Accepted to International Journal of Internet Manufacturing and Services

(Accepted Feb 11, 2009)

A Rule Language and Framework for RFID Data Capture and Processing in Manufacturing Enterprise System

Xiaoyong Su

Impinj Inc,
301 N. 34th St, Suite 300
Seattle, WA 98103

Rajit Gadh

University of California, Los Angeles
420 Westwood Plaza, UCLA,
Los Angeles, CA 90095

Abstract: In an RFID-enabled enterprise manufacturing system, diverse RFID devices and varying RFID data formats exist as a result of dynamically changing business operations, varying needs at different physical asset touch points/conditions, etc. This presents a significant challenge in device deployment, device control, RFID data capturing, RFID data integration – that are essential stacks within enterprise manufacturing information infrastructure. It is well known that rule-based methodologies are widely applied for business processing modelling and logic flow control. However, it is not common to apply rules to every stack in an information capturing infrastructure. In this paper, we propose using rules as primary instruction carrier to configure, control, capture, and integrate RFID data in an enterprise manufacturing system. An XML based rule language is defined and a rule execution engine is implemented for this purpose. Computational performance of the execution engine is evaluated against the use case and further improvement is discussed.

1 Introduction

As the manufacturing of products in commerce becomes global, the distribution of products manufactured and other operations associated with the product for an enterprise are increasingly relying on trading partners (such as suppliers and retailers). The products that undergo manufacturing and distribution in such global environment are tracked by way of an identification number (called identification data). The identification data attached to the products called business objects (such as parts, assemblies, products, shipments, assets and personnel etc.) no longer belongs to a single business entity (collectively refer to manufacturers, suppliers, retailers, ant transportation carriers etc.). It is captured, processed, and exchanged among the trading partners. Radio Frequency Identification (RFID) technology is becoming an increasingly popular identification data capture method due to its high speed capturing and non-line-of-sight capability (X. Su, et al, 2007a). However, controlling an RFID device and capturing identification data carried by RFID tags is more complex than traditional barcode scanning. This is because of the existence of numerous RFID standards (Craig K. Harmon, 2008), higher complexity of device-host interfaces, and, greater number of data/formats that can be carried by RFID tags (X. Su, et al, 2007b; Tim Coltman, et al, 2008).

Using RFID technology in a business setting becomes a challenge when there are multitudes of trading partners involved. There exist a multitude of data formats and multitude of data carriers (such as different types of RFID tags) because one company may have to support different requirement of each trading partner. The variation in tag-type requires different RFID reader types, which therefore increases the complexity of the data capture, data processing, and data exchange. In addition, trading partners are not static and a company might switch its partners frequently. For example, Wal-Mart has more than 21000 suppliers worldwide in 2003 (Charles Fishman, 2007). In 2004, 200 out of 10000 applicants became Wal-Mart's new suppliers (Gwendolyn Bounds, 2006). Suppliers in turn are constantly trying to expand their customers. For a given supplier, if its customer changes, it might have to use different RFID tags and different coding schemes. Correspondingly, the data capture and data processing must also be updated. Because of this reason, the RFID infrastructure is affected by when a supplier is added or removed.

Even within the domain of a single manufacturer, device configuration and control change overtime. For example, to reduce cost, a pharmaceutical package line usually produces more than one product. These products may vary in size, material, RFID tag type etc. A set of commands/parameters that work on one product may not work on another product.

The complexity of RFID device, dynamical business relationships, and flexible manufacturing processes introduce frequent changes within each information stack of a RFID-enabled enterprise manufacturing system. These stacks include:

(i) RFID device configuration and control. For example, company A supplies product to both Company B and Company C. Company B requires EPC Gen2 (EPCglobal, 2005) RFID tag on the product while Company C requires ISO 18000-6B RFID tag. To reduce the cost, Company A deploys RFID which support both EPC Gen2 tag and ISO 18000-6B tag. Thus, ability of switching protocols of RFID reader is a requirement. In another example, both liquid product and plastic product are produced by Company A. It is difficult to read an RFID tag that is attached to a bottle with liquid inside while it is relatively easy to read the RFID tag attached to a bottle with powders. In case of liquid presenting, multiple readers or antennas are aggregated to improve the read ability.

(ii) RFID data capturing and processing. Identification data need not always follow a single standard even though the industry is trying to establish information exchange standards. This is true especially when the user memory of the RFID tag is used to store customized data – here the format of the data varies. For example, Company A write data in its own format into RFID tag that attached a goods. Company B however, uses a different format and encrypted. Company C receives goods from both A and B. As goods arrives C, data carried on RFID tags is only useful when it is correctly read back, usually, by an information system. The information has to know what format is the data and how to decode the data. If the Company C has 100 suppliers, then in the worst case, the information system has to have 100 decoding methods. If the suppliers are changed, then the middleware has to be modified to adapt the change. Using the rules-based approach, this problem is alleviated.

(iii) RFID data exchange and integration. Nowadays, data exchange is very common in globalized commerce environment and exchange crosses industry boundary. RFID information is not an exception. It is even more complex since RFID data exists in the whole product live cycle from manufacturing line to end consumer.

It is well known that a system relying on the static parameters cannot adapt to the complex and rapidly changing environments. To adapt to the complexities and changes, rule-based data processing and rule based device configuration and control is applied since the rule based system allows changing of behaviour at runtime without interfering with the operation. We identify three processes that require rule based processing in an RFID-enabled enterprise manufacturing system. They are: rule based device configuration and control, rule based RFID data capturing and processing, and, rule based data exchange and integration. To provide a flexible yet scalable RFID infrastructure that allows changing of device configuration, device control, data capturing and integration on the fly, a semantic rule language, based on structured Extensible Markup Language (XML) (Gerd Wagner, 2002) is defined and a rule execution engine is implemented as WinRFID rule framework - a essential component of WinRFID RFID integration platform (B.S. Prabhu, et al, 2005; X. Su, et al, 2007c). The rest of this paper is organized as follows: literature review is performed in section 2, framework of a rule language as well its syntax are discussed in section 3, section 4 describes the rule engine architecture and section 5 evaluates performance of the execution engine applied to the use case of RFID data capturing. The research is summarized in section 6.

2 Literature review

A rule is a set of declarative logical statements used by an execution system to determine the system status or its next behaviour. Rules have been widely used in fields such as database, logic programming, artificial intelligence, information processing, and moreover, the enterprise operations. Using rule decouples core function modules of a system from various input conditions. A desired behaviour of a system can be achieved by medication of the rules without touching the core function modules of a system. Thus, a rules based system is agile and can adapt to dynamically changing environment.

Due to the nature of the RFID technology presented in a manufacturing system and/or its existence in the product lifetime as discussed in our introduction, it is obvious that rule based processing is the first choice in an RFID-enabled manufacturing system and beyond. Although there are still not many research works on rule based processing in the field of RFID-enabled manufacturing system, some researchers use rules for RFID data transformation and enterprise information integration. Wang and Liu (2005) discuss a rule based RFID data transformation approach that converts raw RFID data into high level business aware and semantically presented data. Zang and Fan (2007) describe a rule based event processing approach for enterprise information system. RFID data along with spatial and temporal information are evaluated against a set registered rules or constraints. Outputs of this evaluation process are events that can be used for driving activities in enterprise information system. Although they define a complete architecture including a rule language for declaring rules, we observed that the rule language that Zang and Fan defined is not flexible enough to cover complexity of a RFID enabled manufacturing system.

There are various rules existing in different application domains. The most studied and used rules are: derivation rules, integrity constraints, and event-condition-action rule, (Floriman Rosenberg, et al, 2005; Gerd Wagner, 2002; Steffen Schott, et all, 2003).

A derivation rule is also called deduction rule. It is a statement that is derived from other statements by an inference or calculus. A system using derivation rule does not store conclusion but rather a set of conditions. For example, a simple derivation rule could be: an ID carried by RFID tag is a SGTIN-96 number if it has first 4 digits as “3004” and has total 96 bits in length.

The Integration Constraint is an assertion that all evolving states and the relationships of the states must satisfy (J. Gu, et al, 1997; Gerd Wagner, 2002). The constraints can be classified as two types: state constraints and process constraints. State constraints are normally persistent in a system while the process constraints are transitional. For example, all CNC machine with tag ID starts with “F001” must be operated under room temperature less than 80 degrees Fahrenheit.

An event-condition-action rule is also called reaction rule. As indicated in its name, it contains three essential elements in a rule: Event, also called Facts, is usually a complex data type input to be evaluated against the conditions; Condition is a set of criteria or expressions that used to evaluate the Event; Action is a function call that will be invoked as long as the Event satisfies the Condition. In ECA, logical operations can be performed by calculating the input against pre-defined conditions. The comparison then yields a TRUE or FALSE result. Based on the result, a pre-defined action is triggered. For example, in an RFID-enabled manufacturing system, every goods has a unique RFID tag attached. As the goods move through each processing point. The RFID tag is read by RFID reader. At this point, the RFID data is an input. Once the RFID data satisfies a set of condition, an action is taken on the goods.

The ECA rule supports relatively simple atom but also can yield complex rules by combining multiple rules in a systematic manner. We observed that features of ECA rule fit in three information stacks that we identified for a RFID-enabled manufacturing system. Thus we utilize the reaction rule for enabling rule-based RFID data capturing and processing in a RFID-enabled manufacturing system.

3 Design the Rules

3.1 Boolean logic and AND/OR tree

Since the TRUE and FALSE are used as the reference rule decision in a ECA engine, we represent the conditions in rules by using Boolean logic. Boolean logic is used to determine a system status through a set of variables. Or conversely, a Boolean logic statement may be mapped to: given a Boolean formula, find out the values of the variables so that the formula is satisfied. This is the most famous and well-studied Boolean satisfiability problem (SAT) (Gerd Wagner, 2002). The variables have only two values: TRUE or FALSE. The Boolean logic uses three operators: AND, OR, and NOT. The AND operator is also referred as a conjunction and the OR operator is referred as a disjunction. Only Boolean logic is identified within WinRFID because the operations of the rules are comparisons which eventually yield

TRUE or FALSE. Thus, Boolean logic is used as the primary way to represent rules and execute the rules in WinRFID.

There are two kinds of formulae defined in Boolean logic: Conjunctive Normal Form (CNF) and Non Conjunctive Normal Form (Non-CNF). The CNF is a conjunction of disjunctions. A disjunction is also called a clause that has a group literals with “OR” relation. The literal is a variable or negated variable. All the other formulas that are not subjected to CNF are Non-CNF.

Let’s take a look on the following examples.

$$f1\ (a, b, c) = a \text{ AND } b \text{ AND } c \qquad (1)$$
$$f2\ (a, b, c) = a \text{ OR } b \text{ OR } \neg c \qquad (2)$$
$$f3\ (a, b, c) = (a \text{ OR } b \text{ OR } c) \text{ AND } (\neg a \text{ OR } b \text{ OR } c) \qquad (3)$$
$$f4\ (a, b, c) = (a \text{ OR } b \text{ AND } c) \text{ OR } (\neg a \text{ AND } b \text{ AND } c) \qquad (4)$$

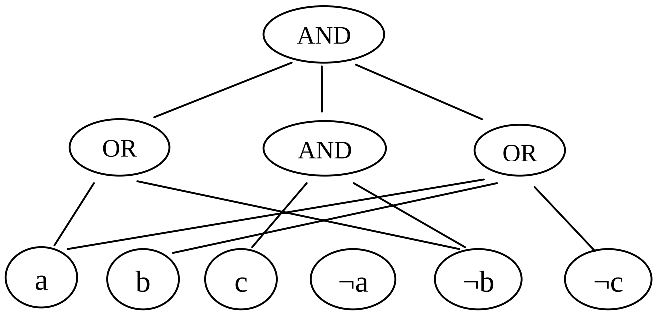
Where a, b, and c are literals and ¬ represents NOT.

As we can see, the formula f1 (a, b, c), f2 (a, b, c), and f3 (a, b, c) are CNF. f4 (a, b, c) is Non-CNF. The concept of CNF is produced to solve the SAT problem which is typically to find the value of variable to satisfy the formula. In WinRFID, we want to obtain a system status from a set of conditions, so we are not constrained to the CNF formula.

The Boolean logic can be represented in an AND/OR tree (B.H. Anold and B. Henry, 1962) as illustrated in Figure 1. Each node in the AND/OR are label either “AND” or “OR”. Each node has the value “TRUE” or “FALSE”. Given a tree, the evaluation of the tree is done recursively in the following ways:

• The value of an OR node is true when at least one of its children is TRUE, otherwise is FALSE
• The value of an AND node is true if all its children are TRUE, otherwise is FALSE

Figure 1 AND/OR tree



Represent this tree into a Boolean logic formula, we have:

$$f\ (a, b, c) = (a \text{ OR } \neg b) \text{ AND } (\neg b \text{ AND } c) \text{ AND } (a \text{ OR } b \text{ OR } \neg c) \qquad (5)$$

Following the Boolean logic formula, we can construct the conditions for the reaction rule.

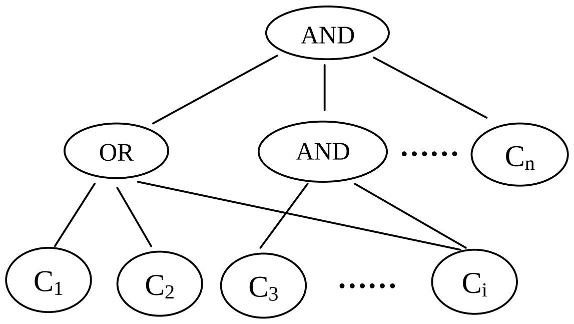
3.2 Abstraction and Analysis

Structured data is the commonly used data format in general computations. A structured data element usually contains several fields (also called members). Each field is either a simple data type (such as an integer, float, double, char, string, and datetime) or complex data type. The complex data type is normally a user-defined type. It contains a combination of the simple data types or several other complex data types.

The data types defined in WinRFID are structured data that consists of simple data type. In this research, we do not intend to define a rule language that can directly deal with the complex data type because the simple data structure would yield higher efficiency and performance. However, the rule language defined supports the complex data type by using composite rules.

As mentioned earlier, the Boolean logic is a calculus of a set of TRUE or FALSE value. The TRUE or FALSE represents a satisfaction of a condition. If we represent the condition as a node, a tree of conditions similar to the Boolean logic AND/OR tree can be established (As shown by Figure 2). In this tree, the condition node is the least unit or leaf. Each OR node or AND node has a set of child nodes (Either another AND/OR node or condition node)

Figure 2 Condition Tree (Where Ci is Condition)



The condition contains information that states how to compare an input value with criteria to yield a TRUE or FALSE value. The comparison is usually a logic operation. It always execute as the following:

```
If a value satisfies a criterion
    return TRUE
else
    return FALSE
```

The following logic comparison operations are identified in WinRFID. For the simple data type such as integer, float, double, char, and datetime, the logic operations are: “Less Than”, “Less or Equal To”, “Greater Than”, “Greater or Equal To”, “Equal To”, “Not Equal To”; For the string type, the logic operations are: “Equal To”, “Contains”, “Start With”, “End With”, and “Not Equal To”.

From the analysis, we can conclude that in a condition, there are four attributes: Logic Operator, Data Type (the structured data type defined in the program), Data Field (member or field of the structured data), and the criteria (or value) that has been evaluated against. Thus, a condition has the following relation in the form of tuple:

$$C = (LO, DT, DF, V) \qquad (6)$$

Where

C:	Condition
LO:	Logic Operator
DT:	Data Type
DF:	Data Field
V:	Value

The AND/OR node is much simpler than a condition node. Because it contains a set of condition nodes or AND/OR nodes, let’s use Conditions node to represent the AND/OR node. In a Conditions node, besides the child nodes, the Boolean logic that connects the child nodes together needs to be a given and it should be the only explicit attribute to be a given. The conditions node can be represented as:

$$Cs|BL=AND = C1\&C2\&\ldots\&Cn \qquad (7)$$

Or

$$Cs|BL=OR = C1|C2|\ldots|Cn \qquad (8)$$

Where
Cs: Conditions
Ci : Condition or Conditions, i
BL: Boolean Logic

3.3 Define the rule language

The rule language is a set of definitions that specify the grammars or regular expressions to represent the logics or process work-flows. Since XML is becoming the de-facto standard for designing a rule language, we also use XML to organize the grammars and expressions. XML-based rule languages exist in various applications. For example, the well-known XSLT and XQuery (S. Boag, et al) are used for XML document transformation and query formulation; Rule Markup Language (RuleML) (Harold Boley, et al, 2001; Gerd Wagner, 2002) is used in Semantic Web where the web is used for automation, integration and reuse. Business Process Execution Language (BPEL) (Harold Boley, et al, 2001) is a XML-based language to facilitate Web Services orchestration. Strictly speaking, the BPEL is not a pure rule language. It is a sophisticated tool that can conduct inter-business, business-to-business operation based on Web Services. It has been widely used for the enterprise Web-Services-Oriented-Architecture solution and adopted as an ad-hoc standard by several organizations. Besides these well-known XML-based rule languages, there are many other XML-based rule languages which were designed for particular applications.

Because of their application dependency and unique characteristics, none of existing rule languages can be directly applied to fit the requirement of WinRFID. Thus, based on the data types and the features of XML, a specially designed rule language to process the structured data and the combination of the Boolean logic operations is implemented in WinRFID.

3.3.1 Logic comparison operator abbreviations

To simplify the syntax and process, a concise representation mechanism to express the logic comparison operation identified in Section 7.3 is defined. In the context of XML, there are no operators defined for "Contains", "Start With", and "End With". According to XML 1.0 rules, the operators must be quoted by using "&". For example < and <= operators must be present as &lt and &le. Because of the above reasons, several XML compliant operators are defined (As shown in Table 1)

Table 1 Logic comparison operator definition

Logic Comparison	Abbreviation	Operator	XML compliant operator
Less Than	LT	<	&lt
Less or Equal To	LE	<=	&le
Greater Than	GT	>	&gt
Greater or Equal To	GE	>=	&ge

Equal To	ET	=	&et
Not Equal To	NE	!=	&ne
Contains	CT		&ct
Start With	SW		&sw
Start At	SA		&sa
End With	EW		&ew

3.3.2 Build Condition node

From the equation (6), a tuple of condition contains four elements: LC (Logical Comparison), DT (Data Type), DF (Data Field), V (Value). Let’s transform the Condition node into a XML node. We can have the following forms:

```
<Condition DT=”Data Type” DF=”Data Field” LC=”Operator” V=”Value”/>
Or
<Condition Expression=”DT.DF Operator Value”/>
```

The first form uses the attribute property of a XML node, the process could be standard and each attribute could be identified individually. However, it is not straightforward and a little difficult to understand. The second form is more concise and straightforward. Because of this reason, we chose the second one as our predominant rule representation format.

For example, if we want to capture all the SGTIN RFID tag belonging to company ABC which has been assigned Company Prefix (EPCglobal, 2006) 06520222, the condition can be written as:

```
<Condition Expression =”SGTIN.CompanyPrefix &et 06520222”/>
```

Besides supporting data types defined in WinRFID, a pattern matching process is introduced to support raw data types such as binary string, digits string, and general string. For example, if we want to capture an id with string “12868” which starts at the fourth digit, the condition can be written as:

```
<Condition Expression =”RAW &sa 4 &et 12868”>
```

3.3.3 Build Condition node

The Conditions node is a node which has a set of child node (either Condition node or Conditions node). The Boolean logic operation of the child node is defined as an attribute of the node. As indicated in Equations (7) and (8), the Boolean operation of the child nodes is either AND or OR. Let’s transform the Conditions node into XML node, we have:

```
<Conditions BL=”Boolean Logic”>
</Conditions>
```

Consider child nodes, the overall Conditions node has the follow syntax:

```
<Conditions BL=”Boolean Logic”>
    <Condition />
……
    <Condition />
    <Conditions />
    ……
    <Conditions />
```

```
</Conditions>
```

For example, if we only want to process the SGTIN RFID tag which was tagged on Company ABC's (Company Prefix is 06520222) product D (Serial Number is 12828828). We have:

```
<Conditions BL="AND">
    <Condition Expression = "SGTIN.CompanyPrefix &et 06520222"/>
    <Condition Expression = "SGTIN.SerialNumber &et 12828828" />
</Conditions>
```

If we want to process SGTIN RFID tags that were tagged on all Company ABC's (Company Prefix is 06520222) A category products (Item Reference Number is 001883) and all Company BCD's (Company Prefix is 06520211) products except the B category products (Item Reference Number is 002201). We can have the following conditions syntax:

```
<Conditions BL="OR">
    <Conditions BL="AND">
            <Condition Expression = "SGTIN.CompanyPrefix &et 06520222"/>
            <Condition Expression = "SGTIN.ReferenceNumber &et 001883" />
    </Conditions>
    <Conditions BL="AND">
            <Condition Expression = "SGTIN.CompanyPrefix &et 06520211"/>
            <Condition Expression = "SGTIN.ReferenceNumber &ne 002201" />
    </Conditions>
</Conditions>
```

3.3.4 Build action node

As mentioned earlier, the Action is a function call which is invoked once the conditions are satisfied. It can be on the local or remote machine. There are various ways to make a function call. Generally, the client needs to embrace the class name (When functions are encapsulated into a class), function name, and one or more parameters into the function calling request. If the client calls the function in the remote components or web service based functions, it is necessary to provide a URL which points to the description of how to invoke the remote calling. Let's write the function call as the following:

$$A = C.F(P1, P2, \ldots, Pn) \qquad (9)$$

Where

A: Action
C: Class
F: Function
P1~ Pn: Parameters

Translate the relation (9) in XML based syntax, we have:

```
<Action>
    <Function Name="Function Name" Class="Class Name" Desc="URL of description">
        <Parameters>
            <P1></P1>
            <P2></P2>
            ……
            <Pn></Pn>
        </Parameters>
    </Function>
</Action>
```

The tag <Pn> is the name of the parameters. It could be a simple data type or complex data type. When the complex type used, the < Pn> represents a data type node. The Class, Function and the Description URL are represented as attributes.

For example, there is a function called "SendMail" in a class "Messaging", it has parameters including receiver email, subject, and message body. All the parameters are a string. Following the Action node syntax definition, we have:

```
<Action>
    <Function Name="SendMail" Class="Messaging" URL="">
        <Parameters>
            <receiver_email>who@abc.com<receiver_email>
            <subject>This is a test</subject>
            <body>message body </body>
        </Parameters>
    </Function>
</Action>
```

The parameters tags' name should exactly match the parameters' name. The function name and class name are also exactly the same as the one defined in the program. The reason for keeping the name consistent is discussed in the next section.

By combining the Condition node with the Action node, a complete ECA rule language is defined. For example, if we put the Condition node example and Action node example together, we have:

```
 <Conditions BL="AND">
    <Condition Expression = "SGTIN.CompanyPrefix &et 06520222"/>
    <Condition Expression = "SGTIN.SerialNumber &et 12828828" />
</Conditions>
<Action>
    <Function Name="SendMail" Class="Messaging" URL="">
        <Parameters>
            <receiver_email>john@abc.com<receiver_email>
            <subject>This is a test</subject>
            <body>message body, let's write down something here</body>
        </Parameters>
    </Function>
</Action>
```

This presents the following rule: if a SGTIN tag which is being tagged on company 06520222's product 12828828 is sensed, send an email notification to john@abc.com.

4 Execution engine

The rule language can not produce any outputs. An execution engine that is used for comparing/calculating the event against the conditions and triggering the action is required.

4.1 Execution engine model

Figure 3 shows the black box view of the execution engine's architecture. In WinRFID framework, the Events/Facts are identification data captured by a device and the event is generated by the system. The Rules contain filtration conditions or subscription conditions and actions. The actions define how to trigger an action, once the Events/Facts satisfy the conditions. The Events/Facts and Rules are the only two inputs of the execution system. The execution engine evaluates the Events/Facts against the Rules. Based on the definition of Action in the Rules, the execution engine will invoke a function call or return a status that can be used for further decision in the system.

Figure 3 Execution

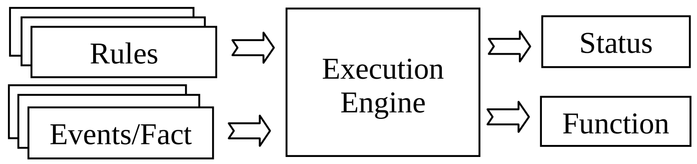
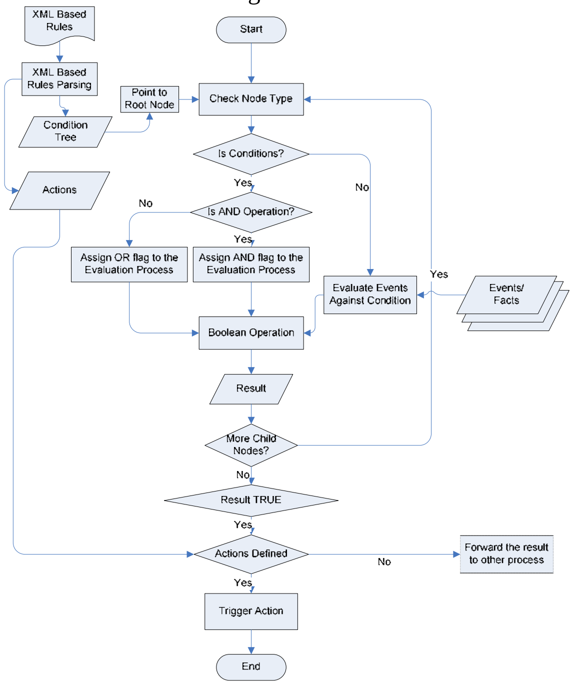


Figure 4 Logic flowchart of the Execution Engine



As mentioned earlier, the evaluation process of the Events/Facts is based on the Boolean logic in WinRFID framework. The Events/Facts are evaluated against each condition. The result which is in the value of TRUE or FALSE is then used for the Boolean operation transverse through the whole AND/OR tree.

The Logic flowchart of the Execution Engine is show in the Figure 4. At the beginning of the process, the rules in form of XML format are loaded into the execution engine and are parsed into two parts: Conditions (The condition tree) and Actions. The rule engine evaluates the identification data by invoking the Execute method when the there is identification data is captured. The evaluation processes include the evaluation against the condition (comparing an attribute of the Events/Facts with the expected value), the Boolean operation of the conditions, and the invoking of the action if it is necessary.

5 Quantitive measurement

Besides the functional evaluation that pass different data to rule engine to test the correctness of the rule engine. The speed of processing rule is very critical especially in a high speed data processing environment. The WinRFID rule engine is measured on a Dell precision workstation quipped with a 2.0GHz Xeon processor and 1.0GB RAM. In the figures, the level means the depth of the condition. The level of conditions defined as the following:

(1) One level conditions

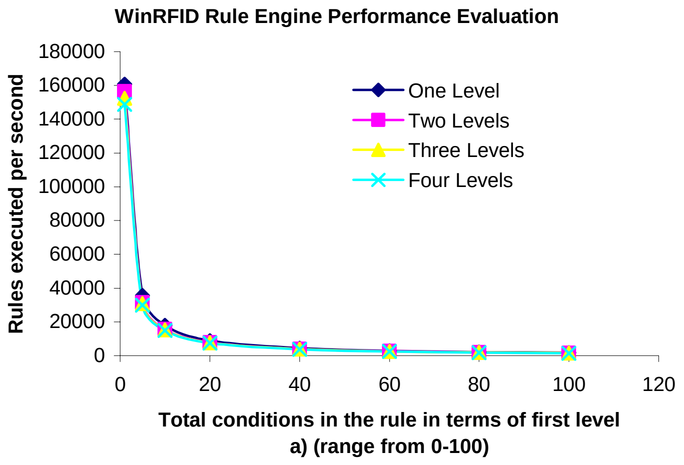
```
<Conditions BL=”AND”>
    <Condition Expression = “SGTIN.CompanyPrefix &et 06520222”/>
    <Condition Expression = “SGTIN.SerialNumber &et 12828828” />
</Conditions>
```

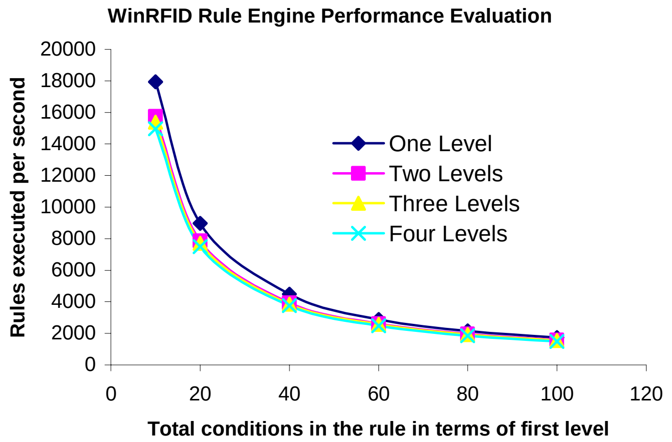
(2) Two level conditions

```
<Conditions BL=”AND”>
    <Conditions BL=”OR”>
            <Condition Expression = “SGTIN.CompanyPrefix &et 06520222”/>
            <Condition Expression = “SGTIN.SerialNumber &et 12828828” />
    </Conditions>
</Conditions>
```

We measured the number of rules can be executed per second. Level of conditions and number of conditions in the rule are parameters of the measurement. Four types of rules including one-level-conditions-rule, two-level-conditions-rule, three-level-conditions-rule and four-level-conditions-rule are measured. For each type of rules, 1, 5, 10, 20, 40, 60, 80, 100 conditions are measured. As shown in Figure 5. We observed that the speed of rule processing significantly decreased as the number of conditions increases. The level of condtions affects the process speed. But the effect is not as significant as the effect of the number of conditions.

Figure 5 WinRFID Rule Engine Performance Evaluation





b) (range from 10-100)

6 Summary

RFID-enabled enterprise manufacturing systems are subject to dynamical changes in business operation, manufacturing process and device complexity, rule-based device configuration, control and data integration.. We propose within our WinRFID middleware, a rule framework which contains a semantic rule language and a rule engine based on reflection technology. The rule language including syntax and semantics used to create condition node based on XML is defined.

True and False are naturally used as conditions for most control decision in a RFID-enabled enterprise manufacturing system. The approach to using Boolean logic is studied in this research and Boolean logic is utilized to build a condition tree that is used to represent nestings. These nested conditions are presented in the form of XML language. Because of both extensibility and flexibility of the Boolean logic tree and XML language, the rule framework can use simple conditions to form a fairly complex rule.

The proposed research is verified against a scenario of setting filter conditions in an RFID data capture process. In the test, comparison of the input and pre-defined conditions are logic operations to yield a TRUE or FALSE result and then trigger an action. The computational speed of rule processing is evaluated to determine whether the rule engine is capable of handling complex conditions and actions in domain of RFID-enabled manufacturing system. The result of the measurements shows acceptable performance of the rule engine. However, the performance significantly drops as the number of conditions in a rule increases. This is because the rules are intercepted by the engine at runtime. The greater the number of condition nodes intercepted, the more is the time consumed. This is generally true for a rule engine. The result indicates that for a practically usable rule-based RFID manufacturing system, a dedicatedly designed rule is required to fulfill real-time interception and operation.

References and Notes

Xiaoyong Su, Chi-Cheng Chu, B. S. Prabhu, Rajit Gadh, 'On the creation of Automatic Identification and Data Capture infrastructure via RFID and other technologies', *The Internet of Things: from RFID to the Next-Generation Pervasive Networked Systems,* Lu Yan, Yan Zhang, Laurence T. Yang, Huansheng Ning (eds.), Auerbach Publications, Taylor & Francis Group, pp 33-52, 2007

Xiaoyong Su, Chi-Cheng Chu, B. S. Prabhu, Rajit Gadh, ‘On the Utilization and Integration of RFID data into Enterprise Information Systems via WinRFID’, *Computers in Engineering Conference*, Las Vegas, NV. , Sep 4-7, 2007

B.S. Prabhu, Xiaoyong Su, Charlie Qiu, Harish Ramamurthy, Peter Chu, Rajit Gadh, ‘WinRFID – Middleware for Distributed RFID Infrastructure’, *International Workshop on Radio Frequency Identification (RFID) and Wireless Sensors*, Indian Institute of Technology, Kanpur, India, November 11-13, 2005.

C. Zang , Y. Fan, ‘Complex event processing in enterprise information systems based on RFID’, *Enterprise Information Systems*, Vol.1 n.1, February 2007, pp.3-23

Xiaoyong Su, Chi-Cheng Chu, B.S. Prabhu, and Rajit Gadh, ‘On the Identification Device Management and Data Capture via WinRFID Edge-Server’, *IEEE Systems Journal*, 1(2), Dec 2007, pp95-104.

Fusheng Wang , Peiya Liu, ‘Temporal management of RFID data’, *Proceedings of the 31st international conference on Very large data bases*, Trondheim, Norway, August 30-September 02, 2005

Tim Coltman, Rajit Gadh, Katina Michael, ‘RFID and Supply Chain Management: Introduction to the Special Issue’, *Journal of Theoretical and Applied Electronic Commerce Research*, 3(1), 2008, pp 3-4

Florian Rosenberg, Schahram Dustdar, ‘Business Rules Integration in BPEL – A Service-Oriented Apporach’, *Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*, 2005, pp476 – 479

Gerd Wagner, ‘How to Design a General Rule Markup Language?’, *XML Technologien für das Semantic Web - XSW 2002*, Proceedings zum Workshop, June 24-25, 2002, p.19-37

Steffen Schott , Markus L. Noga, ‘Lazy XSL transformations’, *Proceedings of the 2003 ACM symposium on Document engineering*, Grenoble, France, November 20-22, 2003

J. Gu, P. W. Purdom, J. Franco, and B. W. Wah. ‘Algorithms for the Satisfiability (SAT) Problem: A Survey’. *DIMACS Series in Discrete Mathematics and Computer Science*, 35:19-151, 1997.

B.H. Arnold, B. Henry, ‘Logic and Boolean algebra’, Prentice-Hall, 1962

S. Boag, D. Chamberlin, J. Clark, M. Fernandez, D. Florescu, J. Robie, J. Siméon, M. Stefanescu, ‘Xquery 1.0: An XML Query Language’, http://www.w3.org/TR/xquery/

Harold Boley, Said Tabet and Gerd Wagner, ‘Design Rational of RuleML: A Markup Language for Semantic Web Rules’, *Semantic Web Working Symposium (SWWS’01),* Stanford University, July/August 2001, pp381–401

Gerd Wagner, ‘How to Design a General Rule Markup Language’, *Invited paper at workshop XML Technologies for the Semantic Web (XSW2002)*, Berlin, Germany, June 2002.

Bob May, Praveen Savur, ‘Business Process Execution Language, Part 1: An Introduction’ http://developers.sun.com/prodtech/javatools/jsenterprise/tpr/reference/techart/bpel.html

Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler, Francois Yergeau, ‘Extensible Markup Language (XML) 1.0 (Third Edition)’, *W3C Consortium*, http://www.w3.org/TR/2004/REC-xml-20040204/

EPCGlobal, ‘Tag Data Structure Standards Version 1.3’, *EPCGlobal*, http://www.epcglobalinc.org/standards_technology/Ratified%20Spec%20March%208%202006.pdf

Charles Fishman, ‘Wal-Mart You Don’t Know’. *Fact Company Magazine*, Dec 19, 2007, http://www.fastcompany.com/online/77/walmart.html.

Gwendolyn Bounds, ‘The Long Road To Wal-Mart Shelves’, *The Wall Street Journal*, June 5, 2006, http://wsjclassroom.com/monday/mx_06june05.pdf

EPCglobal, ‘EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communication at 860 MHz – 960 MHz’, Version 1.0.9, January, 2005. http://www.epcglobalinc.org/standards/Class_1_Generation_2_UHF_Air_Interface_Protocol_Standard_Version_1.0.9.pdf

Craig K. Harmon, ‘Supply Chain RFID & Bar Code International Standards Efforts’, *AIDC standards update*, January 7, 2008 http://www.autoid.org/presentations/2008/IntlStdsUpdate_20080107_ckh.ppt