http://dx.doi.org/10.6109/ijice.2011.9.6.712

A Framework for Supporting RFID-enabled Business Processes Automation

Mikyeing Moon, *Member, KIICE*

***Abstract*—Radio frequency identification (RFID) is an established technology and has the potential, in a variety of applications, to significantly reduce cost and improve performance. As RFID-enabled applications will fulfill similar tasks across a range of processes adapted to use the data gained from RFID tags, they can be considered as software products derived from a common infrastructure and assets that capture specific abstractions in the domain. This paper discusses a framework that supports the development of RFID-enabled applications based on a business process family model (BPFM), explicitly representing both commonalities and variabilities. To develop this framework, common activities are identified from RFID-enabled applications and the variabilities in the common activities are analyzed in detail using variation point concepts. Through this framework, RFID data is preprocessed, and thus, RFID-enabled applications can be developed without having to process RFID data. Sharing a common model and reusing assets to deploy recurrent services may be considered an advantage in terms of economic significance and the overall product quality afforded.**

***Index Terms*— framework, business process family model, RFID-enabled application, commonality, variability**

I. INTRODUCTION

RADIO frequency identification (RFID) technology is considered the next step in the revolution affecting supply-chain management, retail, and beyond. At present, the reliability of RFID hardware is improving, the EPCglobal body is developing widely accepted standards [1,2], and middleware firms are providing software to link RFID data with business applications. Therefore, firms must modify relevant business processes to gain the full benefits from RFID [3,4,5,6]. Although RFID middleware deletes duplicate readings from the same tag and helps manage the flow of data, developers are required to implement systems that derive meaningful high-level data, containing information that is more useful for the application than the simple RFID data itself. In implementing an RFID-enabled application, the developer must collect RFID data, access the data server to retrieve reference information for the RFID data, and process business logic. Most RFID-enabled applications will include these activities in processes adapted to use the raw data gained from RFID tags. Therefore, these activities may be considered as common components across the whole family of applications sharing the same architecture. As a result, RFID-enabled applications can easily be constructed by reusing these common components. However, actual management research and academic management literature on the use of RFID-enabled applications is still scarce. Work related to RFID-enabled applications focuses mainly on case studies and discussions of business opportunities. Thus, in order to develop RFID-enabled applications effectively, it is necessary to systematically identify common activities and to define a variability mechanism.

In this paper, we suggest a product-line engineering approach that systematically supports RFID-enabled application development. More concretely, we describe a method for developing a business process family model (BPFM) for RFID-enabled applications that reflects the characteristics of process-oriented software. A BPFM describes the business processes of common skeletons in the systems and then describes somewhat abstract types of slightly different parts in the systems. The model can explicitly represent common and variable activities in RFID-enabled applications. It also identifies the variation points in the activities. We have developed a supporting tool to support RFID application development based on the proposed RFID BPFM. The tool provides the common reusable parts of the BPFM for RFID data processing and helps specify commonality and variability in RFID BPFM.

II. BPFM FOR RFID-ENABLED APPLICATIONS

The development of product lines consists of two main phases [7,8]. The first phase is called domain engineering. The important task in this phase is to develop a BPFM describing both the commonalities and variabilities of the applications belonging to the domain. The second phase is called application engineering. This phase uses the artifacts provided during domain engineering to quickly

Manuscript received August 16, 2011; revised September 7, 2011; accepted September 17, 2011.

Mikyeong Moon is with the Division of Computer & Information Engineering, Dongseo University, Busan (Email: mkmoon@dongseo.ac.kr)

produce applications in the domain. Namely, a specific application in the RFID domain is developed. Fig. 1 shows how the reusable BPFM and concrete business process models, based on these assets, are developed.

The domain layer comprises RFID generic activities and the domain activities. The RFID generic activities that process the raw RFID data are common to many RFID domains. Even though they are commonly used in various RFID domains, they may have variabilities that need to be altered by the domain developer and the application developer. These RFID generic activities will be discussed further in Section A. The domain activities may be identified through analysis for an individual domain. The domain activities can explicitly represent both commonalities and variabilities. The RFID BPFM is developed with these RFID generic activities and the domain activities. This will be discussed further in Section B.

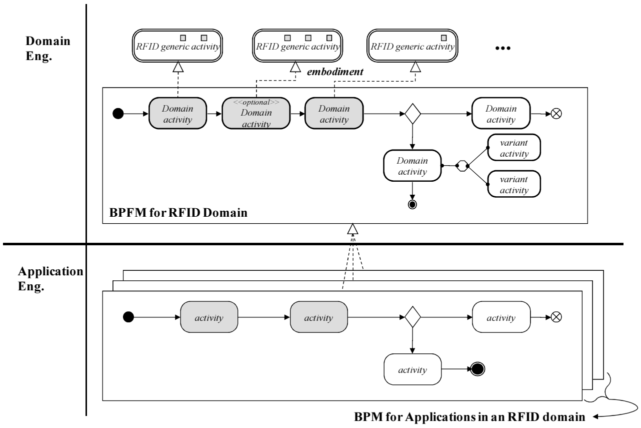


Fig. 1. BPFM for RFID-enabled applications.

A. RFID generic activity

The RFID generic activities are categorized as one of three types—trigger activities, reference activities, and rule manager activities. To be able to graphically represent the variability information of an RFID generic activity, we have expanded the activity modeling element in the UML activity diagram as depicted in Fig. 2. RFID generic activities are represented as rounded rectangles. In addition, a hexagon with the text "R" is attached to a rounded rectangle to distinguish RFID generic activities from other activities. The variation points in an RFID generic activity are represented by doughnut symbols in the rectangle. Each doughnut symbol is associated with variants, which are listed in a box. The graphical notation for the variants shows the variation point cardinality as an upper suffix to the braces. The association is a model element that represents the relationship between a variation point and the variants. A solid line indicates that at lest one variant should be selected at the variation point: while a dash line indicates that a variant may or may not be selected at the variation point. A variation point without variants indicates an open boundary, that is, new variants can be realized at a variation point at a later time (during application development). The underlined variant is the default variant that should always be realized.

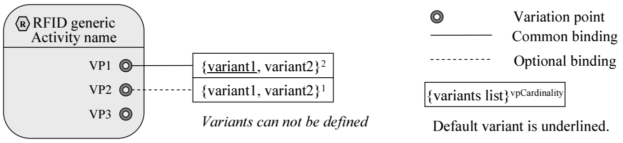

R
RFID generic Activity name
VP1
VP2
VP3
{variant1, variant2}²
{variant1, variant2}¹
Variants can not be defined
Variation point
Common binding
Optional binding
{variants list}vpCardinality
Default variant is underlined.

Fig. 2. Graphical notations for variability of activity.

1) Trigger activities

A trigger activity specifies processes that request RFID data [9]. It consists of elements that describe the RFID data of interest and the RFID reader-control information related to the events, such as a start or stop trigger for an event cycle, the repeat period, and the duration. In addition, it specifies the element that receives the RFID data from the RFID middleware. TimeTrigger events make it easy for developers to build two different classes of RFID-enabled applications: real-time applications and batch-oriented applications.

① EventTrigger Activity

This activity includes RFID data gathering techniques. RFID data is requested according to the operational mode and is processed according to the elements of the event specification. Therefore, the operation mode (event request method) and elements of the event specification—logical readers, boundarySpec, and Reportspecs—are identified as variation points. The Event request methods are subscription, polling a predefined ECSpec, and polling a new ECSpec, which are interaction rules dictating how applications should receive data from RFID middleware. The LogicalReaders define only an abstracted layout that frees applications from needing to know the explicit physical layout of their environment. The boundarySpec indicates when to start collecting data and when to stop collecting data. These start and stop conditions can be time-, stability-, or trigger-based. The data that has been collected during the event cycle is filtered and grouped and a set of reports are generated. The Reportspecs dictate what tags are included in any given report.

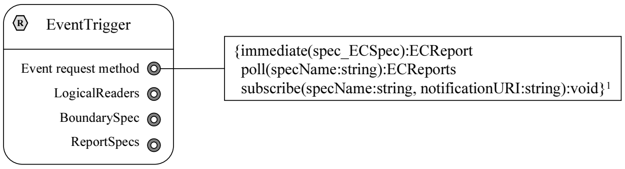

R
EventTrigger
Event request method
LogicalReaders
BoundarySpec
ReportSpecs
{immediate(spec_ECSpec):ECReport
poll(specName:string):ECReports
subscribe(specName:string, notificationURI:string):void}¹

Fig. 3. EventTrigger activity.

② TimeTrigger Activity

RFID data is processed at the setting time or during the setting period. The variation point of this activity is the trigger time, which may be delayed to the application development time, or may even be varied in real-time.

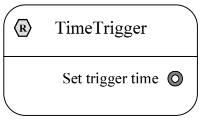


Fig. 4. TimeTrigger activity.

2) Reference activities

In general, RFID data directly output from RFID middleware contain only simple information such as the logical reader name, EPC (Electronic Product Code) and time [10]. However, as it is often necessary for the system to drive meaningful high-level events, the information gathered from the middleware is insufficient. In an RFID-enabled application, to obtain the product details, the reference data related to the RFID data must be accessed. These reference data are retrieved from an information service. A reference activity comprises EPCIS [11], ONS [12], and EPCIS Discovery Service (DS) activities, which retrieve data from EPCIS, ONS, and EPCIS DS [13], respectively. EPCglobal defines an ONS and EPCIS to exchange product level information on networks for RFID data and product data.

① EPCIS Activity

An EPCIS repository is a networked database that stores the additional data associated with a tagged object; it provides a standard interface for access and permanent storage of EPC-related data for read-and-write access by authorized parties. This activity defines processes to retrieve data from EPCIS. The query statement for EPCIS is identified as a variation point.

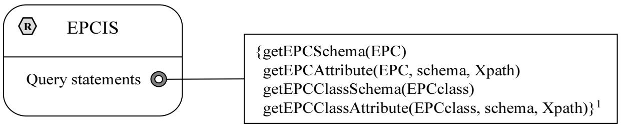

EPCIS
Query statements
{getEPCSchema(EPC)
getEPCAttribute(EPC, schema, Xpath)
getEPCClassSchema(EPCclass)
getEPCClassAttribute(EPCclass, schema, Xpath)}1

Fig. 5. EPCIS activity.

② ONS Activity

Object Name Service (ONS) indicates the location at which information about a particular EPC is available. In other words, it is a lookup service that takes an EPC as input and produces as output the address (URI) of the EPCIS repository. The ONS activity is responsible for accessing ONS. The variation point of this activity is EPC.

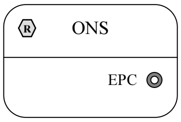


Fig. 6. ONS activity.

③ EPCIS DS Activity

EPCIS DS provides a method for custodians of a particular RFID tag data to update a register within the EPCIS DS to indicate that they are in possession of data related to that particular RFID tag. The register may contain a list of the EPCIS URLs where such information may be obtained. This activity defines processes to retrieve data for tracking and tracing from EPCIS DS. The query statement for EPCIS DS is identified as a variation point.

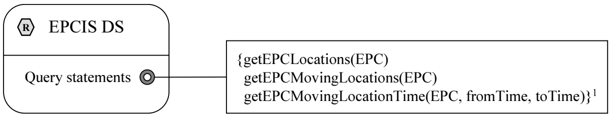

EPCIS DS
Query statements
{getEPCLocations(EPC)
getEPCMovingLocations(EPC)
getEPCMovingLocationTime(EPC, fromTime, toTime)}1

Fig. 7. EPCIS DS activity.

3) Rule manager activities

A business rule constrains some aspects of the business that are related to the RFID data and the reference data. To prevent inaccurate data from being transmitted to applications, the business rules deal beforehand with erroneous or missing information. Rule manager activities process the business rules that are required in the applications. These are composed of several conditions (rule checker activities) and actions (service mapper activities). For example, if a shipment of 24 cases is expected but only 20 tags are read when it arrives, the system can send an alert so that the operator can check the pallet. It automatically triggers any alerts that have been incorporated into the business rules.

① Rule checker Activity

This represents a business rule, which is required in the RFID applications. This activity specifies the RFID business conditions, enabling filtering of irregular RFID data. That is, this function as a semantic filter. If the conditions of the rule are not satisfied, the operator is notified that an execution has occurred. Here, the rule condition is identified as a variation point.

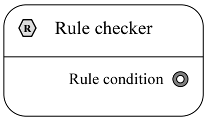


Fig. 8. Rule checker activity.

② Service mapper activity

This activity is responsible for intermediate actions, through which business events can be transferred to specific processes and the processes can be appropriately executed. It is designed to receive multiple streams of business events and to invoke business processes following the specified mapping information through the web services. Using web-services technology, the business processes are invoked by using WSDL (Web Services Description Language) [14] and the end point address of the deployed process. The WSDL of partner link defines services as a collection of network endpoints or ports, where the port types are abstract collection of operations, and the concrete protocol and data format specifications for a particular port type constitutes a reusable binding. The variation points of this activity are shown in Fig. 9.

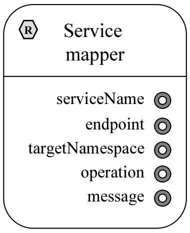


Fig. 9. Service mapper activity.

B. RFID BPFM

The BPFM can be modeled as an UML activity diagram. UML2.0 activity diagrams are typically used for business process modeling, for modeling the logic captured by a single use case or use scenario, or for modeling the detailed logic of a business rule [15,16]. However, model elements of UML2.0 activity diagram should be expanded to represent the variability of activities flows. TABLE shows the model elements for expanded activity diagram.

TABLE I
MODEL ELEMENTS FOR EXPANDED ACTIVITY DIAGRAM

Modeling Elements	Notation	Description
Variation point	vp	The variation point model element is attached to a domain activity to explicitly represent the variation point in the domain activity. The properties of a variation point are denoted in text form.
Variation point binding	●—●	The variation point binding is a model element to represent the relationship between a variation point and variants.
Variants Region	□	The variantsRegion is a model element to contain a set of variants.

Additionally, Fig. 10 shows the graphical notation for a domain activity, which is an embodiment of a RFID generic activity. Domain activities are represented as rounded rectangles, as previously. In addition, a square with the text "D" is attached to the rounded rectangle to distinguish domain activities from other activities. The variation points in RFID generic activities are bound to variants according to the requirements of the RFID-enabled applications. If a variation point is bound to any variants, the doughnut symbol is replaced with a solid circle and the variant is represented in the domain activity rectangle as in the case of VP2 in Fig. 10. If a variation point has open-bounded variants, the specific value may be bound to a variation point, as in the case of VP3 in Fig. 10. However, occasionally, the variation points may be remained as before, as in the case of VP1 in Fig. 10. The binding of these variation points is delayed until the application development phase.

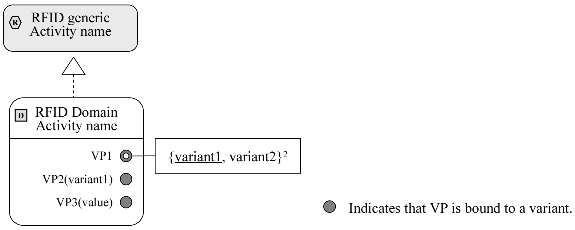

R
RFID generic Activity name
D
RFID Domain Activity name
VP1
VP2(variant1)
VP3(value)
{variant1, variant2}²
Indicates that VP is bound to a variant.

Fig. 10. Graphical notations for domain activities which embody RFID generic activities.

We describe a RFID BPFM in detail for a case study on RFID-enabled smart-shelf applications, i.e., the smart-shelf domain. Smart shelves are equipped with readers that inform the store staff of when the shelves must be replenished. The system provides complete data acquisition control over the products on the shelves with real-time event triggering and automatic inventory capture. Furthermore, smart shelves automatically recognize when the expiration date of a product has been exceeded and inform the staff accordingly. The BPFM can be completed through variability analysis of domain activities and their flows.

1) Variability analysis of domain activities

First, most of the activities that should be processed by RFID-enabled applications are identified. The activities can be driven by the embodiments of RFID generic activities or by an analysis of the specific RFID domain. Embodiment is defined as the process through which RFID generic activities are realized depending on requirements of the specific domain. In the smart-shelf domain, the variation point in the EPCIS activity—a query statement— is bound to the variant getEPCAttribute. The Rule checker activity is embodied as three different activities: "*Check product quantity*," "Check expiration date of product," and "Check the misplaced product," as shown in Fig. 11.

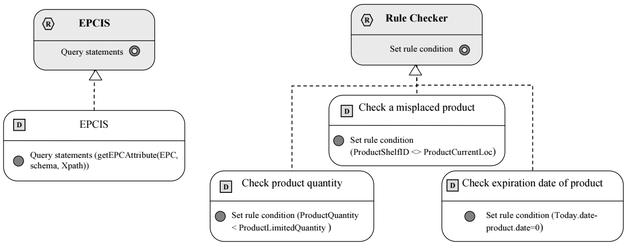

R
EPCIS
Query statements
D
EPCIS
Query statements (getEPCAttribute(EPC, schema, Xpath))
R
Rule Checker
Set rule condition
D
Check a misplaced product
Set rule condition (ProductShelfID <> ProductCurrentLoc)
D
Check product quantity
Set rule condition (ProductQuantity < ProductLimitedQuantity)
D
Check expiration date of product
Set rule condition (Today.date-product.date=0)

Fig. 11. Embodiments of EPCIS activity and Rule Checker activity.

The activities that are extracted by analyzing the RFID domain are the "*Update inventory bookstock*" activity, "*Find product type in backroom*" activity, etc. In addition, a smart shelf can provide an alert when it is time to restock or when a potential shoplifting situation arises, such as when an unusually large number of products have been removed simultaneously. Therefore, the "*Alert staff with picking list*", "*Alert staff with removing list*," and "*Alert staff with rearrangement message*" activities are identified.

The variability of a domain activity is realized as a CV_property, an attribute of the domain activity. A CV_property is defined as a property that can be common or optional. A common property means that the activity should be realized in most applications. The stereotype <<common>> may be used to explicitly represent the common CV_property, but also may be omitted. The optional property means that the activity may only be realized in a specific application. The stereotype <<optional>> is introduced to explicitly represent the optional CV_properties of the domain activity. Namely, the domain activity may or may not appear in the application architecture depending on the extent to which the domain activity has the property of being common or optional. For example, smart shelves on a sales floor can provide an alert when it is time to restock or of a potential shoplifting situation if an unusual number of products are removed simultaneously. In addition, they also automatically recognize when the product expiry dates have been exceeded, and inform the staff accordingly. In this example, only some applications in the smart-shelf domain may have the function of managing expiry dates, so the property "*Check expiry date of product*" is determined as optional. Fig. 12 shows a part of the identified domain activities with the CV_property.

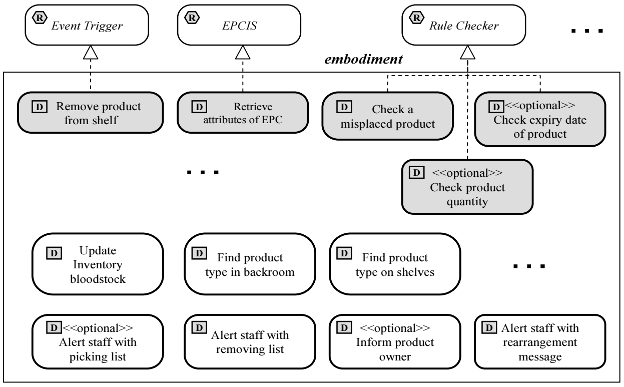

Event Trigger
EPCIS
Rule Checker
. . .
embodiment
D Remove product from shelf
D Retrieve attributes of EPC
D Check a misplaced product
D <<optional>> Check expiry date of product
. . .
D <<optional>> Check product quantity
D Update Inventory bloodstock
D Find product type in backroom
D Find product type on shelves
. . .
D <<optional>> Alert staff with picking list
D Alert staff with removing list
D <<optional>> Inform product owner
D Alert staff with rearrangement message

Fig. 12. Domain activities in the smart shelf domain.

2) Variability analysis of domain activities flows

The variability of domain activities flows is realized as a variation point in more detail. The properties of a variation point —variation point cardinality (vpCardinality), variation point type (vpType), constraints, etc. — are denoted in text form. The number of variants that can be chosen is the variation point cardinality. It indicates how many variant domain activities can be realized in a specific application. A variation point type can be classified into three types as follows:

① Selection decision vpType

Alternative variants of activities can be generalized into an abstract domain activity. Then, this generalized domain activity will have one or many variant domain activities. The relationship between a generalized activity and variants activities is modeled with a variation point. In ❶ of Fig. 13, activity A3 represents a generalized activity and A31, A32, and A33 represent variants activities. This type of a variation point means that various business processes are made by selection of variants activities. In addition, the number of variants activities with which the generalized activity can be replaced is represented as vpCardinality. If vpCardinality is 1, the different business processes can be generated according to selection of variants activities. If vpCardinality is more than 1, the selected variants activities can be modeled by three flow patterns; sequence, parallel, conditional flow.

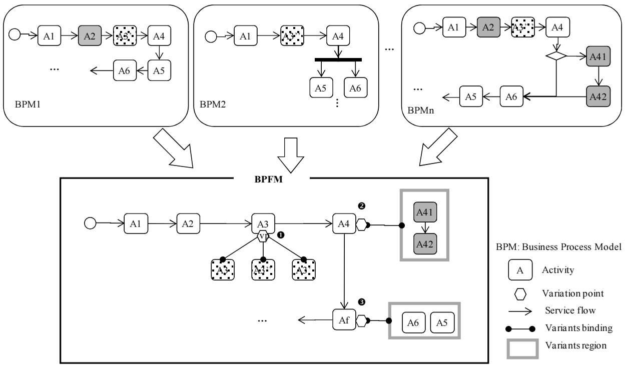


Fig. 13. BPFM with three variation point types.

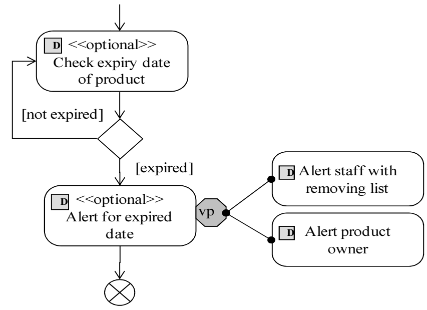
② Extension decision vpType

If a set of domain activities with flow optionally appear in business processes under a specified condition, the domain activities are explicitly represented as variability of extended variants. In ❷ of Fig. 13, the domain activities A41 and A42 are represented in variantsRegion which is attached to the variation point model element. The usage of the extended variants is determined by Boolean decision.

③ Flow decision vpType

If the flows between activities cannot be determined in domain engineering, the flow decision can be delayed until application engineering. As shown in ❸ of Fig. 13, the activities whose flows are not determined are represented in a variantsRegion which is attached to the variation point model element. The various flows between the activities can be generated by the flow decision of developers. The flows may be more complicated than the flows in selection decision vpType. That is, the flow can be a compound flow with sequence, parallel, and conditional flows.

Fig. 14 shows a partial BPFM for the smart-shelf domain. The common CV_properties of the domain activities are represented without stereotypes. Only some of the applications in the smart-shelf domain may have the function of managing expiration dates, so the property "*Check expiration date of product*" is determined as being optional. The BPFM is designed to automatically trigger any alerts that have been incorporated into the business rules. Further, the application may inform its owner or alert the store staff directly when an RFID-tagged product is approaching its expiration date. Thus, the functions are realized as variant activities, and the generalized activity "*Alert about expired date*" is related to its variants. At this time, at least one variant must be realized in a specific application. The stereotype <<1..2>> implies the number of realizable variants in a specific application.



Variation point property			
vp Type	Selection decision		
vp Cardinality	1..2	Variants Flow Pattern	Parallel

Constraints	P_Interface(G_Activity) =
	AND
	R_Interface(G_Activity) =
		P_Interface(V_Activity)
		
		R_Interface(V_Activity)	

P_Interface: Provided Interface G_Activity: Generalized Activity
R_Interface: Required Interface V_Activity: Variant Activity

Fig. 14. an example for selection decision vpType.

III. SUPPORTING ENVIRONMENT

We have developed an RFID_ade (RFID Application Development Environment), which is a supporting tool to support our RFID application development process. The RFID_ade supports the management of the commonalities and variabilities of BPFM and customizes the architectures of individual systems from the BPFM. Moreover, it collects RFID data from the RFID middleware, accesses the data server to retrieve reference data for the RFID data, and processes business logic to implement the RFID-enabled applications. Fig. 15 shows the core components of the RFID_ade and the process for developing RFID-enabled applications.

A. RFID generic activities

Fig. 16 shows the GUI after the developer has opened a BPFM. The RFID-ade consists of four panes: the left pane represents the BPFM structure as a tree, the lower-left pane specifies the property of each activity, the middle-upper pane models the BPFM, and the right pane presents the icons of the RFID generic activities. The RFID generic activities are provided as the common parts of the architecture by the RFID_ade. The variation points of each activity can be set through an activity dialogue box, as shown in Fig. 17; the dialogue box has been implemented depending on the variation points described in Section 2.

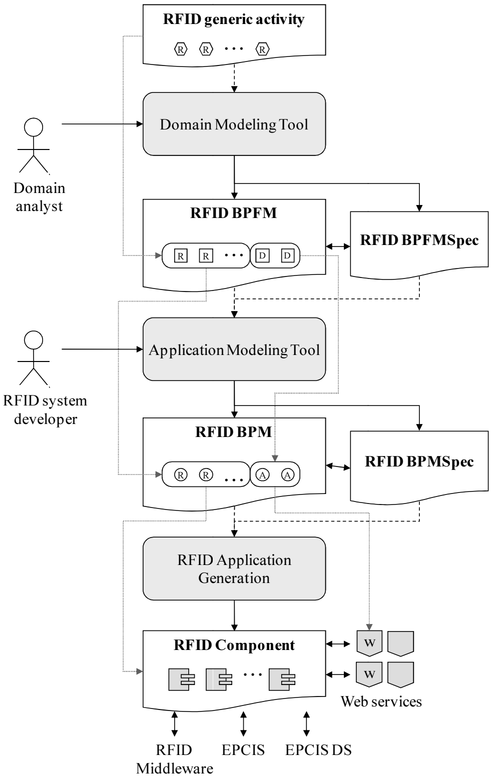
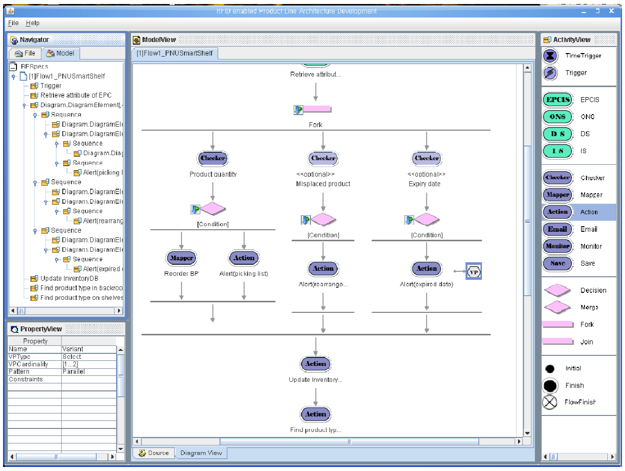


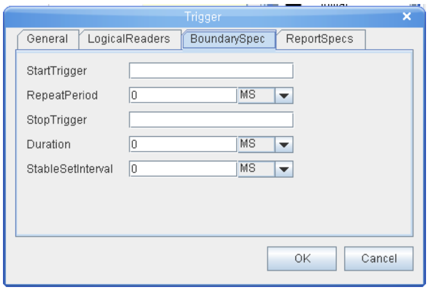
Fig. 15. Supporting environment for RFID-enabled applications.



PropertyView

Property	
Name	Variant
VPTypc	Sclcct
VPCardinality	[1...2]
Pattern	Parallel
Constraints	

Fig. 16. User interface of RFID_ade.



StartTrigger
RepeatPeriod 0 MS
StopTrigger
Duration 0 MS
StableSetInterval 0 MS

Fig. 17. A dialogue box for VPs of RFID generic activity.

B. Modeling of RFID BPFM

In this tool, the RFID domain analyst develops the RFID BPFM using the domain modeling tool, and the RFID BPFM is, in turn, transformed to BPFM specifications (BPFMSpec). As shown in the center pane of Fig. 16, the BPFM is modeled with the activity icons. An icon can be dragged and dropped onto an acceptable part within the model view. The model may be newly generated in the GUI view or may be converted from a BPFMSpec source, which will be shown in another tab on the center pane. The BPFMSpec described in XML is generated automatically and simultaneously when modeling a BPFM. The variation points of the activity flows are described using the activity-property window, as shown in Fig. 18.

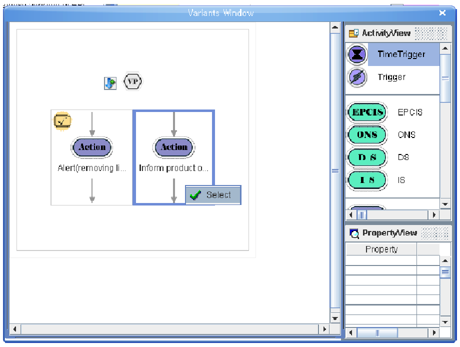
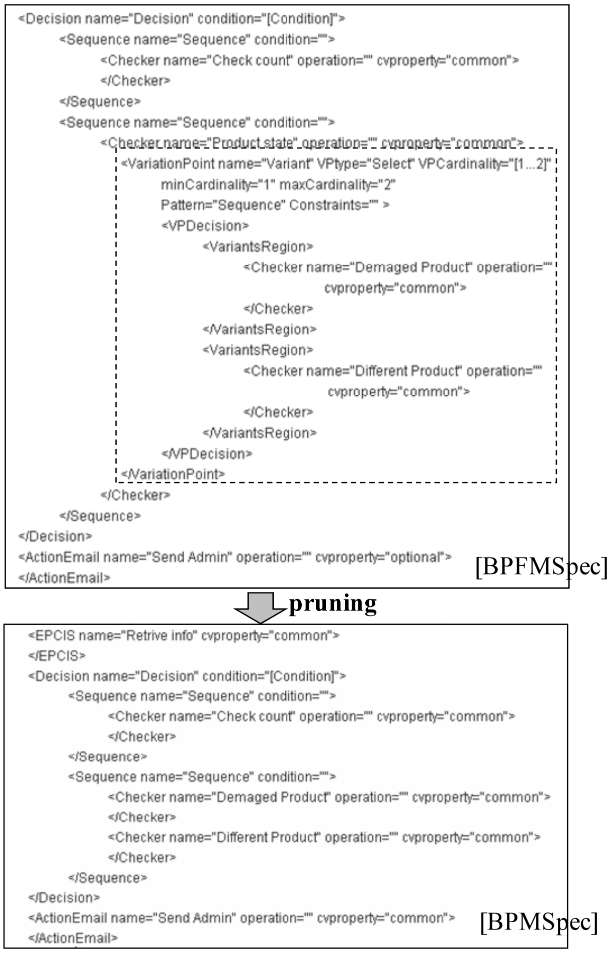


Fig. 18. Activity-property window for VPs of flows.

C. Execution of RFID-enabled Applications

The RFID-ade allows developers to quickly create specialized BPMs for RFID-enabled applications. It contains the functions for deciding and pruning the variabilities of the BPFM. First of all, the variabilities of the optional domain activities can be eliminated by deciding on whether the domain activity should be included in the BPM or not. In addition, the application developers can eliminate variabilities by deciding the variants of the variation points in the BPFM. Then, the RFID-ade automatically prunes the unselected domain activities or variants. In case of the selection decision type, the decision on which variants are selected may be more complicated. In case of a single selection decision, one variant can be selected and the selected variant is automatically included in the BPM. If more than one variant is selected, an error message is displayed. In the case of a multi-selection decision, several variants can be selected according to the value of VPcardinality. If the number of selected variants is more than the VPcardinality scope, an error message is displayed. The selected variants are automatically included in the BPM with the flow pattern —sequence, parallel, or decision flow. The flow pattern decides the flows of the selected variants.

Fig. 19 shows the BPMSpec generated from a BPFMSpec. It can be observed that the specification part for the variation points (dot box in this figure) is eliminated in the BPMSpec. The BPMSpec derived from the decision and pruning process can be executed on the RFID-ade. It parses the BPMSpec, composes the ECSpec, and sends it to the RFID middleware. It receives the ECReport from the RFID middleware, and processes the RFID data according to the business rules described in the BPMSpec. In response to the BPMSpec, the RFID-ade triggers business processes with their corresponding data. The progress of the flow of activities is controlled by the process variable and the transition condition.



```
<Decision name="Decision" condition="[Condition]">
    <Sequence name="Sequence" condition="">
        <Checker name="Check count" operation="" cvproperty="common">
        </Checker>
    </Sequence>
    <Sequence name="Sequence" condition="">
        <Checker name="Product state" operation="" cvproperty="common">
          <VariationPoint name="Variant" VPtype="Select" VPCardinality="[1...2]"
              minCardinality="1" maxCardinality="2"
              Pattern="Sequence" Constraints="" >
              <VPDecision>
                  <VariantsRegion>
                      <Checker name="Demaged Product" operation=""
                          cvproperty="common">
                      </Checker>
                  </VariantsRegion>
                  <VariantsRegion>
                      <Checker name="Different Product" operation=""
                          cvproperty="common">
                      </Checker>
                  </VariantsRegion>
              </VPDecision>
          </VariationPoint>
        </Checker>
    </Sequence>
</Decision>
<ActionEmail name="Send Admin" operation="" cvproperty="optional">
</ActionEmail>
```

[BPFMSpec]

pruning

```
<EPCIS name="Retrive info" cvproperty="common">
</EPCIS>
<Decision name="Decision" condition="[Condition]">
    <Sequence name="Sequence" condition="">
        <Checker name="Check count" operation="" cvproperty="common">
        </Checker>
    </Sequence>
    <Sequence name="Sequence" condition="">
        <Checker name="Demaged Product" operation="" cvproperty="common">
        </Checker>
        <Checker name="Different Product" operation="" cvproperty="common">
        </Checker>
    </Sequence>
</Decision>
<ActionEmail name="Send Admin" operation="" cvproperty="common">
</ActionEmail>
```

[BPMSpec]

Fig. 19. BPMSpec driven by deciding and pruning variabilities.

IV. RELATED WORK

Methodology for supporting the development of RFID-enabled applications: RFID has recently gained enormous attention across various sectors of industry, the media, and academic research. However, management research and academic management literature on the use of RFID is still scarce. The publications that cover RFID applications focus mainly on case studies and discussions of business opportunities. Vendors such as Sun Microsystems [1], IBM [2], Oracle [3], and Microsoft [4], have been extending their application development and middleware technology stacks to handle RFID. In order to realize the time and cost benefits associated with RFID, certain business processes should be changed. Existing business processes should adapt to using the data gained from RFID tags. To the best of our knowledge, few existing approaches focus on RFID technology integration of development methodologies, i.e., focus on how RFID-enabled applications are organized and developed.

Applying product-line approach to the development of applications: [17] proposed techniques for defining frameworks within the WfMS (workflow management system) domain that offer guidelines for the architecture design. These techniques involve the concept of a model framework from Catalysis as a base to generate components. [18] defined a component-based product line for WfMS; members of the WfMS family can be generated from the proposed product line by providing their specific requirements. [19] summarized 15 cases of applying the software product-line engineering paradigm in industry; they reported on examples of cost reduction, shorter development times, and quality improvement. As far as we know there are no previous works that take a similar approach in this domain of RFID-enabled applications.

V. CONCLUSIONS

RFID technology can be used to significantly improve the efficiency of business processes by providing the capabilities of automatic identification and data capture. Obviously, RFID technology and applications based on product lines are still developing. This paper has proposed an approach for developing RFID-enabled applications based on BPFMs in the RFID domain. We first identified the common activities in the RFID domain and analyzed their variabilities. Then, we suggested a modeling approach that explicitly represents these variabilities in the BPFM. Using this BPFM, we have developed several RFID-enabled applications. The results show that RFID-enabled applications need not carry the additional burden of RFID data processing, thereby substantially reducing the cost of developing and managing RFID applications. Our future research activities will include an extension to the context-aware RFID BPFM, which will enable processing of other types of sensors such as temperature, humidity, shock, and location sensors.

ACKNOWLEDGMENT

This work was supported by Dongseo University “Dongseo Frontier Project” Research Fund of 2010.

REFERENCES

[1] Sun Microsystems, http://www.sun.com/software/solutions/rfid/

[2] IBM, http://www306.ibm.com/software/pervasive/w_rfid_premises_server/, December 2004.

[3] Oracle, http://www.oracle.com/technology/products/iaswe/edge_server

[4] Microsoft, http://www.microsoft.com/business/insights/about/aboutus.aspx

[5] Research Center Logistics Information Technology, http://lit.pusan.ac.kr

[6] M. Moon, Y. Kim, and K. Yeom, “Contextual Events Framework in RFID System”, In proceedings of third International Conference on Information Technology (IEEE Computer Society) pp. 586-587, 2006.

[7] Weiss, D.M., Lai, C.T.R., *Software Product-Line Engineering: A Family Based Software Development Process*, Addison-Wesley, ISBN 0-201-694387, 1999.

[8] Process Family Engineering in Service-Oriented Applications, URL: http://www.pesoa.de/index-en.html

[9] K. Traub, S. Bent, T. Osinski, S. N. Peretz, S. Rehling, S. Rosenthal, B. Tracey, “The Application Level Events (ALE) Specification, Version 1.0,” EPCglobal Proposed Standard, February 2005.

[10] EPCglobal Inc., URL: http://www.epcglobalinc.org

[11] EPCglobal, “EPC Information Services (EPCIS) Version 1.0 Specification,” EPCglobal Working Draft, June 2005.

[12] M. Mealling, “EPCglobal Object Name Service (ONS) 1.0,” EPCglobal Working Draft, December 2004.

[13] IBM Corporation and Microsoft Corporation, “Security in a Web Services World: A proposed Architecture and Roadmap”, white paper, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwssecur/html/securitywhitepaper.asp, April 2002.

[14] Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/ws, March 2001.

[15] N. Russell, W.M. van der Aalst, A. ter Hofstede, and P. Wohed, “On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling “, In Third Asia-Pacific Conference on Conceptual Modeling (APCCM2006), vol. 53 of CRPIT, pp. 95–104, Australia, 2006.

[16] 7. H. Gomma, *Designing Software Product Lines with UML, From Use Cases to Pattern-Based Software Architectures*, Addison-Wesley, 2004.

[17] I.M.S. Gimenes, L. Barroca, “Enterprise Frameworks for Workflow Management Systems”, Software Practice & Experience, No. 32, pp.755-769, 2002.

[18] I. M. de Souza Gimenes, F. R. Lazilha, E. A. de Oliveira Junior, and L. Barroca, “A Component-based product line for Workflow Management Systems”, CLEI Electronic Journal, Vol. 7 No. 2 paper 5, 2004.

[19] Pohl, K., Böckle, G., van der Linden, F., *Software Product Line Engineering - Foundations, Principles, and Techniques*, Heidelberg Springer-Verlag, 2005.

Mikyeong Moon is an associate professor of computer and information engineering at the Dongseo University in Busan, Korea. She received the BS degree in 1990 and the MS degree in 1992, both in computer science from the Ewha Womans University, Seoul, Korea, and the PhD degree in computer science and engineering from Pusan National University in 2005. Her current research interests include software reuse, product line engineering, requirement engineering, and RFID solutions.