
   - $\text{DOM}(tp)$ returns value range of data type $tp$.

3. **Predicates:**
   - $\text{changesTo}(c, e, s, t)$ – an instance of class $c$ changes to state $s$ after event $e$ at time point $t$;
• \textit{holdsState}(c, s, t) – an instance of class \( c \) holds state \( s \) at time point \( t \);
• \textit{invokes}(op) – invoke operation \( op \);
• \textit{occurs}(e, c, t) – event \( e \) occurs at time point \( t \), triggered by an instance of class \( c \);
• \textit{throws}(e, c, t) – an instance of class \( c \) throws event \( e \) to the communication bus at time point \( t \); and
• \textit{discards}(e, c, t) – drops event \( e \) from the communication bus at time point \( t \), triggered by an instance of class \( c \).

Our model is to use these terms, functions and predicated to define its key notions.

\textbf{Definition 1 (Base class).} A base class abstracts the behaviours, attributes of a type of entities, such as pallets, forklift cars, etc. Formally, a base class can be defined as tuple \((c, A, M, S, s, t, F)\), where:

• \( c \) is the class name;
• \( A \) is a set of attributes;
• \( M \) is the list of methods;
• \( S \) is a set of status;
• \( F \subseteq S \times S \) describes the state transition behaviours of the class;
• \( s \in S \) is the initial state, i.e. \( \forall x \in S, \neg \exists (x, s) \in F \); and
• \( t \in S \) is the terminating state, i.e. \( \forall x \in S, \neg \exists (t, x) \in F \).

Based on \( S \) and \( F \), each class can build up a state transition diagram (STD), which represents the stages in the lifecycle of its instances, while an object maintains this STD to indicate the progress stage. For example, Figure 3 shows the content of class “assembly line” and its corresponding STD.

\textbf{Definition 2 (Instance).} An instance represents a concrete object attached with an RFID tag. Formally, an instance \( i \) of class \( c \) is defined as tuple \((o, \mu, q)\), where:

\begin{verbatim}
| Assembly line :: Assembly line |
| - Number of available products |
| - Number of available pallets |
| - Line ID |
| - ... |
| + StandBy() |
| + Pause() |
| + Init() |
| + PackPallets() |
| + Stop() |
| + Continue() |
\end{verbatim}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{std_diagram.png}
\caption{Class “assembly line” and its STD}
\end{figure}
• $o$ is the identifier of $i$;
• $\mu: c.A \rightarrow \text{DOM(type } (c.A))$ assigns the attributes in $c.A$ to actual values; and
• $q \in c.S$ denotes the current state of the instance.

**Definition 3 (Event observation).** The action of observing an event raised by a reader or the system can be characterised as $\text{occurs}(e, c, t)$, which denotes that event $e$ occurs at time $t$, triggered by an instance of class $c$. In practice, events flow as a series, which consists of several event observations, such as, [...] $\text{occurs}(e_1, c_1, t_1), \text{occurs}(e_2, c_2, t_2), [...]$ where $t_1 < t_2$. We call such a flow of event observations as an event series.

**Definition 4 (Domain-specific predicate).** For the class instances (or objects) in a specific domain, some dedicated predicates are used to describe the relations between these objects. In this paper, we confine such domain-specific predicates to be first-order predicates. Table I lists some domain-specific predicates in the pallet-packing example.

Based on these predicates, we can define some derivation rules to further exploit the relations between objects:

\[
\text{inside}(A, C) \leftarrow \text{inside}(A, B) \land \text{inside}(B, C);
\]
\[
\text{above}(A, C) \leftarrow \text{above}(A, B) \land \text{above}(B, C);
\]
\[
\ldots
\]
\[
\text{after}(A, C) \leftarrow \text{after}(A, B) \land \text{after}(B, C);
\]
\[
\text{packed_in}(A, C) \leftarrow \text{packed_in}(A, B) \land \text{packed_in}(B, C);
\]

**Definition 5 (Event pattern).** Event patterns are used for eliciting the underlying meanings from tag read event series, and the pattern design is subjected to the business logics of the specific application. Here, we borrow the event-calculus (Shanahan, 1999) like formalism to define an event pattern $\varepsilon$ as:

\[
\varepsilon = \bigwedge_{i = 0}^{n} \left[ \left( \bigwedge_{j = 0}^{m} \exp_{ij} \right) \lor \exp_{i} \right],
\]

where:

$\exp_{ij}, \exp_{i} \in \{\text{occurs}(e, c, t), \text{holdsState}(c, s, t)\} \cup \{\text{domain-specific predicates}\}$.

<table>
<thead>
<tr>
<th>Predicates</th>
<th>Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial Topological</td>
<td></td>
</tr>
<tr>
<td>adjacent_to(A, B)</td>
<td>Object $A$ is next to object $B$</td>
</tr>
<tr>
<td>inside(A, B)</td>
<td>Object $A$ is inside object $B$</td>
</tr>
<tr>
<td>Directional</td>
<td></td>
</tr>
<tr>
<td>above(A, B)</td>
<td>Indicate the directional relationships between two objects.</td>
</tr>
<tr>
<td>below(A, B)</td>
<td>These relationships are particularly useful for inventory management and shelf organisation</td>
</tr>
<tr>
<td>Temporal</td>
<td></td>
</tr>
<tr>
<td>after(A, B)</td>
<td>Object $A$ is observed after the observation of object $B$ by the same reader</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td></td>
</tr>
<tr>
<td>packed_in(A, B)</td>
<td>Object $A$ is packed in object $B$</td>
</tr>
</tbody>
</table>

**Table I.**
Samples of domain-specific predicates
Definition 6 (Business rules). Business rules specify the behaviours of objects, such as state transitions and operation invocations. Syntactically, a business rule can be defined in the following two forms with different predicates:

(a) changesTo(c, e, s, t) ← |e|;
(b) invokes(op) ← |e|

where:
- $e$ is an event pattern;
- $|e|$ denotes the Boolean value of event pattern $e$; and
- predicates changesTo and invokes are defined in the beginning of this section.

Example 1 lists some involved event patterns and business rules in the pallet-packing scenario:

Example 1. Partial content of the pallet packing context

Class:
$AL$ – Assembly line.

Events:
Arrives – a product arrives to the assembly line;
sentOff – a pallet of packed products are sent off.

Counter:
emptyPallets – this counter records the number of empty pallets at the assembly line.

Rules:
(1.1) changesTo($AL$, sentOff, “wait for pallets”, t) ← occurs(sentOff, $AL$, t) ∧ emptyPallets = 0;
(1.2) invokes(“call for pallets”) ← holdsState($AL$, “wait for pallets”, t) ∧ occurs(Arrives, $AL$, t);
(1.3) changesTo($AL$, Arrives, “ready”, t) ← holdsState($AL$, “wait for pallets”, t) ∧ occurs(Arrives, $AL$, t) ∧ ¬ emptyPallets = 0;

Rule (1.1) specifies that the assembly line will change to state “wait for pallets”, if it has no empty pallets after sending off the packed pallets. Rule (1.2) specifies that the assembly line will request for new pallets, if it is in state “wait for pallets”, when new products arrive. Similarly, Rule (1.3) specifies the condition for the assembly line to transit from state “wait for pallets” to state “ready”. Owing to the space limit, we do not list the full set of rules. Subject to the nature and inherent logics of concrete applications, event patterns and business rules can be customised to adapt to different application scenarios.

Definition 7 (Composite class). A composite class encapsulates multiple classes with related rules into a new class, which inherits the attributes and behaviours of the constitute classes. Formally, a composite class can be defined as tuple $(C, P, R)$, where:

- $C$ is the set of constitute classes;
- $P$ is the set of involved event patterns; and
- $R$ is the set of business rules defined on $C$ and $P$. 
Such a composite class builds up a self-maintained unit, which owns better reusability by hiding the internal complexity, and thereby enhances the scalability of RFID system integration. For example, assembly lines and related pickers, forklift cars, drivers, and operators can join into a packing station. Such a packing station can handle the packing process from picking products, transferring pallets, and packing pallets. At conceptual level, we define a composite class to represent such combined entities. Figure 4 shows the content of composite class “packing station” and its STD, which composes the STDs of its component classes.

Definition 8 (RFID application context). An RFID application context stands for an actual environment where classes and business rules are deployed with real event series. Such an RFID application context \( SC \) can be defined as tuple \((T, R, E, OP)\), where:

- \( T \) is a finite set of RFID classes, including both base and composite classes, with distinct names such that every class referenced in \( T \) also occurs in \( T \);
- \( R \) is the set of business rules, which are defined on the classes in \( T \);
- \( E \) represents the real-time event series obtained through the reader network; and
- \( OP \) represents the set of operations provided by the edge systems and upper-level application systems.

An RFID application context corresponds to a self-containing and self-acting encapsulated entity which can invoke the operations of edge systems or external systems in response to the real-time events.

Figure 5 shows the relationships between the notions of the proposed model. The shadowed area represents the static part of the RFID application context, which consists of classes, rules, and event patterns. A composite class may contain both base class(es) and composite class(es), and a class can inherit the characteristics of another class by extending the latter. Event patterns can extract the business meanings from tag read events, and with event patterns the rules can define the conditions of state transitions or operation invocations. The unshadowed area represents the run time.

**Figure 4.** Composite class “packing station” and its STD
part, which describes the interactions between objects. The status and attribute values of an object indicate the progress stage in its lifecycle. An RFID reader sends reading events when it observes a pass-by RFID-tagged object.

**Deploying RFID application contexts**

We have conducted a pilot installation of the proposed model on a packing station at a DC to evaluate if and how our approach can support RFID application contexts in practice. The basics of this scenario have been introduced in the motivating example section. RFID sensors are fixed to the receive gates of the assembly line, and the tagging is done at item level. In this section, we use a counterfeit-checking process and the aforementioned pallet-packing process to illustrate the deployment in details.

A counterfeit product copies the RFID tag of a sold genuine product, and therefore it has the same tag ID with the genuine one. When it passes a reader, it can be identified by searching the tag IDs of previously sold products, and be picked out in the end. Example 2 lists the key content of the counterfeit checking context, which is to be deployed to the edge system at the assembly line:

**Example 2.** Partial content of the counterfeit checking context

- **$L$** – List of recall or counterfeit goods;
- **Class:**
  - $AL$ – Assembly line.
- **Events:**
  - $\text{counterfeitFound}$ – a counterfeit is found;
  - $\text{Arrives}$ – a product arrives to the assembly line.
- **Operation:**
  - $\text{Pick out}$ – pick out a counterfeit product.
- **Predicates:**
  - $\text{onList}(G, L)$ – product $G$ is on the list of recall or counterfeit goods.
Rule (2.1) indicates that if a product is detected to be on the counterfeit list on its arrival to the assembly line, then the assembly line will pick it out and signal a "counterfeitFound" event. If not, the assembly line will signal a "non-counterfeit" event, according to Rule (2.2).

In regard to the pallet packing context, Example 1 has introduced part of its content, and here Example 3 lists the enriched content of this context:

**Example 3.** Enriched content of the pallet packing context

**Classes:**
- $G$ – Product;
- $P$ – Pallet;
- $AL$ – Assembly line.

**Movement-related predicate:**
- $moveWith(G, P)$ – Product $G$ moves together with Pallet $P$.

**Events:**
- $Arrives$ – a product arrives to the assembly line;
- $sentOff$ – a pallet of packed products are sent off;
- $Packs$ – a product is packed in a pallet;
- $Unpacks$ – a product is unpacked from a pallet.

**Counter:**
- $emptyPallets$ – this counter records the number of empty pallets at the assembly line.

**Operation:**
- $Pack goods$ – pack a product to a pallet.

**Rules:**
1. $(1.1)$ $\text{changesTo}(AL, sentOff, \text{“wait for pallets”}, t) \leftarrow \text{occurs}(sentOff, AL, t) \land \neg emptyPallets = 0$;
2. $(1.2)$ $\text{invokes}(\text{“call for pallets”}) \leftarrow \text{holdsState}(AL, \text{“wait for pallets”}, t) \land \text{occurs}(Arrives, AL, t)$;
3. $(1.3)$ $\text{changesTo}(AL, Arrives, \text{“ready”}, t) \leftarrow \text{holdsState}(AL, \text{“wait for pallets”}, t) \land \text{occurs}(Arrives, AL, t) \land \neg emptyPallets = 0$;
4. $(1.4)$ $\text{changeTo}(G, Packs, \text{“Loaded”}, t) \land \text{packedIn}(G, P) \leftarrow \text{occurs}(Packs, G, t) \land \text{occurs}(Packs, P, t) \land \text{holdsState}(G, \text{“Unloaded”}, t) \land \neg \text{packedIn}(G, P)$;
5. $(1.5)$ $\text{changeTo}(G, Unpacks, \text{“Unloaded”}, t) \land \neg \text{packedIn}(G, P) \leftarrow \text{occurs}(Unpacks, G, t) \land \text{holdsState}(G, \text{“Loaded”}, t) \land \text{packedIn}(G, P)$;
6. $(1.6)$ $\text{moveWith}(G, P) \leftarrow \text{packedIn}(G, P)$;
7. $(1.7)$ $\text{discards}(Arrives, G, t) \leftarrow \text{occurs}(Arrives, G, t) \land \text{moveWith}(G, P)$;
8. $(1.8)$ $\text{invokes}(\text{“Pack”}) \leftarrow \text{occurs}(\text{non-counterfeit}, AL, t) \land \text{occurs}(Arrives, G, t) \land \neg emptyPallets = 0$.

...
Rules (1.1-1.3) have been explained in Example 1. Rules (1.4 and 1.5) denote the conditions for determining whether a product is packed in a pallet or unloaded from a pallet. Rule (1.6) derives a conclusion that a product moves together with a pallet after being packed in the pallet. Rule (1.7) denotes that if a product moving with a pallet, the location of the product is correlated to the location of the pallet. Rules (1.8) and (2.1 and 2.2) may act at the same time when a product arrives, yet events “counterfeitFound” and “non-counterfeit” can distinguish the cases of a counterfeit being found or not. Correspondingly, “Pick out” and “Pack” operations will be invoked, respectively. In this way, the pallet packing context and the counterfeit-checking context are coordinated.

These two examples are only used to demonstrate the deployment of business rules, and they do not list the full details of the contexts due to the space limit.

In our pilot installation, we set up a series of Boolean variables (fluencts) to characterise the status of objects. The querying engine is implemented on IBM Discrete Event Calculus (DEC) Reasoner (IBM-DEC, 2008) on Linux platform, with (relsat-2.02, 2006) as the propositional satisfiability solver. Correspondingly, the defined rules are rewritten in its encoding format as shown below:

```prolog
load foundations/Root.e
load foundations/DEC.e
fluent Waiting() 
fluent CallforPallets() 
predicate NoEmptyPallets() 

; content for rule (1.1) 
[time] Happens(sentOff(), time) & !NoEmptyPallets() & !HoldsAt(Waiting(), time) -> Initiates(sentOff(), Waiting(), time).

; content for rule (1.2) 
[time] Happens(Arrives(), time) & HoldsAt(Waiting(), time) & !HoldsAt(CallforPallets(), time) -> Initiates(Arrives(),CallforPallets(), time).
```

**System architecture**

As mentioned in the “Introduction” section, the event management layer and the BPM component lie between the edge systems and enterprise application systems. Following this scheme, we deploy the facilitating data-driven middleware system to couple RFID tag read events and system operations together and thereby connect and mediate the business process execution and the edge-level devices.

According to the full-spectrum architecture of RFID solution shown in Figure 1, SAP AutoID lab has proposed a system design of a data-driven middleware linking the received RFID read events from layer 1 and the advanced applications of layer 3 and upper. Figure 6 shows the conceptual architecture of the data-driven middleware and the linked systems (Bornhövd et al., 2004).

Based on this conceptual architecture, we enrich it into a detailed middleware system design by extending it with necessary components, in accordance to our object-oriented RFID model. In Figure 7, the shadowed part shows our middleware system design. This middleware system collaborates with Edge Systems, RFID Information Services, and Enterprise Applications. An RFID Edge System directly interacts with the real world. The RFID Information Services are deployed at both the
edge side and the enterprise side, and these services together construct a distributed on-demand repository of information related to individual RFID tags. The Enterprise Applications may include Enterprise Resource Planning, Product Lifecycle Management system, etc. Detailed introduction of these systems can be found from EPCglobal’s documents. The designed data-driven middleware receives RFID tag read events from the edge systems, maintains the RFID application context, and triggers proper operations according to the monitored dynamics of related RFID objects.

The explanations on these constitute components of the data-driven middleware system are given below:

- The Rule Repository stores event patterns, business rules, and domain-specific predicates. These data can be used to generate queries to elicit business meanings from large volumes of RFID tag read events, derive out implicit state transitions, and determining operation invocations.
- The Runtime Database maintains the real-time information of objects, including the status and the attribute values of class instances.
- The Class Database maintains the static information of classes, such as the class definitions and the inheritance/reference relations between classes.
- The Event Manager acts as the main driver of the middleware system. At one hand, it receives the tag read series from RFID edge systems; at the other hand, it generates proper queries using the event patterns, predicates, and rules stored in the Rule Repository. These queries can elicit the business meanings from the tag read series, and according to these query results the Event Manager will coordinate the Update Controller and the Invocation Trigger for further processing, or report the elicited business events to Enterprise Application Systems.
- The Update Controller is responsible for updating the Runtime Database. The updating is initiated either by the Event Manager, or by the Update Controller to
handle the state changes caused by other changes. The updating behaviours mainly follow the rules of type (a) stored in the Rule Repository.

- The Invocation Trigger is responsible for invoking the operations of Enterprise Application Systems or Edge Systems. The Invocation Trigger is also monitoring the state changes in the Runtime Database. The invocations can be started either by the query results from the Event Manager, or by the state changes in the Runtime Database. The invocation triggering behaviours mainly follow the rules of type (b) stored in the Rule Repository.

This architecture design emphasises the event handling and business logic deployment. The data-driven middleware system monitors the run time object movements through the received events, analyses the influences of these events, and updates the status of involved objects or invokes operations of edge systems or external systems according to the defined business rules. This architecture provides a reference scheme for integrating edge systems and legacy application system with business process supports.
By changing the business rules and schema, the middleware system can support different business application contexts with the same edge infrastructure.

**Review and discussions**

*Object-oriented vs process-centric approaches*

Traditional process management approaches model business processes as a series of connected activities, which sticks to a control-flow-oriented perspective, and its execution follows an activity-completion-triggered mechanism. To adapt to the distributed computing environment where communications go through events/messages, Prof. Scheer has developed event-driven process chain in 1992. The event-driven process chain is the key component of SAP R/3’s modelling concepts for business engineering and customisation, and has also been integrated in SAP’s Netweaver System (Scheer et al., 2005). This method models a business process from the perspective of event flows, and the business process is driven by received events. Reluctantly, event-driven process chain still sticks to the control-flow-oriented scheme without many concerns on data flows or material flows. Business Process Execution Language for Web Services (WS-BPEL) (Andrews et al., 2003) provides a popular business process modelling solution particularly in the web service environment. WS-BPEL models the interactions between partners as service invocations and enables the communication via messages. The structure of a WS-BPEL process still strictly follows the pre-defined process control flow. Consequently, WS-BPEL only fits into the scenario where the activity sequence, the message sequence and the service invocation sequence are all pre-fixed.

In process-centric approaches, each process is designed to describe the procedure of fulfilling a business function. According to the embedded business logic, such a process may involve many related organisational units, staff, services as well as items, and thereby create a complex model. Normally, such models can be represented as easily readable diagrams. In comparison, object-oriented approaches do not follow the control flow between activities. Instead, they model the business contexture and behaviours with objects and related rules. Each involved object has its own lifecycle, and its behaviours are regulated by rules. A group of objects collaborate together to achieve a given goal, and such collaboration creates a business contexture. Table II lists the comparison between these two kinds of approaches.

From this comparison, we can see that object-oriented approaches provide more powerful business modelling expressivity, which can precisely describe the item-level behaviours. Compared to process models, rule-based approaches treat the handling items individually, and therefore provide a finer control over business transactions.

<table>
<thead>
<tr>
<th></th>
<th>Process-centric approaches</th>
<th>Object-oriented approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modelling focus</td>
<td>Function oriented</td>
<td>Goal oriented</td>
</tr>
<tr>
<td>Business logics are</td>
<td>Pre-defined process models</td>
<td>Business contexture and</td>
</tr>
<tr>
<td>embedded in […]</td>
<td>Activity sequences</td>
<td>behaviours</td>
</tr>
<tr>
<td>Main content</td>
<td>Relatively complex</td>
<td>Objects and declarative</td>
</tr>
<tr>
<td>Execution mechanism</td>
<td>No explicit process</td>
<td>rules</td>
</tr>
<tr>
<td>Reusability</td>
<td>inheritance support</td>
<td>Lightweight</td>
</tr>
<tr>
<td>Human readability</td>
<td>Easy to read</td>
<td>High reusability via class</td>
</tr>
<tr>
<td></td>
<td></td>
<td>inheritance</td>
</tr>
</tbody>
</table>

Table II. Comparison between process-centric approaches and object-oriented approaches.
Moreover, the rules directly define the responses of objects on receiving specific events or in given conditions, and therefore they can run more efficiently. Particularly, when handling a large volume of items (process instances) at real time, the rule-based approaches provide a better performance. These features show that object-oriented approaches fit more into the business process modelling and automation in RFID-enabled applications.

Conclusions
This paper looked into the incorporation of business process control and automation into RFID-enabled applications. A framework was proposed to model business interactions from an object-oriented perspective, which decomposed business processes into business rules which regulate the interaction behaviours between involved classes. This framework adopted a purely event-driven mechanism to capture the run time dynamics in the RFID-applied environment. An architecture design was also given with the emphasis on the interoperability with RFID edge and application systems.

Our follow-up work is to further refine the proposed object-oriented framework, and justify its extensibility and flexibility. In the future, we plan to deploy domain-specific ontology technologies for unifying the different RFID information systems.

References


Corresponding author
Xiaohui Zhao can be contacted at: x.zhao@tue.nl

To purchase reprints of this article please e-mail: reprints@emeraldinsight.com
Or visit our web site for further details: www.emeraldinsight.com/reprints