An Analysis of Physical Object Information Flow within Auto-ID Infrastructure

by

Tatsuya Inaba

Submitted to the Engineering Systems Division on May 7, 2004 in Partial Fulfillment of the Requirements for the Degree of

MASTER OF ENGINEERING IN LOGISTICS

ABSTRACT

The application of Radio Frequency Identification (RFID) has been studied for decades, and many field trials have been executed to evaluate the usability of RFID systems, the business case of RFID applications and so forth. One of the trial fields is its application to supply chain management (SCM) because the RFID technologies are thought to improve visibility of physical objects dramatically. Through this trial phase, benefits and feasibility of RFID have been confirmed, and as a result, major retailers, such as Wal-Mart, Target, and Metro, have decided to implement RFID. At the same time, these trials reveal the necessity of RFID standards. Among these newly developed RFID standards, Auto-ID standard, which was originally developed by Auto-ID Center, is a strong candidate to be a de-facto standard.

Auto-ID technologies consist of data standards and software architecture components. Data standards also consist of two components: Electronic Product Code (EPC) and Physical Markup Language (PML). On the other hand, software architecture components consist of four components: readers, Savant, EPC Information Server (EPC-IS), and Object Name Service (ONS). EPC-IS, which defines the interface of the servers that store physical object information, plays a key role in realizing business processes that the RFID technologies are expected to realize. In this thesis, we propose architecture of EPC-IS by defining the requirements for EPC-IS through generic business processes executed in Auto-ID infrastructure. The architecture we propose is not a monolithic message schema but three simple message schemas with vocabulary sets that are separately defined in dictionaries. By taking this structure, we achieve robust and scalable interface. We also evaluate our proposal by applying it to the problems found in the RFID trials and possible future business processes.

1. Introduction

The history of Radio Frequency Identification (RFID) is long, and studies and implementations of RFID go back to the mid-20th century [1]. Although the technology was categorized as RFID, the technological element behind RFID applications at that time was different from that of today. Currently RFID technology premises the use of the IC tag, which stores information about the object to which the IC tag is attached. This IC tag-based RFID technology has been studied for decades, and based on these studies field trials have been executed to evaluate wide range of applications.

One of the trial areas of RFID has been its application to supply chain management because this technology is thought to dramatically improve the visibility of physical objects and this increased visibility will bring companies in the supply chain many benefits, such as the reduction of inventory, reduction of inventory management costs, and realization of value added service.

Through trial phase, benefits and feasibility of RFID infrastructure has been confirmed, and as a result, major retailers, such as Wal-Mart, Target, and Metro, have decided to deploy RFID implementation. At the same time these trials reveal the necessity for RFID standards. RFID standards are beneficial for many reasons. Major benefits are: 1) guarantee of interoperability, 2) acceleration of implementation, 3) reduction in cost of RFID components.

Among the RFID standards, the Auto-ID standard is a strong candidate to be a de-facto standard. This standard was originally developed by Auto-ID Center, a federation of research universities that was originally founded in 1999 and is now Auto-ID Lab. Auto-ID technologies consist of data standards and software architecture components. Data standards consist of two components: Electronic Product Code (EPC) and Physical Markup Language (PML), and software architecture components consist of four components: readers, Savant, EPC Information Server (EPC-IS), and Object Name Service (ONS).

In order to fulfill the benefits of Auto-ID technologies, information about physical objects needs to be exchanged effectively within the Auto-ID infrastructure. A key component to realize this information exchange is EPC-IS, an interface for servers that store physical object information. However, specifications of EPC-IS have not been published yet. This is the current situation, and there is a gap between fundamental expectations towards Auto-ID infrastructure and the current publication of standards. In this thesis, we would like to fill the gap by proposing a solution: architecture that exchanges information with EPC-IS.

2. Background

The necessity of EPC-IS has been mentioned from the initial stage of the Auto-ID technology development because one of the primary concepts of Auto-ID technologies was to store minimal data in the IC tag and to store other related data in the servers of the physical object network. In order to realize this concept, PML was developed as a common language to describe information about physical objects, and a database to store the related information of physical object has been proposed [2].

Research of EPC-IS is difficult by nature because it is affected by industries to which EPC-IS are applied but defining the common use cases among different industries is not easy. However, several studies have been done by Cambridge University, one of the universities in the Auto-ID Lab. Approaches taken in the research are to analyze characteristics of the data stored in the server of physical object network and also to define how to use the data [3].

From the studies, data type is categorized into two: historical data and property data. The property data is also divided into product-level property data and instance-level property data. This data type definition helps the researchers to categorize use cases of each data type, and from this categorization they analyze efficient methods to retrieve data stored in the servers. One fundamental assumption they make is that the use of EPC-IS is different in each industry. Based on this assumption, their focus is to provide technology components and to assess how they are compatible with existing industry standards.

In this thesis, we take a different approach from those studies. We use the same data type category that they define and assume business processes based on the different types of data. The difference is how to achieve compatibility with the existing industry standards. Our approach is to propose information flow architecture with compatibility at the component level, which is defined in dictionary as explained in Chapter 3.

The methodology we follow in this thesis has three steps: 1) requirement analysis, 2) modeling, and 3) evaluation. In the requirement analysis step, we define generic business processes. One of the reasons why EPC-IS specifications have not been published is the lack of business processes with EPC-IS, and we define generic business processes by investigating current trials and expected business processes for RFID. After we specify these business processes, we identify the requirements for EPC-IS.

In the modeling step, we model architecture of EPC-IS. Since the requirements are not comprehensive, we do not assume that this model is used as a standard. However, this architecture of EPC-IS and the ways in which it is used will be applicable to a new standard.

In the evaluation step, we evaluate the model we propose in the previous step with problems identified in the trials and the further possible business processes. With this evaluation step, we will modify the model and improve the robustness of the architecture, if necessary.

3. Requirement Analysis

3.1. Introduction of Generic Business Process

We choose four generic business processes to better capture the requirements for EPC-IS. They are: 1) query product-level data business process, 2) query instance-level data business process, 3) query location data business process, and 4) query path data business process.

Regarding system layout, we premise the structure in Figure 1

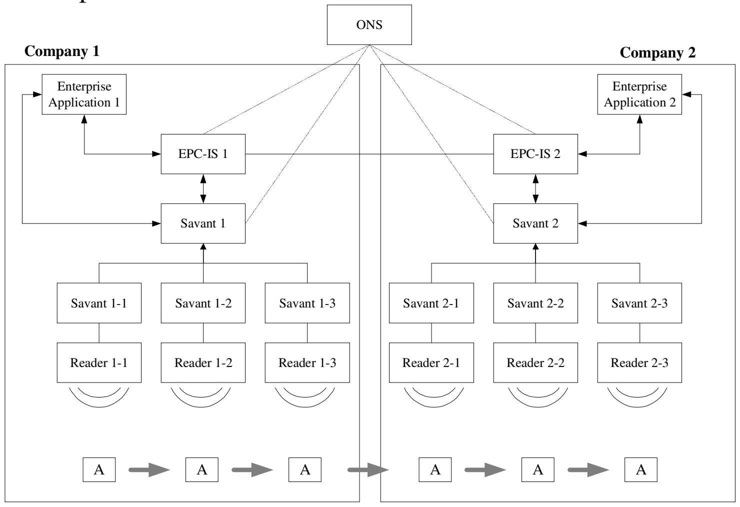

Figure 1: System layout

Query product-level data business process is the business process that is executed when company2 queries product-level data of a product to Company1's server (with EPC-IS) which stores data of the product.

Query instance-level data business process is the business process that is executed when company2 queries instance-level data of a product to Company1's server (with EPC-IS) which stores instance-level property data.

Query location data business process is the business process that is executed when Company2 queries current location data of the individual item to a server (with EPC-IS) which stores location data.

Query path data business process is the business process that is executed when Company2 queries path data that individual item takes by using EPC sent by Company1. Event data is stored in servers (with EPC-IS) of Company1, Company2 or the parties that handle the item and store the data.

For each generic business process, we develop message flow diagram and identify necessary message types, necessary data to be sent in each message, and other mechanisms that are required to EPC-IS.

3.2. Requirement Definition

From the generic business processes we define in the previous section, we distill requirements for EPC-IS.

3.2.1. Message Types

From the generic business processes, we find that three types of messages are necessary for EPC-IS: Notify message, Query/Response message, and Acknowledge message. Notify message is used when Enterprise Application, Savant, EPC-IS or other Auto-ID compliant components create or notify the data that is supposed to be stored in the server with EPC-IS. It is used for both property data and historical data, and has a capability to send data of multiple products and instances.

Query message is used when Enterprise Application, EPC-IS or other Auto-ID compliant components query data to the server with EPC-IS, and Response message is the response to the query. Key for the query may EPC, property data, date time stamp of historical data, and any other data defined in the message exchange.

Acknowledge message is used to notify the result of Notify message. In order for a sending system component to make sure the notify message is arrived at the receiving component, this Acknowledge message is necessary.

3.2.2. Scalability of the Message

Although four generic business processes are based on the RFID trials and expected business processes of RFID, they do not cover all the business processes with EPC-IS. Therefore, the model needs to have scalability to accommodate other business processes and future business processes.

3.2.3. Data type

From the generic business processes, we find that EPC-IS needs to have a capability to deal both property data and historical data. For the property data, required actions are create, update and query, but it is conceivable that this data is deleted as well. Historical data is stored when an event occurs to each individual item. Attributes which need to be stored are event type, EPC of subjected item, date time stamp, and relevant data, such as reader EPC.

4. Modeling

4.1. Modeling Direction

From the requirement analysis, one key requirement for the modeling is scalability. There are two reasons for it: one is that functions of EPC-IS has not determined and the other is that Auto-ID technologies are applied to many industries, which may have different requirements from we assume. In order to guarantee the scalability, we adopt two-layer structure for EPC-IS interface: 1) Message structure and 2) Dictionary structure. For Dictionary structure, we also adopt Action class, which describes event of historical data and action to property data, and Value class, which is the attribute of Action class.

4.1.1. Separation of Message structure from Dictionary structure

In order to guarantee the scalability, we take into account maintenance cycle of the message structure and components embedded in it. Generally the need for changing component requirement is more frequent than need for message. If property of physical object is embedded in message as XML tags, for example, if we assume to send a box shape product and model tags like <height>, <depth>, and <width> in the message, every time new shape item needs to be sent, entire message needs to be modified, even if fundamental message structure may not need to be modified. Therefore, we model message structure and components separately, and propose the way to maintain components with the dictionary.

4.1.2. Meaning of Adopting Dictionary

If components in the messages are separately defined in a dictionary, the dictionary is a vocabulary set used to describe physical object, which is equivalent to PML Extension. By separating components from message, we propose architecture of both EPC-IS and the PML Extension.

4.1.3. Dictionary Structure

We define schema of the dictionary by analyzing the data that needs to be stored in the dictionary. Table1 shows the data structure for property data. There are two types of actions for property data: One is for operating data stored in the servers, and the other is for asking data transmission to other servers. Actions for data operation are “create”, “update”, and” delete”, and the data dealt with these actions includes EPC of subjected item and relevant data, such as size, expiration date, and related business document of the item. Unit of measure and format of the value are also necessary for property data.

On the other hand, Action for data transmission is “notify”, and the data includes shipment identifier, purchase order identifier, shipped item EPC, and company profile.

Considering the consistency of the data structure, we assume using EPC for document identifier, such as shipment identifier and purchase order identifier. There is no impact of this assumption for the dictionary schema.

Table 1: Structure of Property Data

Action	Key	Relevant data
Create, Update, Delete	EPC of instance, EPC of product	Size, expiration date, business document
Notify	Shipment identifier (EPC)	Purchase order identifier, EPC of shipped items, company profile

The same structure is applied to historical data. Action for historical data is event, such as “detect”. Key for the historical data is EPC of the subjected individual item, and in addition to that, relevant data, such as date time stamp and Reader EPC needs to be collected. (Table2)

Table 2: Structure of Historical Data

Action	Key	Relevant data
Detect	EPC of instance	Date time stamp, reader EPC

From this observation, we propose the same dictionary schema structure for the property data and the historical data.

4.2. Dictionary

From what observed, we propose the same dictionary schema for both property data and historical data. We use UML Class Diagram for dictionary schema design [4].

Based on the data structure, we see that each action takes more than one relevant data but that the number of relevant data is different in each case. Therefore, we develop Value class separate from Action class and link them with identifiers.

We also understand the need to aggregate Value class when a set of values have some special business meanings. For example street, city, state, zip, and country are separate values, but usually they are for specific business entities, such as manufacturer address and retailer address. Therefore, we develop ValueSet class and link it with Action class and Value class. We also understand that, in some cases, aggregation goes multiple layers, so we add recursive structure in ValueSet class.

There are several ways to exchange unit of measure among Auto-ID components, but to reduce the data size and conversion load of each component, we propose to define unit of measure in the dictionary, which is defined as Format class and UOM class associated with Value class.

5. Message

5.1. Notify Message

Figure3 shows the message structure of Notify message. The relation between Notify class and PhysicalObject class is one to many in order for Savant to send data of multiple physical objects. Properties of Action class, dicRef and ver, come from dictionaries. By using identifier in the dictionary, the sender and the receiver can identify the action of the data. The relation between the PhysicalObject class and the Action class is one to many. With this structure, Savant can send multiple events of a physical object.

Action class has two associated classes, ElementSet class and Element class. ElementSet class and Element class correspond for ValueSet class and Value class in dictionaries respectively. This is because some instances of Actions take instances of Value directly and others take instances of ValueSets first and then instances of ValueSets take instances of Value. The reason why ElementSet class takes recursive structure comes from dictionary structure.

Element class takes Value class for its associated class. Value class is for actual historical data and property data. Since instance of each class in dictionary takes one data, the relation between Element class and Value class is one to one.

5.2. Query/Response message

Figure5 shows the message structure of Query/Response message. The structure of Response class is identical to Notify class. Since Query class describes the attributes that a query sender wants to get, Query class does not take PhysicalObject class. If a sender wants to retrieve all the data related to one EPC, it can use Value class. Each instance of Action in the dictionaries is designed to have EPC as an attribute of it.

There are several ways to develop instances of Query Message. Since our proposal uses XML, XML Query (XQuery) proposed at World Wide Web Consortium (W3C) may be applied [5], [6]. However, XQuery is more compatible to the XML instances with solid tag names than those with generic tag names like our model. Therefore, we propose different query model in this thesis.

Simple query is constructed by sending tags with dictionary reference in the Query message. For example, when a retailer wants to retrieve Reader EPC of a product, the company sends a Query message indicating the required data with tags. In the sample below, we assume that a company wants to know the EPC of readers that scan a product of which the company knows the EPC (urn:epc:1.10.100.2). When a trading company receives the query, it retrieves servers which store historical data and sends back the response. In this example, two readers (urn:epc.1.10.110.1 and urn:epc:1.10.110.2) scan the product.

Sample Query Message

```
…
<Action dicQuery=”EA001-01” name=”detect”>
  <Element dicQuery=”EV001-01” name=”EPC”>
    <Value> urn:epc:1.10.100.2</Value>
  </Element>
  <Element dicQuery=”EV003-01” name=”ReaderEPC”>
    <Value></Value>
  </Element>
</Action>
…
```

Sample Response Message

…

```
<PhysicalObject EPC=”urn:epc:1.10.100.2”>
  <Action dicQuery=”EA001-01” name=”detect”>
    <Element dicQuery=”EV001-01” name=”EPC”>
      <Value> urn:epc:1.10.100.2</Value>
    </Element>
    <Element dicQuery=”EV003-01” name=”ReaderEPC”>
        <Value>urn:epc:1.10.110.1</Value>
    </Element>
  </Action>
  <Action dicQuery=”EA001-01” name=”detect”>
    <Element dicQuery=”EV001-01” name=”EPC”>
      <Value> urn:epc:1.10.100.2</Value>
    </Element>
    <Element dicQuery=”EV003-01” name=”ReaderEPC”>
      <Value>urn:epc:1.10.110.2</Value>
    </Element>
  </Action>
</PhysicalObject>
…
```

Figure 4: Sample instance of simple query

More complex queries are also developed in the same manner. If a retailer wants know the data scanned in a specific time range, it fills the time range for the Value tag instance, and if the company wants to know the data scanned by several readers, it enumerates EPC of the readers for the Value tag instance.

Regarding sending result in Response message, we propose simple method although there are several ways to send results in hierarchical structure. We design a property, category, which has a list of values: success, fail, and partly success. If all the results of that level are the same, either success or fail is selected. When parts of the results of that level success, partly success is selected.

5.3. Acknowledge message

Figure6 shows the message structure of Acknowledge message. Since result of the Notify message may be sent either message level, PhysicalObject class level, or Action class level, Result class is associated with Acknowledge class, PhysicalObject class and Action class. Results are sent in the same way as Response message.

6. Evaluation

In the previous two chapters, we define requirements for EPC-IS and, based on the requirements, we model messages used for EPC-IS as well as PML Extension as Property Dictionary and Event Dictionary. In this chapter, we evaluate the proposed model from other perspectives, such as issues raised in RFID trials and expectation towards RFID. The points we deal with are as follows: 1) Imperfection tag detection and 2) Further business processes.

6.1. Imperfection tag detection

Although RFID obviates the need for contact and error rate is lower than bar code, the read efficiency is not 100%. From the field trial conducted by Auto-ID Lab, read rate is about 97% [7]. This rate becomes worse when items are packed in cases or cases are loaded on pallets.

In order to increase read rate, companies start to think about using EPC of a case as a representing EPC of entire items in it and EPC of a pallet as a representing EPC of cases on it and so forth. Suppose cases loaded on a pallet are sent from location A to location B, the relation between EPC of each case and EPC of a pallet can be linked at location A. If the relation is successfully linked and operators at location B can use the data, they deem all the cases are arrived when they scan EPC of the pallet. This technique is called aggregation, association or inferred reading [8].

In order to realize inferred reading, there are three things that need to be done: the event of aggregation and disaggregation need to be defined, the historical data needs to be stored, and the data needs to be effectively retrieved by Query/Response message from a receiving company.

Table3 shows the event data for aggregation and disaggregation. Attributes of the events are similar to detect event previously defined in requirement analysis. This means that these events are accommodated with Event Dictionary and that messages modeled in Chapter 5 handle historical data of these events.

Table3: Structure of aggregation/disaggregation

Action	Key	Relevant information
Aggregate	EPC of instance	Date time stamp, Reader EPC, Aggregated EPCs, Aggregation direction
Disaggregate	EPC of instance	Date time stamp, Reader EPC, disaggregated EPC

To accommodate inferred reading, we add new class, InferredReading class, as an association class of PhysicalObject of Notify message and Response message.

6.2. Further Business Processes

To evaluate the robustness of the model, we define three more business processes, which would be realized by Auto-ID technologies: 1) Handling trace business process, which is useful in such cases as international shipment which requires handling by many different companies and pharmaceutical tracing which may be legislated in some of the states [9], 2) Assembly trace business process, which is useful in such cases as tracing hazardous materials in electronic appliances, which is useful under the environment where the use of hazardous substances is strictly prohibited [10], and 3) Order track business process, which is useful in such cases as tracking the status of products which have long manufacturing process time and doing quality error management. [11].

In each case, we confirm that our model accommodate information that needs to be exchanged between servers and software components within Auto-ID infrastructure.

7. Summary and Suggestions

7.1. Summary

7.1.1. EPC Information Service Architecture

A key objective of this thesis is to propose EPC-IS architecture. We start with generic business process definition which includes the servers with EPC-IS. By defining business processes, we make sure that EPC-IS we model has the capability to support these business processes.

Then we define requirements derived from generic business processes, and propose architecture which consists of messages and dictionaries. This separation makes it possible to maintain each of the schemas independently and reduce the impact of business process change by modifying dictionary instances.

Message and dictionary schemas are evaluated with further requirements which are raised in RFID field trials and expected for RFID future applications. Through this evaluation, we modify message schema and increase dictionary instances and make them more robust.

7.1.2. Message

One of the proposals we make is message schema. In order to realize generic business processes, we develop three messages: Notify message, Query/Response message, and Acknowledge message. Notify message is used for registering and updating the physical object information, Query/Response message is used for retrieving information stored in the server with EPC-IS, and Acknowledge message is used for notifying acknowledgement and exception of Notify message.

7.1.3. Dictionary

Another proposal we make is dictionary schema and instances. We develop one dictionary schema and two sets of instances as dictionary. One dictionary is Event Dictionary which maintains events and attributes of them, and the other is Property Dictionary which maintain attributes of both product-level and instance-level property.

Since attributes defined in the dictionaries are used to describe physical object information within Auto-ID infrastructure, these dictionaries are also used as the PML Extension.

7.2. Suggestions for Further Study

In this thesis we show the architecture of EPC-IS. Since starting from generic business process, we assume this architecture is developed to actual EPC-IS standard. Therefore, one future study area is to develop EPC-IS standard with real business requirements.

Another study area is the application of EPC-IS to EDI/B2Bi standards. In this thesis, we discuss the possibility of sending data via EDI/B2Bi connection and conclude that there is little difference between using EDI/B2Bi connection and connection with EPC-IS. However, we do not discuss the impact of using EPC-IS connection to exchange other messages defined in EDI/B2B standards, such as quote, purchase order, and invoice. Since it is envisaged that companies will maintain fewer connections within their systems and between their trading partners, this study is necessary to further deploy Auto-ID infrastructure.

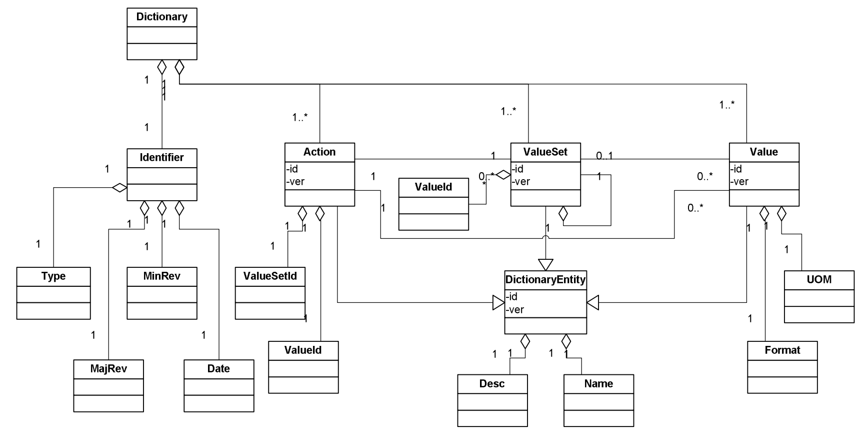

Figure 2: Class diagram for Dictionary

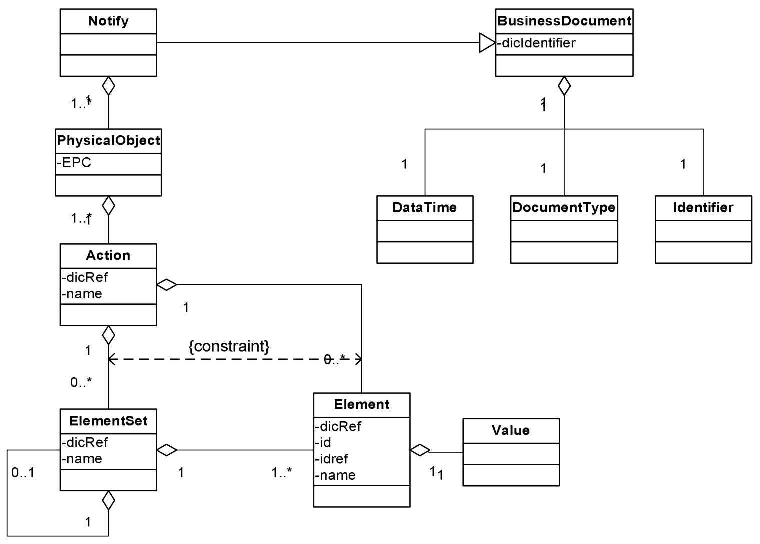

Figure 3: Class diagram of Notify message

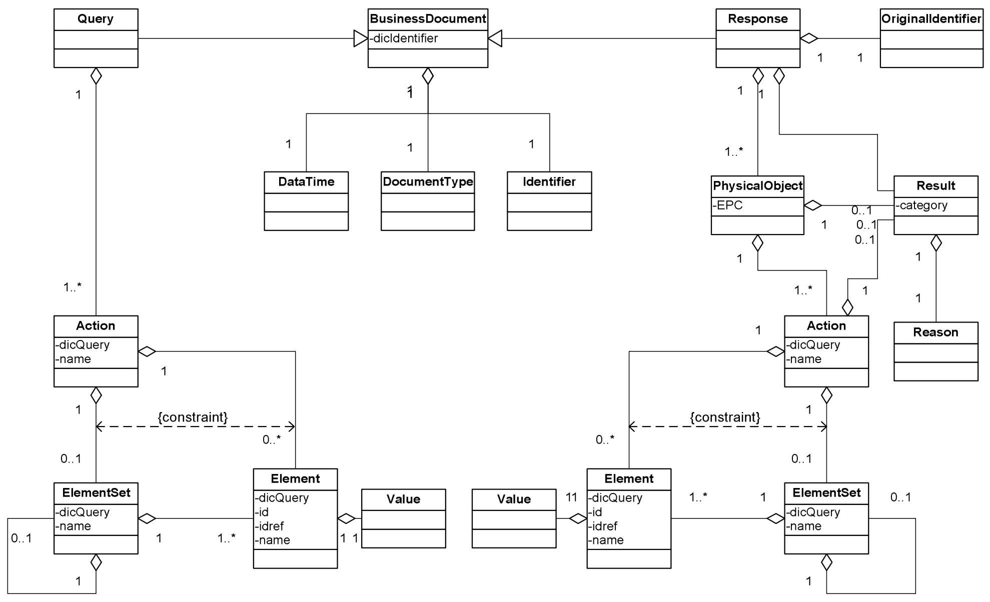

Figure 5: Class diagram of Query/Response message

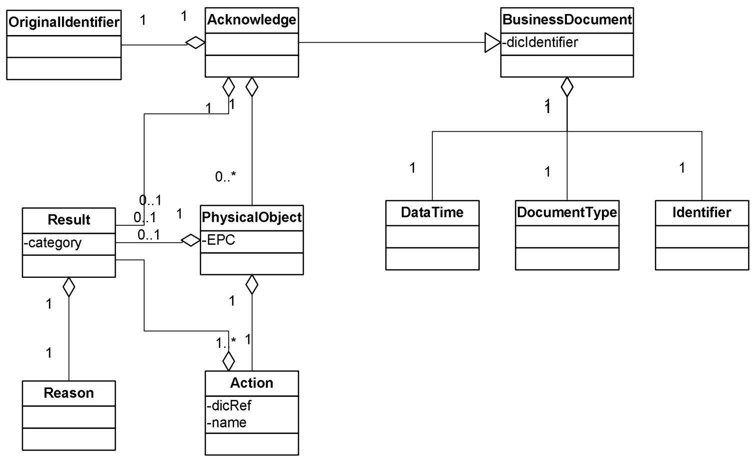

Figure 6: Class diagram of Acknowledge message

Reference

[1] The Association for Automatic Identification and Data Capture Technologies. *Shrouds of time, The history of RFID*

[2] Sanjay Sarma, David L. Brock & Kevin Ashton. *The Networked Physical World Proposals for Engineering the Next Generation of Computing, Commerce & Automatic-Identification*

[3] Mark Harrison, Humberto Moran, James Brusey, Duncan McFarlane. *PML Server Developments*

[4] RosettaNet. *PIP2A9 Technical Product Information Exchange Protocol*

[5] World Wide Web Consortium.

[6] W3C. *XQuery 1.0: An XML Query Language*

[7] Auto-ID Lab Silvio Albano, Daniel W. Engels. *Technical Report Auto-ID Center Field Trial: Phase I Summary*

[8] Auto-ID Lab Silvio Albano. *Auto-ID Field Test Lessons Learned in the Real World*

[9] U.S. Food and Drug Administration. *COMBATING COUNTERFEIT DRUGS A Report of the Food and Drug Administration*

[10] European Union. *DIRECTIVE 2002/95/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 January 2003 on the restriction of the use of certain hazardous substances in electrical and electronic equipment*

[11] Duncan McFarlane, Sanjay Sarma, Jin Lung Chirn, C Y Wong, Kevin Ashton. *Submitted to Journal of EAIA, July 2002 THE INTELLIGENT PRODUCT IN MANUFACTURING CONTROL*