Inf Syst Front
DOI 10.1007/s10796-008-9109-0

Architecture, implementation and application of complex event processing in enterprise information systems based on RFID

Chuanzhen Zang · Yushun Fan · Renjing Liu

Abstract Enterprises have to be increasingly agile and responsive to address the challenges posed by the fast moving market. With the software architecture evolving into service-oriented architecture, and the adoption of Radio Frequency Identification (RFID), event processing can fit well in enterprise information systems in terms of facilitation of event aggregation into high level actionable information, and event response to improve the responsiveness. To make it more applicable, the architecture of event processing in enterprise information systems is proposed; event meta model is put forth, and the rules are defined. To improve the detect efficiency, classification and partition of event instance is utilized. We have implemented the event processing mechanism in enterprise information systems based on RFID, including the data structures, optimization strategies and algorithm that is considered as one of the contributions. The performance evaluations show that the method is effective in terms of scalability and the capability of event processing. Complex event processing can improve operation efficiency and discover more actionable information, which is justified by the application.

Keywords Event driven architecture · Complex event processing · Event meta model · RFID · Workflow

C. Zang (✉) · Y. Fan
Department of Automation, National CIMS Engineering and Research Center, Tsinghua University, Beijing 100084, China
e-mail: zangchuanzhen@tsinghua.org.cn

R. Liu
School of Management, Xian Jiaotong University, Xian 710049, China

1 Introduction

The fast moving market makes it inevitable for enterprises to be more agile and responsive. To address these challenges, existing information systems such as Enterprise Resource Planning (ERP), Supply Chain Management (SCM), Customer Relationship Management (CRM), and Manufacturing Execution System (MES) (Hou et al. 2007; Xu et al. 2006; Xu 2007; Li et al. 2008; Warfield 2007; Zhang and Bhattacharyya 2007; Zhang et al. 2007) have improved the operation efficiency of enterprises to some extent. In addition, solutions such as on demand enterprise and real time enterprise aim at improvement of agility and responsiveness.

Only when they are supported by effective architectural styles are these information systems and solutions possible (Fan et al. 2005). Service-oriented architecture (SOA) and event-driven architecture (EDA) tend to be dominant in the architectural area. They both aim at maximizing the re-use of application-neutral components that increase information technology (IT) adaptability and efficiency, and serve as key contributors to the evolving process of business integration (Maréchaux 2006).

Although these breakthroughs in information systems, solutions and software architectures work to some extent, there are still some disadvantages. Generally speaking, there are three information gaps in the existing information systems.

The first information gap is the one between the information systems and the physical situation. It is difficult to automatically retrieve real-time, accurate, and detailed data such as what the accurate inventory volume is or where the parts are.

The second information gap is the one between the enormous primitive data and the actionable information for

executives. Even though the data in the physical world can be retrieved, there is no effective mechanism to discover the business level, actionable information behind all these low level data. For example, a credit card is used in Beijing at 9:21 AM, and after a few minutes there is a transaction of the same card in New York. An effective method that can tell there is a card fraud is substantially needed.

The third information gap is the one between the business level, actionable information and the enterprise response. Currently, there is no effective mechanism or architecture to automatically support response to the actionable information. For example, when a credit card fraud is discovered, effective measures should be automatically taken to prevent this fraud.

Radio Frequency Identification (RFID) (Ohkubo et al. 2005; Nath et al. 2006) is the first step to integrate virtual world and physical world to bridge the first information gap. RFID can play an essential role in object tracking and SCM. In addition, it promises to improve the operational performance in terms of inventory management, manufacturing process automation, even the personnel management. The data of the physical world in enterprises can be read in bulky manner by elaborately deployed readers without line of sight, and then be processed by edge server (RFID middleware) and enterprise information systems, when the items, equipments and people are attached by RFID tags. Such RFID enabled enterprise environment has been piloted world wide, and it will be rolled out in the near future.

But the volume of data generated by RFID systems can be enormous, which is beyond the processing capabilities of existing information systems (Bornhovd et al. 2005). New mechanisms are needed to process the enormous volume data, to discover the business information behind these data as well as to respond to the information automatically, namely, to bridge the second and the third information gap. Complex Event Processing (CEP) (Luckham 2002) is a good approach to address problems posed by the three information gaps. The key ideas of CEP are as follows: firstly, primitive event extraction from large volume data; secondly, event correlation or event aggregation to create business event with event operators according to specific rules; thirdly, event processing of primitive or composite event to obtain their time, causal, hierarchical and other semantic relationships; and fourthly, response to the actionable business information.

The paper discusses where event processing can fit in the enterprise information systems with the adoption of RFID. The meta model, event context and event pattern rules are formulated in detail. System implementation and optimization strategies are also proposed elaborately. The method and implementation are justified by performance evaluation as well as application in refrigerator manufacturing plant.

The remainder of this paper is organized as follows: in Section 2, related works are discussed. In Section 3, the architecture of event processing in enterprise information systems is put forth and discussed in detail. CEP language is described in Section 4, and the system implementation and optimization strategies are proposed in Section 5 followed by performance evaluation in Section 6. Section 7 describes how CEP works in manufacturing enterprise. Conclusion and future works are presented in Section 8.

2 Related works

The adoption of RFID poses new challenges for data processing and management. Major IT vendors have provided sensor solutions to address the challenges, such as Sun EPC Network (Gupta and Srivastava 2005), SAP Auto-ID Infrastructure (Bornhovd et al. 2005), Oracle Sensor Edge Server (Oracle 2005), IBM WebSphere RFID Premises Server (IBM 2005), UCLA WinRFID Middleware. These platforms collect data from sensors and RFID tags, process them with relative simple rules with the aim at filtering and duplicate removing, and then forward them to applications. Thus the applications need more efforts to find the more valuable, actionable, and business oriented information behind all these primitive and low level data.

ALE (EPCglobal 2007) is an EPCglobal standard, and it specifies an interface through which clients may obtain filtered, consolidated RFID data from a variety of sources. Neither a consistent method to aggregate different events except for RFID events into actionable information, nor automatic response to actionable information is provided in ALE.

Accada (Accada 2007) is an open source RFID prototyping platform that implements the EPC Network specifications. It is intended to foster the rapid prototyping of RFID applications. Accada's Filtering and Collection Project implements the ALE Specification defined by EPCglobal and its members. So, the problems that exist in the ALE specification also exist in Accada.

Data mining is usually used in business intelligence and financial scenario to extract information from the enormous data sets (Xu et al. 2007; Duan et al. 2007). It is the principle of sorting through large amounts of data and picking out relevant information. First, data mining does not provide any automatic method to respond the extracted information. Second, in essence, data mining is a statistical approach, and its effectiveness of data mining is guaranteed by enormous data, which means that it is a post analysis and does not provide real time information extraction.

Ontology based knowledge discovery consists of an ontology, some data and also an inference mechanism. Ontology defines rules that formally describe how the field

of interest looks like. The data can be any data related to this field of interest that is extracted from various resources such as databases, document collections, the Web etc. The inference mechanism would deploy rules in form of axioms, restrictions, logical consequences and other various methods based on the formal definition in the ontology over the actual data to produce more information out of the existing one. Information extracted through ontology based knowledge discovery does not have to be responded, while information discovered through CEP is actionable and needs to be responded.

To discover the more valuable, actionable business level information, CEP is a good candidate. In fact, many research initiatives in the past have looked at event processing in active database research community (Liu et al. 1998). They are normally use ECA rules for event processing, and are difficult to support the RFID data processing which is characterized by temporal constraints.

As a matter of fact, event processing is not new, and can be found over the history of computing (Etzion 2005), from exception handling in programming languages to active database, from network and system management to business activity monitoring. In these research areas, events are typically captured in dedicated applications or run on their own, without significant oversight, which is used at some niches. But in event driven applications, it is another story, where events serve as the core construct. And it is very possible for event driven programming to evolve from being used at some niches to the main stream of application (Etzion 2005).

Recently there are some work that takes advantage of CEP to process the RFID data such as (Wang et al. 2006); however, they are lack of implementation details and optimization strategies.

3 Event processing in enterprise information systems

We argue that event processing will serve as a key player in building a flexible and responsive business when used either independently or together with SOA.

Application architecture of event processing in enterprise information systems is illustrated in Fig. 1.

In the daily business operations, millions of events can be produced by employees, customers, and suppliers through enterprise information systems, database and RFID. These events typically include data and messages that record activities. Specifically, enterprise information systems such as ERP, SCM, CRM and MES emit many events that are records of activities; inserting, updating and deletion of data entries in database can also be viewed as events; the majority of events come from the extraction of massive RFID data when the objects in enterprises are identified by RFID Tag in item level; all kinds of services in SOA can also emit events. In addition, events can also be intercepted from message middleware.

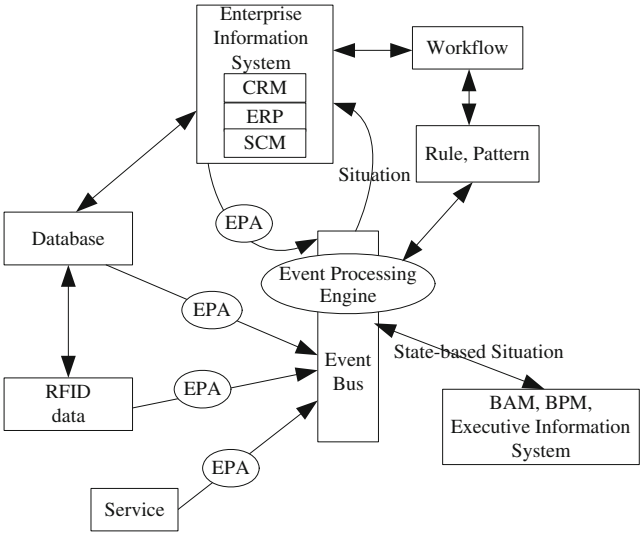


Fig. 1 Architecture of event processing in enterprise information system

These events are preprocessed by event processing agent (EPA) in terms of filtering duplicate events, rectifying errors, format match etc.

After being processed by EPA, these events are transferred into event bus, where events are aggregated into complex pattern according to certain rules, and then are sent to the subscribers. The detected complex event or situation can then drive other activities in information systems. The rules and patterns are predefined though modeling tools, and they can be transformed from workflow model to some extent.

The architecture is an EDA story, which takes advantage of event, event aggregation, event response and event subscription to improve the agility and responsiveness. The core part of this architecture is the event processing engine that implements all the ideas of CEP.

4 Complex event processing

Any piece of information that users and systems are interested in can be defined as an event, such as state transition, activity record etc. Events can be categorized into primitive event and composite event (complex event). Event type and instance are generally represented by upper and lower case letters respectively, such as E and e.

Definition 1 *Event type E can be defined as* $E=(id, a, c, t_b, t_e)$. *Each event type is uniquely identified by id;* $a=\{attr_1, attr_2, \ldots, attr_n\}$, $n\geq 0$, *is the attribute set that characterizes*

the event type; c= {e_1,e_2, …, e_n}, n≥0, is the causality vector which contains the causal events that caused this event type to happen. Causality vector facilitates the behavior analysis of the distributed system. t_b, t_e is the starting and ending time respectively.

4.1 Event meta model

Events are not isolated, and they are interrelated with other constructs of information systems, such as event operators, context and process, which is illustrated by Fig. 2.

Events can be extracted from services, database, RFID and activities. Events are generally categorized into primitive events and complex events, both of which are characterized by properties, and there are causality relationships between events. Operators combine events together to form complex events or situations. These operators include logical operators, time operators, causality operators and RFID operators.

Event context is necessary for events to be aggregated into high level actionable complex event. It is the context information that aims at facilitation of event aggregation, including semantic space, abstraction hierarchy and workflow model.

4.2 Event context

RFID events and other extracted primitive events are low level and need context to be aggregated into actionable business information. Here, event context is used to denote any information needed to transform the low level event to high level information. Event context contains such elements as semantic space, workflow model, abstraction hierarchy of different dimension such as product, time, etc.

```
<SemanticSpacename="work shift">
    <initiator event=" shift transition">
        <condition name="group group01" />
    </initiator >
    <location name="shop floor" />
    <role name="worker" />
    <state name="production" />
    <terminator event="shift transition">
        <type name="discard" />
    </terminator >
</SemanticSpace>
```

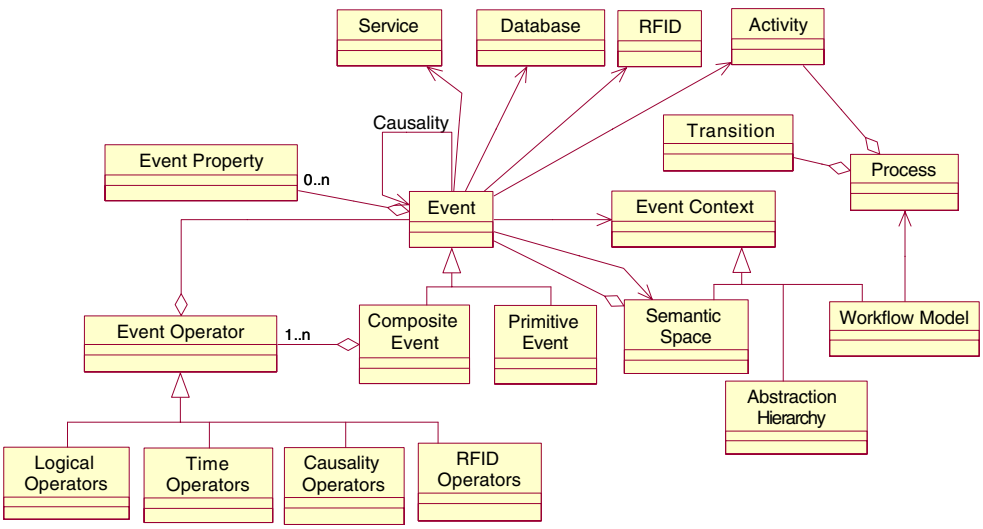
Fig. 3 An example of semantic space

Semantic space is a relatively independent context of events, which is bounded by two events called initiator and terminator. The occurrence of an initiator event initiates the semantic space, and the occurrence of terminator event terminates it. In addition, it includes people, location, role, state and other relatively independent context information. Semantic space is an extension of lifespan in Amit (Adi and Etzion 2004). An example of semantic space is shown in Fig. 3.

Only the condition defined by "condition" holds can the initiator event initiate the semantic space. The type defined by "type" in the definition of terminator event means that the complex event related to this semantic space is to be discarded if it has not been reported and the semantic space will be terminated. There is another value of "type", namely, "deferred" that means that the semantic space will be terminated after all the unreported complex events have been reported according to the available event instances and context.

To be process compliance, workflow model is necessary to be transformed into event context. The information in the workflow model can be used to route the flow of items

Fig. 2 Event meta model



```
<operator name="AND">
        <length>50</length>
        <use>first</use>
        <consume>delete</consume>
</operator>
```

Fig. 4 Definition of event operator

identified by RFID Tags, to check if the actual work route follows the model.

Similar to the data cube in data warehouse, an item can have a set of dimensions describing its characteristics, e.g., product, brand, manufacturer. Each of these dimensions has an associated concept hierarchy (Gonzalez et al. 2006).

4.3 Rules for CEP

Definition 2 *CEP rule is defined as "EVENT <complex event pattern> IF <qualification> DO <action>".*

The key word "EVENT" denotes the complex event expression. The action denoted by "DO" will be activated if conditions represented by "IF" hold, which is the key idea of CEP, namely, event response.

Definition 3 *complex event expression is composed by operands and operators. Its definition is "complex event pattern = {EACH} operator (operand {(con)}, ...) {WHERE [equivalence test], parameterized predicates} {CONTEXT context} {WITHIN, INTERVAL, AT}". For example: EACH SEQ (IN_FOAM_ROOM (reader = "05AE") x, OUT_FOAM_ROOM y) WHERE [refrigerator-id] and x.weight<y.weight CONTEXT work shift WITHIN 10 min.*

The brace "{}" denotes the optional item. "EACH" means that every instance of the complex event should be reported, otherwise, only the first complex event instance is reported.

"operand" refers to event type that is logically combined together with "operator" to create a new kind of event type. The "con" is used to retrieve specific event instance, such as (reader="05AE").

"WHERE" denotes the constraint conditions between operands, such as equivalence test and parameterized predicates. Equivalence test refers to the common attributes of all the operands, for example [refrigerator-id] means x. refrigerator-id = y. refrigerator-id. Parameterized predicates are used to restrict different operands, such as x.weight<y. weight.

"CONTEXT" specifies the context information needed to detect the complex event.

"WITHIN"denotes the time length of the complex event, "INTERVAL" denotes the specific time interval, and "AT" denotes the specific time point when the complex event happened.

The definition of operator requires the following items, as shown in Fig. 4.

1. The capacity of the operands linked by the operator. When the number of the instance exceeds the capacity, some instances must be deleted according to aging strategy. The aging strategy is the rules that sort event instances according to specific attribute such as time.
2. Event instance using and consuming strategy. Complex event is defined with event type, and each event type can have many instances. The instance using strategy is used to decide which instance can be selected as the composite instance. When the instance has been processed, it will be either deleted or preserved according to the consuming strategy.

The commonly used event operator are AND, OR, NOT and SEQ, which take the same syntax and semantic as described in (Wang et al. 2006). In addition, there are other operators in terms of time, causality, and RFID (Zang and Fan 2007)

5 CEP implementation

There are many core parts in the implementation of CEP, such as event extraction, event aggregation, event response. The most important one is the complex event detection and optimization strategies.

5.1 Data structures

Complex event expression is firstly compiled into a tree structure. For example, AND (RFIDEvent(reader = "05AE"), RFIDPartEvent), its tree structure is illustrated in Fig. 5.

The data structure used for complex event detection contains primitive event classification table, complex event classification table and shared pool of intermediate result, as illustrated in Fig. 6.

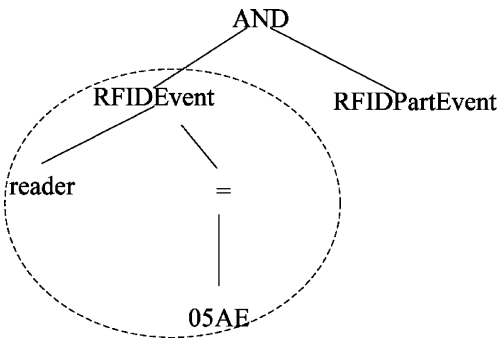


Fig. 5 Tree for complex event

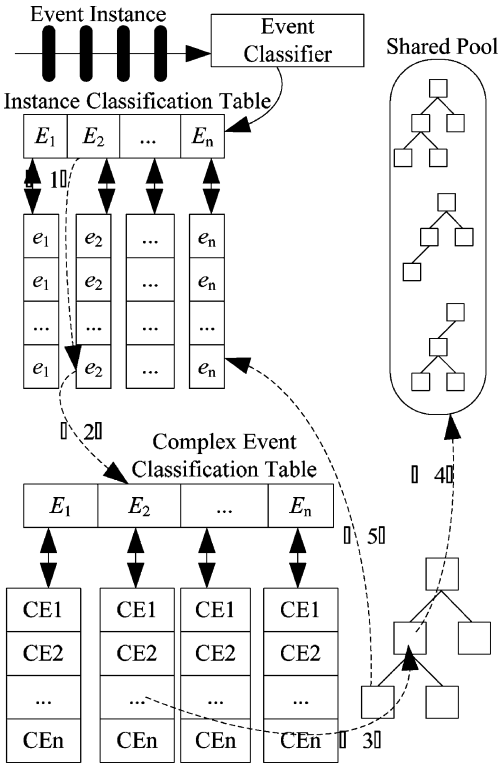


Fig. 6 Data structure for complex event detection

Primitive event classification table: each event type maintains an instance table of its own that has a configured capacity. If the number of event instance exceeds the capacity, the table will delete some instances according to aging strategies.

Complex event classification table: each event type maintains a complex event classification table, which contains all the complex event expressions that take the event type as a composite operand. When a complex event expression is compiled and verified to be correct, it will be saved to all its operands' complex event classification table. For example, $CE_1 = AND(E_1, E_2)$, $CE_2 = AND(E_1, E_3)$, the complex event classification table of E_1 contains CE_1 and CE_2; the table of E_2 contains only CE_1, and the table of E_3 contains only CE_2.

Shared pool of intermediate result: the intermediate detection results of each complex event are cached and saved for incremental detection, especially to share the detection results for different complex event expressions which have common sub-expression. The shared pool provides basis for optimization of the complex event detection. For example, CE_1 = SEQ (AND (A_1, A_2), A_3), CE_2 = OR (AND (A_1, A_2), A_3), during the process of CE_1 detection, if AND(A_1,A_2) is firstly detected, it will be cached in the shared pool as an intermediate result. When the instance of A_3 arrives, the ultimate detection result can depend on the intermediate result and A_3, thus it is not necessary to detect AND (A_1, A_2) repeatedly. At the same time, the detection of CE_2 can also use shared pool to improve detection efficiency.

The advantages of the data structure over other similar research projects such as Amit (Adi and Etzion 2004) are as follows: Shared pool of intermediate results serves as the basis for complex events detection optimization. The more are there identical sub expressions between complex event patterns, the more efficient is pattern detection. In addition, shared pool provides fundamental supports for some of the following optimization strategies.

5.2 Optimization strategies

The optimization strategies discussed here consist of classification and partition of event instances, operator implicit semantics, and constraints pushing down.

I. Classification and partition of event instances

To improve the detection efficiency of complex event, the component event instances are needed to be classified and partitioned by means of keys. Keys are used to match different instances that refer to the same entity; a key denotes a semantic equivalence among attributes that belong to different events (Adi and Etzion 2004).

Keys together with semantic space can partition the event instances in different level. A global key partitions the semantic space, and a local key divides all instances in a global partition into different groups, which is illustrated by Fig. 7.

Definition 4 *Global key = {attr |attr ∈ situation.operand.a and attr ∈ situation. semanticspace.initiator.a and a ∈ situation. semanticspace.terminator.a}. A global key is the common attributes of operands, initiator and terminator of*

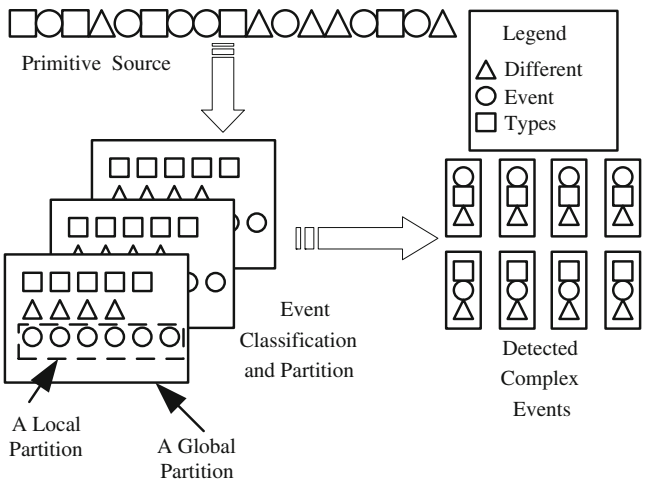


Fig. 7 Classification and partition of event instances

semantic space, where situation means complex event pattern and operand means component event type.

Definition 5 *Local key = {attr |attr ∈ situation.operand.a}. A local key is the common attributes of operands, where situation and operand take the same meaning as the above definition.*

The implementation procedures of classification and partition of event instances are as follows:

First, event contexts are specified in the definition of complex event pattern.

Second, according to the semantic space in the event context, the complex event patterns that share the same common sub expressions in the same context are compiled into aggregation trees.

Third, the event classifier divides all the related events into different global partitions and local partitions, according to the occurrence frequency of global and local keys.

Fourth, based on the different partitions, complex event patterns detection efficiency can be improved.

II. Operator implicit semantics

Theorem 1 *There is implicit semantic containment relation between operators SEQ, AND, OR, that is,*

$$SEQ(E_1, ..., E_n)\ WITHIN\ T \Rightarrow AND(E_1, ..., E_n) WITHIN\ T$$
$$\Rightarrow OR(E_1, ..., E_n)\ WITHIN\ T$$

Theorem 2 *There is implicit semantic exclusive relation between SEQ (E_1, $!E_2$) WITHIN T and SEQ(E_1, E_2) WITHIN T. They can not happen at the same time. "!" means not.*

Based on the theorems, implementation procedures are as follows: when a complex event pattern is validated to be true, the inference complex event pattern can also be validated. Alternately, when a pattern needs to be validated, the shared pool is traversed to find whether there is a pattern that can infer the validation of the former one.

III. Constraints pushing down

In the process of complex event computation, there could be many intermediate results, some of which satisfy the constraints and others do not. To improve the detection efficiency, it is essential to reduce the intermediate results as soon as possible. An effective measure is to push down the complex event constraints and exclude the unrelated intermediate results as early as possible. For example, SEQ (AND (A, B), C) WITHIN 30 equivalents to SEQ (AND(A, B) WITHIN 30, C) WITHIN 30.

Another constraint that needs to be pushed down to reduce intermediate results is equivalence test (Fig. 8), such as

$$AND(A, B)\ WHERE\ [attr_1, attr_2].$$

It is necessary to reduce the number of candidates according to the equivalence attribute and partition the candidates to improve the detection efficiency.

Definition 4 *If there are k equivalence attributes, and they have i_1, i_1,...,i_k partitions respectively, then the following sets are given:*

$$I_{1,1} = \{e|e.attr_1 = value_1\},$$
$$\ldots,$$
$$I_{1,i_1} = \{e|e.attr_1 = value_{i_1}\},$$
$$\ldots,$$
$$I_{k,i_k} = \{e|e.attr_k = value_{i_k}\},$$

For example, in Fig. 11,

$$I_{1,1} = \{a_1, b_3, b_6, a_7\},$$
$$I_{1,2} = \{a_2, a_4, b_5\},$$
$$I_{2,1} = \{a_2, b_3, b_5, a_7\},$$
$$I_{2,2} = \{a_1, a_4, b_6\}.$$

Then,

$$R_{1,1} = I_{1,1} \cap I_{2,1} = \{b_3, a_7\},$$
$$R_{1,2} = I_{1,1} \cap I_{2,2} = \{a_1, b_6\},$$
$$R_{2,1} = I_{1,2} \cap I_{2,1} = \{a_2, b_5\},$$
$$R_{2,2} = I_{1,2} \cap I_{2,2} = \{a_4\}.$$

Thus, the detection result is $R_{1,1}$,$R_{1,2}$,$R_{2,1}$.

Based on the definition, the implementation procedures are as follows:

During the process of the partition of the event instances, different sets are generated according to the equivalence attributes. If the intersections of these sets are not empty sets, and the number of the elements of them equals the number of the equivalence attributes, the detection results can be discovered in the process of event instances partition.

instance	a1	a2	b3	a4	b5	b6	a7
attr1 value	1	2	1	2	2	1	1
attr2 value	3	2	2	3	2	3	2

partition	Att r1 =1	Attr1 =2	Attr2 =2	Attr2 =3
	a1	a2	a2	a1
	b3	a4	b3	a4
	b6	b5	b5	b6
	a7		a7	

Fig. 8 Equivalence test constraint

```
1.  PARSE(cmplx_expr)
2.  SAVE_RELATED_CMPLX_EVENT()
3.  PUSHING_DOWN_CONSTRAINTS()
4.  RECV_EVENT_INSTANCE()
5.  for each event instance e
6.    FIND_RELATED_CMPLX_EVENT()
7.     for each complex event expression cmplx_expr
8.       tree=TREE(cmplx_expr)
9.       t=FIND_IDENTICAL_TREE(tree)
10.      t=t FIND_SEMANTIC_CONTAINMENT(tree)
11.      t=t FIND_ SEMANTIC_EXCLUDE(tree)
12.      if it is null
13.        GET_CANDIDATE_LIST()
14.        if there is equivalence test
15.          PARTITION_CANDIDATE_LIST()
16.        end if
17.    REPORT_EVENT()
18.    SAVE_RESULT_TO_SHARE_POOL()
```

Fig. 9 Complex event detection algorithm

5.3 Algorithm

The complex event detection algorithm is illustrated by Fig. 9.

Lines 1–3 represent the initialization, including syntactical and lexical analysis, complex event classification table preparation, and pushing down the constraints. Lines 4–18 represent the complex event detection algorithm. First, event instances are received and save to instance classification table (line 4), and then look up the complex event classification table (line 6) for each instance. Each related complex event is evaluated for happen or not (lines 7–18). After the tree structure is obtained(line 8), if there is the same intermediate result in the shared pool (line 9), it is not necessary to proceed other detection procedures. Then check if there is implicit semantic containment (line 10) and exclusive (line 11) complex event. If there are no such intermediate results, it is necessary to retrieve the candidates (lines 12–16), where these candidate instances are classified and partitioned by keys and semantic space specified in the definition of complex event pattern. During this process, if there is an equivalence test, the instance candidates can be partitioned. The sub complex events are evaluated according to the operator semantics. If the result is true, it is reported and then cached in the shared pool.

The time complexity of this algorithm is O(nmk) in the worst case, where m, n, k is the number of complex event patterns, event instances and event types respectively. Because m and k are relatively fixed values, and not very large in the system, the processing time is proportional to n.

In terms of implementation, Amit and Esper have no optimization strategies as proposed here, including shared pool, operator implicit semantics and constraint pushing down; as a result, better performances can be expected from our method.

6 Performance evaluations

The prototype is implemented in Java language, which is called RTE-CEP, and the test environment is as follows: Pentium IV 3.0 GHz CPU, 1G memory.

There are 20 event types for performance test, that is, E_1, E_2,⋯,E_{20}, each of which has simple attributes and methods to get their attributes. There are also 500 complex event expressions. To test the performance, the over loaded test are used here, namely, each event type is used by at least one complex event expression that uses it as a composite event. Each complex event contains AND(AND(E_1, E_4), AND (E_2, E_3)) to test the function of shared pool, that is, each of E_1, E_2, E_3, E_4 event type has 50 complex event expression in their complex event classification table.

The event instances are generated randomly. For each number of input events, 10 tests are conducted and its mean value is utilized for analysis.

Currently, there is no generally accepted benchmark for CEP, and Accada and Esper are the available candidates for performance comparison. Accada is the implementation of ALE, event sources of which are primarily RFID events, and there is no intrinsic mechanism of event response in accada project; while Esper is more general and more similar to CEP. So, the open source software Esper 0.8.5 (Bernhardt 2005) is used for performance comparison.

The comparison of event processing time is shown in Fig. 10. When the number of input event is less than 700000, RTE-CEP has a slight advantage over Esper; when the number exceeds 700000, the processing time of RTE-CEP is longer than that of Esper, the reason of which is that RTE-CEP use content incremental detection method and the number of detected complex events is much larger than

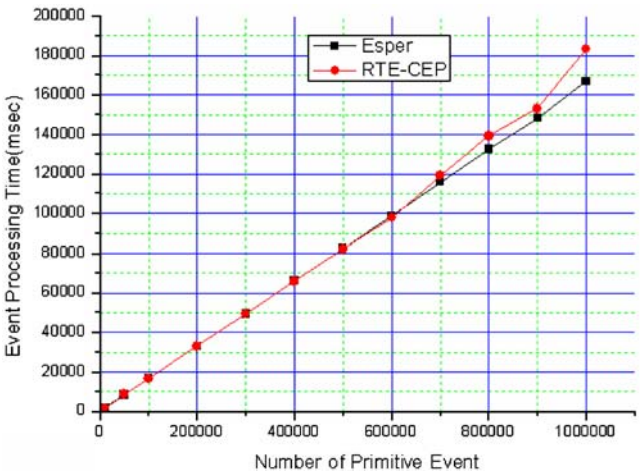


Fig. 10 Event processing time of two methods

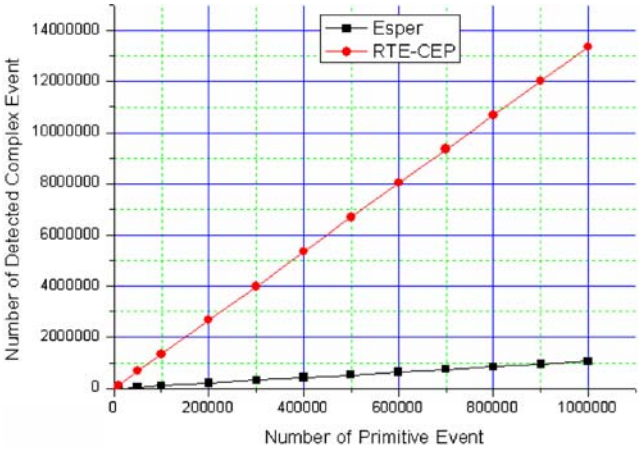


Fig. 11 Number of detected complex event of two methods

Esper. The processing capacity per second of these two methods is similar, namely, about 6000/s in the test environment here.

Figure 11 shows the number of detected complex events of the two methods. RTE-CEP has apparent advantage over Esper because of the content incremental detection method of RTE-CEP. As for AND(E_1, E_4), there are such event sequences as $e_1^1, e_3^2, e_4^3, e_4^4$, where e_i^j means the event instance of event type E_i at j time point, RTE-CEP has detected two complex events((e_1^1, e_4^3), (e_1^1, e_4^4)), while Esper detected only one ((e_1^1, e_4^3)).

In terms of the number of detected complex event per second, there is apparent difference between these two methods. RTE-CEP method can generate 80000/s, but Esper can only generate 6000/s.

In general, our method is better than Esper in terms of capabilities of event processing because of our elaborated data structures and optimization strategies.

7 Applications

RTE-CEP is implemented in a major refrigerator manufacturer in China to address such problems as mismatch between parts and refrigerator, inaccurate assessment of work time and quality, and product shipment error.

The simplified workflow model of the refrigerator manufacturing process is shown in Fig. 12.

The body and liners are prepared in parallel. When they are both ready, the activities of body assembly, foaming, condensation assembly, foaming check and packaging are followed. The "Foaming Check" activity is a "or split" node, where if the foaming is qualified the next activity will be "Packaging" and if the foaming is not qualified the refrigerator must be re-foamed.

Each important part is identified by RFID tag, and RFID readers are deployed in all the working stations to emit RFID events such as "REFRIGERATOR", "COMPRESSOR01", etc. In addition, events can be extracted from MES database. All these events are preprocessed by EPA, and then are aggregated into business level, actionable information by the event processing engine according to the predefined rules.

First, RTE-CEP is useful to address problems in operational level, such as mismatch between parts and refrigerator, inaccurate assessment of work time and quality, and product shipment error.

When a refrigerator is about to assemble a compressor in the station of "Condensation Assembly", the types of refrigerator and compressor are identified by RFID reader. Complex event "A = AND (REFRIGERATOR, OR (COMPRESSOR01, COMPRESSOR02)) WHERE [compressor_type] INTERVAL 10 min" can assure the match between refrigerator and compressor. If A equals NULL, an alert will be activated to tell the worker that the wrong type of compressor is being assembled.

To assess work time and quality accurately, complex event "EACH SEQ (WORKTIME, QUALITY) WHERE [person_id, station] INTERVAL AUGUST" is used. The event can analyze work time and quality in every station of every person according to the accurate record of primitive event WORKTIME and QUALITY.

The product shipment error can be addressed by such complex event as "EACH SEQ (TRUCK, EXIT-READING (type ! = context. type))", where TRUCK event refers to the

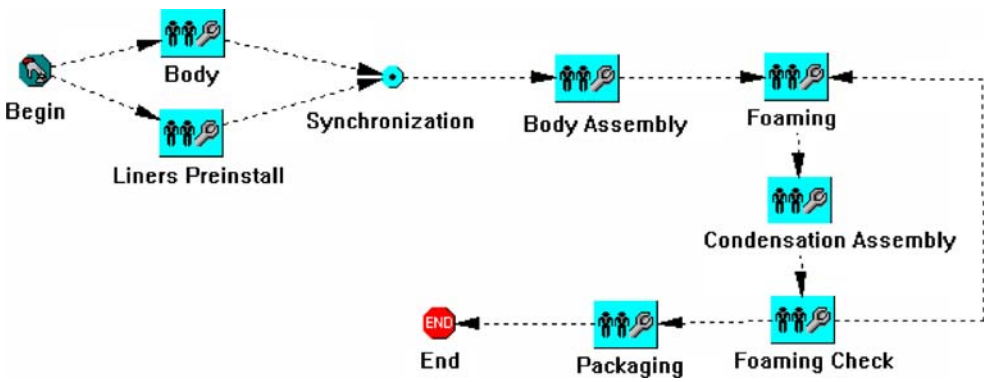


Fig. 12 An example of workflow model in refrigerator manufacturing plant

fact that the vehicle is ready and event "EXIT-READING (type ! = context.type)" assure that the right product is shipped to right customer.

Second, RTE-CEP is helpful at management level. It can facilitate the analysis of such concerns of executives as why refrigerators operated by the third group of workers tend to breakdown. The facilitation is achieved because event processing can maintain the causality vector of complex event, and the concerns can be drilled down to the event sources following the causality vector. In addition, the detected event can be delivered to the subscribers in real time manner, so managers can make quick decisions.

In general, CEP is critical for enterprise information systems based on RFID in the following aspects.

1. CEP can discover more high level, actionable information behind RFID data and other business data.
2. CEP can extract the time, causal and hierarchical relationships between events. It facilitates the analysis of system behaviors and can pinpoint where the root cause is.
3. CEP can improve the system responsiveness in terms of timely delivery of events.

8 Conclusions and future work

Enterprises have to be increasingly agile and responsive to address challenges posed by fast moving market. We argue that with the evolvement of software architecture into SOA and the adoption of RFID, event processing can be an important player in enterprise information systems in that it is easy to construct decoupled, many to many communications systems, and it facilitates event aggregation to derive more actionable information.

In this paper, the idea where event processing can fit in enterprise information systems is put forth. The event meta model and rules of CEP are formulated. The system implementation is discussed in detail. Performance evaluation and application show that event processing is effective to increase agility and responsiveness.

References

Accada (2007). Accada prototyping platform. http://www.accada.org/. Accessed 21 November 2007.

Adi, A., & Etzion, O. (2004). Amit—the situation manager. *VLDB*, *13* (2), 177–203.

Bernhardt, T. (2005). Esper Event Engine. http://esper.codehaus.org. Accessed 21 November 2005.

Bornhovd, C., Tao, L., Haller, S., & Schaper, J. (2005). Integrating smart items with business processes: An experience report, in Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS), 8, pp. 227–235.

Duan, L., Xu, L., Guo, F., Lee, J., & Yan, B. (2007). A local-density based spatial clustering algorithm with noise. *Information Systems*, *32*(7), 978–986.

Etzion, O. (2005). Towards an event-driven architecture: An infrastructure for event processing position paper, in Proceedings of the 1st International Conference on Rules and Rule Markup Languages for the Semantic Web (RuleML), pp. 1–7.

Fan, Y., Huang, C., Wang, Y., & Zhang, L. (2005). Architecture and operational mechanisms of networked manufacturing integrated platform. *International Journal of Production Research*, *43*(12), 2615–2629.

Gonzalez, H., Han, J., & Li, X. (2006). FlowCube: Constructing RFID FlowCubes for multidimensional analysis of commodity flows, in Proceedings of VLDB, 834–845.

Gupta, A., & Srivastava, M. (2005). Developing Auto-ID solutions using Sun Java System RFID software. http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/sjsrfid/RFID.html. Accessed 18 December 2005.

Hou, H., Xu, S., & Wang, H. (2007). A study on X party material flow: the theory and applications. *Enterprise Information Systems*, *1*(3), 287–299.

IBM WebSphere. (2005). IBM Corporation. http://www-306.ibm.com/software/. Accessed 21 November 2005.

Li, L., et al. (2008). Assessing the effects of manufacturing infrastructure preparation prior to enterprise information systems implementation. *International Journal of Production Research*, *46*(6), 1645–1665.

Liu, G., Mok, A. K., & Konana, P. (1998). A unified approach for specifying timing constraints and composite events in active real-time database systems, in Proceedings of the 4th IEEE Real-Time Technology and Applications Symposium, pp. 199–208.

Luckham, D. (2002). *The power of events: an introduction to complex event processing in distributed enterprise systems*. Addison Wesley: Boston, MA.

Maréchaux, J. L. (2006). Combining service-oriented architecture and event-driven architecture using an enterprise service bus. http://www-128.ibm.com/developerworks/webservices/library/ws-soa-eda-esb/index.html#Resources. Accessed 17 September 2006.

Nath, B., Reynolds, F., & Want, R. (2006). RFID technology and applications. *IEEE Pervasive Computing*, *5*(1), 22–24.

Ohkubo, M., Suzuki, K., & Kinoshita, S. (2005). RFID privacy issues and technical challenges. *Communication of the ACM*, *48*(9), 66–71.

Oracle Sensor Edge Server. (2005). Oracle Corporation. http://www.oracle.com/technology/products/sensor_edge_server/index.html. Accessed 21 November 2005.

Wang, F., Liu, S., Liu, P., & Bai, Y. (2006). Bridging Physical and Virtual Worlds: Complex Event Processing for RFID Data Streams, in Proceeding of the 10th International Conference on Extending Database Technology (EDBT), pp. 588–607.

Warfield, J. (2007). Systems science serves enterprise integration: a tutorial. *Enterprise Information Systems*, *1*(2), 235–254.

Xu, L. (2007). Editorial. *Enterprise Information Systems*, *1*(1), 1–2.

Xu, L., Li, Z., Li, S., & Tang, F. (2007). A decision support system for product design in concurrent engineering. *Decision Support Systems*, *42*(4), 2029–2042.

Xu, L., Wang, C., Luo, X., & Shi, Z. (2006). Integrating knowledge management and ERP in enterprise information systems. *Systems Research and Behavioral Science*, *23*(2), 147–156.

Zang, C., & Fan, Y. (2007). Complex event processing in enterprise information systems based on RFID. *Enterprise Information Systems*, *1*(1), 3–23.

Zhang, Y., & Bhattacharyya, S. (2007). Effectiveness of Q-learning as a tool for calibrating agent-based supply network models. *Enterprise Information Systems*, *1*(2), 217–233.

Zhang, H., Kishore, R., Sharman, R., & Ramesh, R. (2007). Agile integration modeling language: a conceptual modeling grammar for agile integrative business information systems. *Decision Support Systems*, *44*(1), 266–284.

Chuanzhen Zang received the Ph.D. degree in Control Theory and Engineering from the Department of Automation, Tsinghua University. His research interests include enterprise information systems, workflow, and RFID (Radio Frequency Identification), etc.

Yushun Fan received his B.S. degree in Automatic Control from Beijing University of Aeronautics and Astronautics, Beijing, China, in 1984, and his M.S. and Ph.D. degrees in Control Theory and Application from Tsinghua University, Beijing, in 1987 and 1990, respectively. He is currently Professor of the Department of Automation, Vice Director of the System Integration Institute, and Director of the Networking Manufacturing Laboratory, Tsinghua University. His research interest includes enterprise modeling methods and optimization analysis, business process re-engineering, workflow management, system integration and integrated platform, object-oriented technologies and flexible software systems, Petri nets modeling and analysis, workshop management and control. He authored nine books in enterprise modeling, workflow technology, intelligent agent, and object oriented complex system analysis, computer integrated manufacturing, respectively, and published more than 250 research papers in journals and conferences. He is a member of the IFAC Advanced Manufacturing Technology Committee. From September 1993 to March 1994, he was a Visiting Scientist at the University Bochum, Germany, supported by Federal Ministry for Research and Technology. From April 1994 to July 1995, he was a Visiting Scientist, supported by Alexander von Humboldt Stiftung, at Fraunhofer Institute for Production System and Design Technology (FhG/IPK), Germany. Dr. Fan served on the Program Committees of the 1992 International Symposium on CIM, Beijing, China, 1997 IEEE International Conference on Factory Automation and Emerging Technology, Los Angeles, CA, and 2002 International Workshop on Emergent Technologies in Engineering Cooperative Information Systems, Beijing, China.

Renjing Liu received the Ph.D. degree from the Department of Management Science and Engineering, Xian Jiaotong University. His research interests include Information Systems, Information Resource Management, Decision Support System, etc.