Transformation from eEPC to S-BPM: A Case Study

Başak Çakar[1] and Onur Demirörs[2]

[1] ASELSAN Inc., SST-MD-YMM
P.K. 1, 06172 Yenimahalle /Ankara, Turkey
bcakar@aselsan.com.tr
[2] Informatics Institute, Middle East Technical University,
06531 Ankara, Turkey
demirors@metu.edu.tr

Abstract. Business process models are vital assets of any organization. The organizations prefer to use one of the modeling methods and notations according to its features like tool support, size of user base, ease of use. On the other hand, prevalent methods (formalisms) to model these processes change over time. Furthermore, some modeling tools and/or methods might be pulled out of the market completely. In order to handle these kinds of changes easily, migration methods has to be defined. In this work, model transformation is proposed as a method to migrate from eEPC to S-BPM. Direct mapping rules are defined to transform models and the application of these rules is demonstrated by on a real world case study.

Keywords: Process Modeling, eEPC, S-BPM, Model Transformation.

1 Introduction

Business process models describe logical order of activities and dependencies [1]. There are many process modeling languages and organizations use them to reach a comprehensive understanding of organizational processes and analyze them. In the frame of this study, we particularly focus on eEPC and S-BPM languages.

EPC is a business process modeling technique developed by Scheer et al. at the Institute for Information Systems in Germany, in 1990s [2][3]. EPC represents business process as an ordered graph which shows chronological sequence and logical interdependencies between elements. In order to model more complex business processes, EPC notation is extended with additional elements from the organization and data view, which is called eEPC (extended Event-driven Process Chain). It is relatively simple notation to model business processes and highly accepted by the practitioners from diverse areas for business process re-engineering, management and documentation.

S-BPM is a paradigm that is developed by Albert Fleischmann [4] to describe and execute business processes from the perspective of subjects. S-BPM gets inspired from natural languages and the structure of S-BPM is similar to sentence structure of natural languages. According to S-BPM, subjects are active elements in a business process. Therefore, they should be the starting point of the activities (like natural language sentences) [5]. Subjects execute business processes by exchanging messages with each others. Interactions between subjects are shown in the Subject Interaction Diagram (SID). SIDs visualize subjects and data flow (exchange messages) among them. Internal activities of subjects are shown in Subject Behavior Diagram (SBD). S-BPM uses top down approach in determining communication between subjects and uses bottom-up approach in determining internal behavior of subjects.

In the industry EPC is widely accepted during the last decade by means of ARIS toolset. A number of organizations represented their processes using eEPC. However; there is a gap between business and information technology systems in the eEPC since it focuses on notations and this makes difficult to design business processes within IT systems [6]. In order to close that gap, S-BPM has emerged as a new modeling paradigm. S-BPM enables to create dynamic business applications and to integrate them into the existing systems seamlessly. This also provides organizations, which use S-BPM as modeling language, competitive advantage. In addition to this, S-BPM provides a better representation for human interaction patterns and its notation is simple and easy to understand. As an alternative to EPC, S-BPM gaining ground with IT support. Migration of legacy eEPC models requires considerable effort. Furthermore, it's a labor intensive work which increases the usage of personal resources and costs dramatically.

Our goal is to facilitate the transformation process by defining rules with direct mapping. Model transformation is adopted as a main method to provide automation. A model transformation takes a source model and transforms it into a target model by using predefined transformation definition (rules) [7]. The both of the models conform to their respective metamodels. A transformation is defined with respect to the metamodels. The transformation definition is executed on concrete models by a transformation engine. In our case, eEPC is the source model and S-BPM is the target model. Transformation definition is explained in details throughout this document. In order to automate eEPC to S-BPM transformation we developed an extension to existing transformation engine namely UPROM (Unified Process Modeling Tool) developed by Bilgi Grubu and SMRG Research Group. UPROM is based on bflow* toolbox which is an open source modeling tool contributed by at the University of Hamburg and the University of Applied Sciences Emden/Leer [8].

Before the conversion, syntax of the eEPC model is needed to be checked. eEPC syntactical validation rules are stated in Section 3 according to formal definitions given in [9] [10]. The rules to check the validity of generated S-BPM model is also defined in Section 4. These rules include constraints on inter-elements relations and element sequence. In this study we check the validation of input and output models manually. However; after the development of the tool is completed, it is done semi-automatically.

In this paper, we define a set of rules (guidelines) to convert eEPC models to S-BPM models and we validate these by applying our transformation method to a case study. The remainder of this paper is organized as follows: Related work is mentioned in Section 2. Notation of eEPC, its validation rules and metamodel from scratch are presented in Section 3. Notation of S-BPM with validation rules and

S-BPM metamodel from scratch can be found in Section 4. Section 5 presents transformation methodology and rules. Case study can be found in Section 6. Discussions and conclusion are presented in Section 7. Finally references can be found in Section 8.

2 Related Work

In the literature, there are many research studies on EPC and eEPC that are useful for model transformation. W.M.P. van der Aalst gives a formal definition which explains the requirements of an EPC element and defines the core elements as well [9]. Kees van Hee et al. define extended-EPC (eEPC) by providing syntax and semantics [10]. Those studies provide a foundation for our transformation and validation rules.

Transformation techniques to generate models based on existing eEPC models are defined for different formalisms like BPMN [10] and UML [11]. Nüttgens et al. [11] transform EPC models into object oriented models. Relations between EPC and UML diagrams (use case, activity diagram, class diagram and application structure diagram) and the details about which information in an EPC element is used to transform related diagram are defined. In this study, mapping between EPC and UML elements is not stated; only structural transformation approach is explained.

In [12], Tscheschner describes a direct mapping technique to convert eEPC to BPMN and defines transformation rules to map eEPC elements to BPMN elements. On the other hand, eEPC and BPMN differ in their semantics and formalization. Therefore, a complete mapping (structural and semantic) is almost not achievable by solely using direct mapping for each and every component. In order to get complete one, elements of core EPC definition and a subset of eEPC elements are used for mapping.

In [13], Aguilar-Savén compares different modeling languages in terms of message exchange, communication partner's role, process flow and timing, visualization of none sequential process steps, understandability and clear structure of models in order to find the most suitable language for a specific project. According to this study S-BPM is very successful in visualizing exchange messages between subjects. Behavior of the communication partners is also well defined in S-BPM and it has a comprehensive notation. On the other hand, eEPC is stronger than S-BPM in terms of showing the details of process flow and ease of understanding.

In [14] Sneed provides a mapping method for bidirectional transformation between BPMN and S-BPM. Transformation consists of two main parts. In the first part; rules for atomic structures (basic modeling constructs) are defined. The first part of the transformation provides mapping for Subject Behavior Diagrams. In the second part, mapping rules for complex structures are defined. Complex structures are used to visualize the communication view between subjects. In contrast to our study, this study focuses on SID generation.

3 eEPC Notation

The subset of eEPC elements covered by our transformation rules are listed in Figure 1. In order to elaborate elements and their relationships, composed meta-model is depicted in Figure 2 (it's also used for model transformation). It is based on the formal definition of EPC defined by W.M.P. van der Aalst [9] and eEPC Kees van Hee et al. [10].

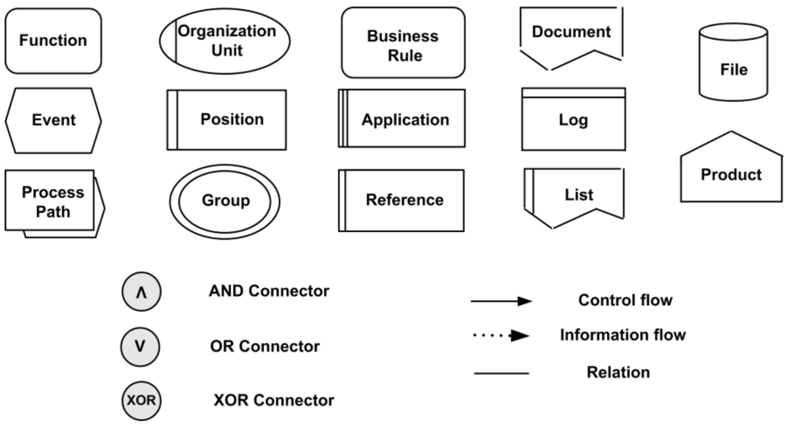


Fig. 1. Covered set of eEPC elements

According to our meta-model, a process consists of at least five process elements (start event, function, end event and control flows between them). Process elements can be workflow elements (function, event, process path, control flow, split connector and join connector) or extended elements (data object, resource object, actor, information flow and relation). eEPC workflow elements are consecutive to each other to form a process flow. Core elements (function, event and process path) are connected to each other by control flows. Data objects (document, list, log, product and file) are connected to functions or process paths via an information flow and they are connected to an actor via a relation. Relation also connects functions and process paths to actors and resource objects (application, reference and business rule).

In order to validate an eEPC diagram the following rules are used [9] [10]:

- There must be at least one start event,
- There must be at least one end event,
- All elements must be connected,
- All functions or process paths must have exactly one incoming and one outgoing control flow,
- Events cannot be consecutive to each other,
- Split connectors must have one incoming control flow and more than one outgoing control flow,
- Join connectors must have more than one incoming control flow and one outgoing control flow.

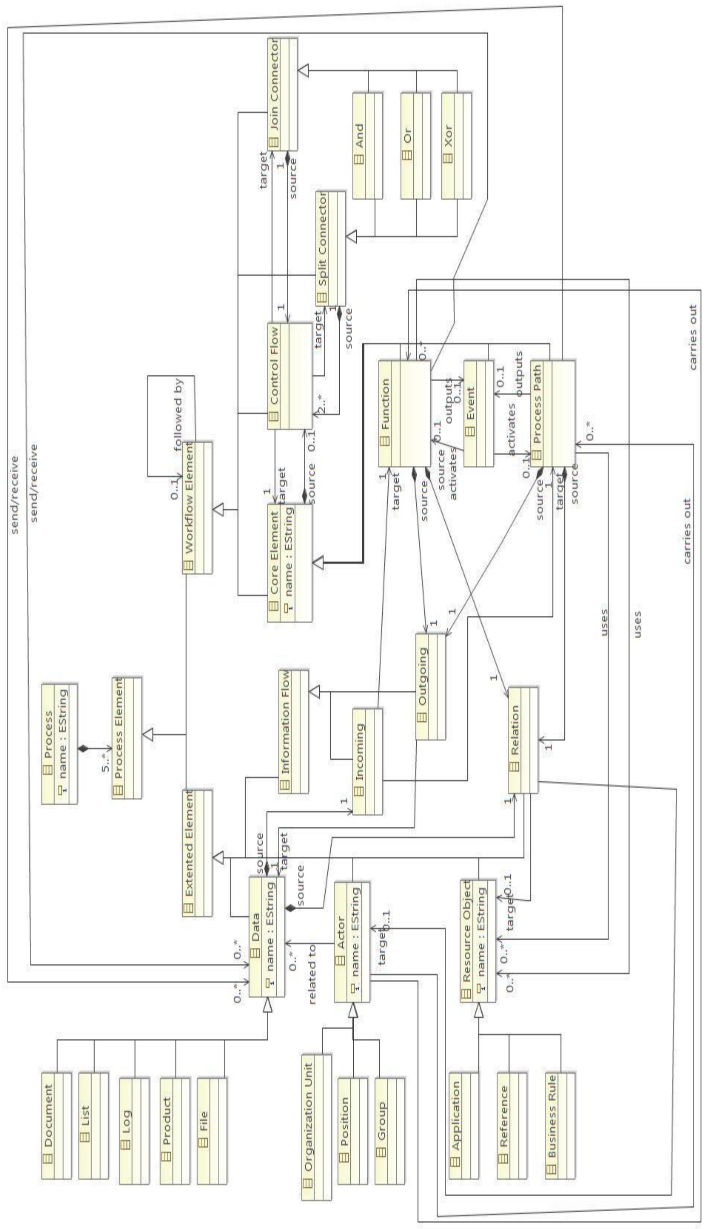


Fig. 2. eEPC Metamodel from scratch

4 S-BPM Notation

Since "eEPC to S-BPM" transformation is an injective non-surjective function, all S-BPM elements are not covered in our mapping. The covered elements of S-BPM elements are listed in Figure 3.

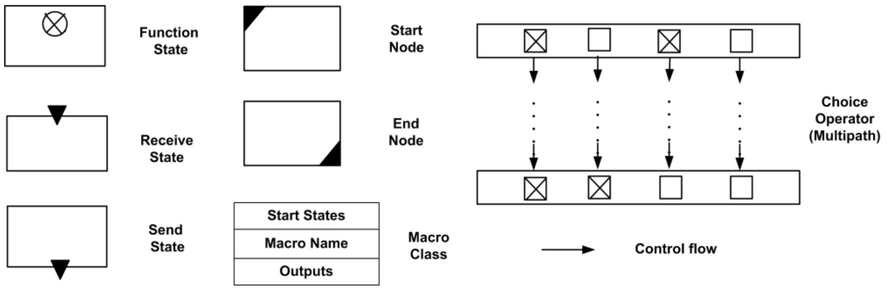


Fig. 3. Covered set of S-BPM elements

SBD elements and their relations are shown in Figure 4 as a meta-model. SBD is composed of at least 3 process elements; start subject state, end subject state and an arc that connected these two states. SBDs consist of subject states (Receive, Send and Function) and outgoing control flows (arcs). A control flow has a label whose value is determined according to its source subject state. "Receive" state depicts the receiving message action and it is followed by a control flow. That flow shows information of sender and received data (Receive Arc). "Send" state is used to show sending message action. It is followed by a control flow (Send Arc) that shows receiver info and sent data. "Function" state with an outgoing arc (State Arc) is used for tasks and post-condition of the function. In addition to those SBD elements, macro classes and choice operators ("Multipath" structure) are also used. Macro classes are used to show sub-processes that repeat in different SBDs in order to avoid repeated patterns. The notation of macro class consists of three parts, in the first part valid start states which activate the sub-process are shown. Name of the macro (sub-process) is shown in the second part. In the final part, the outputs of the sub-process are shown. Choice operator consists of a number of parallel paths which are activated simultaneously. It starts and ends with a bar which includes beginning and end switches for each path.
Validation rules for SBDs:

- There must be at least one start subject state,
- There must be at least one end subject state,
- All elements must be connected,
- All receive nodes must be followed by a receive arc which is annotated by a receive clause,
- All send nodes must be followed by a send arc which is annotated by a send clause,
- All macro classes must have at least one incoming control flow (receive, send or state arc) and at least one outgoing state arc,

- All multipath structures must have exactly one incoming control flow and exactly one outgoing state arc,
- Start bar and end bar of a multipath structure must have the same number of switches.
- All alternate paths in a multipath structure must be start and end with a switch.

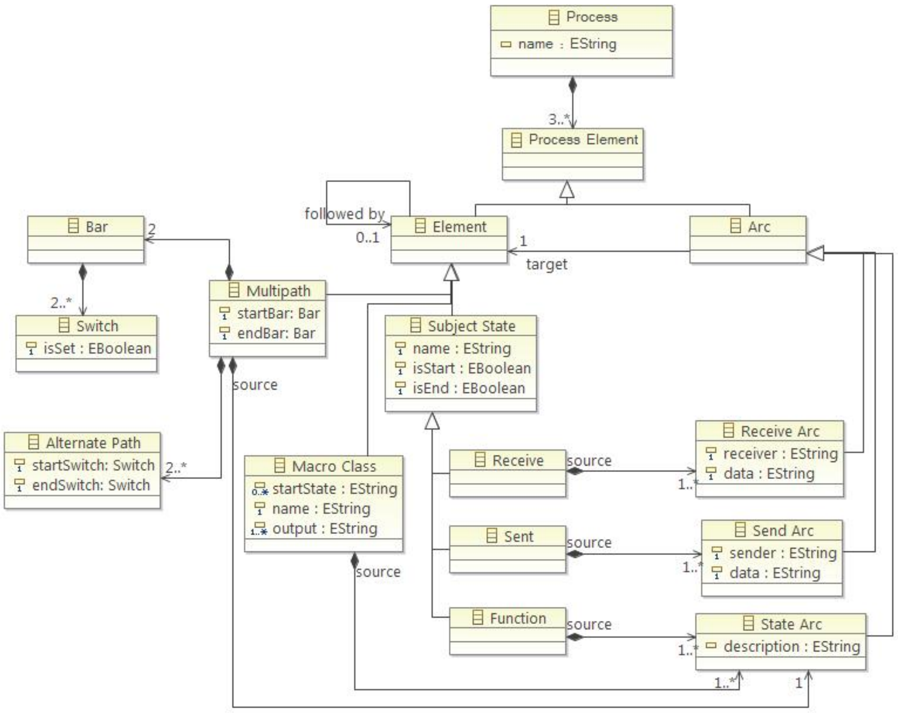


Fig. 4. S-BPM Metamodel from scratch

5 Transformation

eEPC and S-BPM differs the most in the aspect of adopted modeling techniques. eEPC uses flow-oriented approach. Due to that, it is generally considered as a kind of flowchart. It visualizes the sequence of tasks which are performed by different actors. On the other hand, S-BPM uses subject-oriented modeling technique which means that it focuses on subjects (actors) and their relationships. In eEPC, a business process performed by more than one actor can be visualized by one eEPC diagram. However, in S-BPM that business process is visualized by SID in higher level and internal activities of each subject are shown in separate SBDs in lower level. For that reason, our conversion strategy consists of two phases. In the first phase, eEPC model needs to be decomposed into individual eEPC models for each actor. Then, as a second phase, SBDs are generated for each individual eEPC model. Input eEPC diagrams which will be transformed are assumed to be valid according to syntactic and semantic rules defined in [9] [10].

Structural mapping is used as transformation technique; nevertheless there are significant differences between eEPC and S-BPM notations. For transformation, patterns which composed of different elements and transformation rules for those patterns are defined in the following.

5.1 Phase One: Generating Individual eEPCs

As we stated in the previous section, eEPC to eEPC transformation is initially performed in case of more than one actor are responsible for the business process. The tasks of different actors are transformed into individual eEPC diagrams. First, the actors who are only connected to functions are determined and then eEPC diagrams for each of them are generated.

In the individual eEPC generation process, function or function set which are connected to different actors are ignored. In order to generate continuous flow for the actor, the output event of the last function performed by concerning actor and the output event of the last function performed by a different actor are concatenated with an "and” connector(Figure 5).

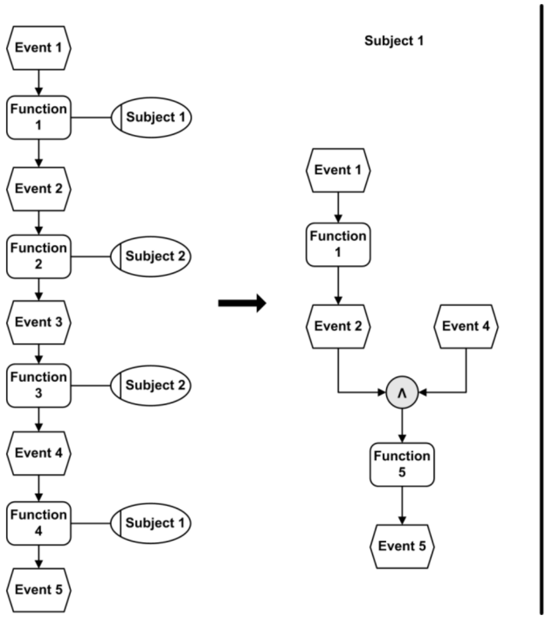


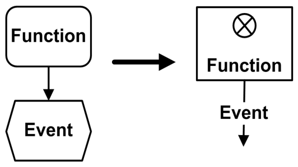
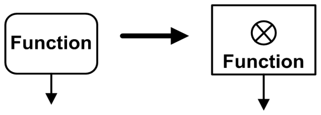
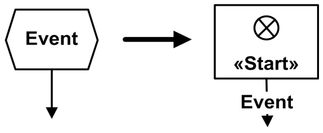
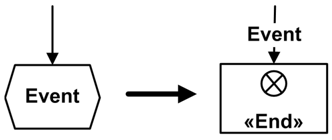
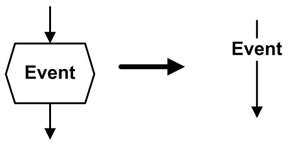
Fig. 5. eEPC to eEPC Transformation

5.2 Phase Two: Generating S-BPM Models

SBD generation starts from the root node and follows through the nodes in sequential order. The events without incoming control flow, the events without outgoing control flow and function-event pairs are taken into account firstly. In the conversion of function-event pairs, relations of the function with other elements (data and resource object) are inspected. Matched patterns (defined in following subsections) converted into respecting target patterns.

Functions and Events. Functions and events are the most crucial elements of eEPC. Functions represent tasks or activities which are executed by organization units, groups or positions. Events show the state of the process. They are triggered by functions and they also trigger functions as well. Function-event pairs show the flow of the business process. Events are problematic in transformation because there is not a corresponding element in S-BPM to map. In order to avoid information lost, following rules are defined in Table 1.

Table 1. Transformation rules for functions and events

Functions followed by an event	
	Functions are directly mapped to performing action (function state) element of S-BPM. Events are shown as text on the outgoing control flow.
Functions without following event	
	Functions without following event are mapped to performing action element and a control flow without any label.
Events without incoming control flow	
	Events without incoming control flow are interpreted as start event. It's mapped to a dummy start function with **«Start»** annotation
Events without outgoing control flow	
	Events without outgoing control flow are interpreted as final event. It's mapped to a dummy end function with **«End»** annotation
Events with incoming and outgoing control flows	
	Events with incoming and outgoing control flows are mapped to a control flow with a label that includes event description.

Data and Subjects. In the eEPC, there are many different types of data and subjects, however the corresponding representations of those objects in S-BPM is not available. Transforming these elements by ignoring their type causes information lost. In order to overcome that, annotations for data and subject types are introduced. Table 2 gives the subject type annotations and Table 3 gives annotations for information, material and resource objects.

Table 2. Subjects and their annotations

Subject Type	Annotation
Organization Unit	«unit»
Group	«group»
Position	«position»

Table 3. Information, Material and Resource Objects and their annotations

Information, Material and Resource Objects	Annotation
Document	«doc »
List	«list »
Product	«product»
File	«file»
Log	«log »
Application	«app»
Reference	«ref »
Business rule	«rule»

Document, list, product, file and log can be seen as a data object. However; application, reference and business rule are resource objects. Applications are systems and supports functions for execution. References (laws, regulations, standards, guidelines, etc…) are used to provide information to execute related function. Business rules restrict the operations of functions. For resource objects, notation with "Used" keyword and respective annotation («app», «ref» or «rule») is used. The description of resource object is also given as a part of this notation. Resource object notation is connected to functions with a dotted line (Refer to Figure 6).

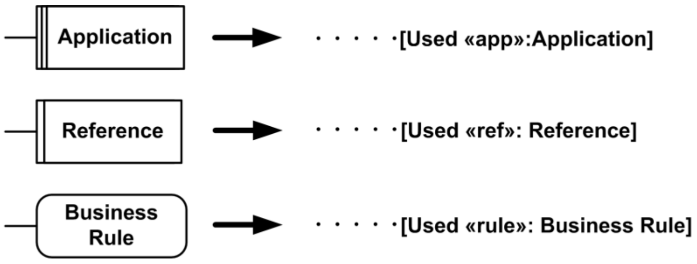
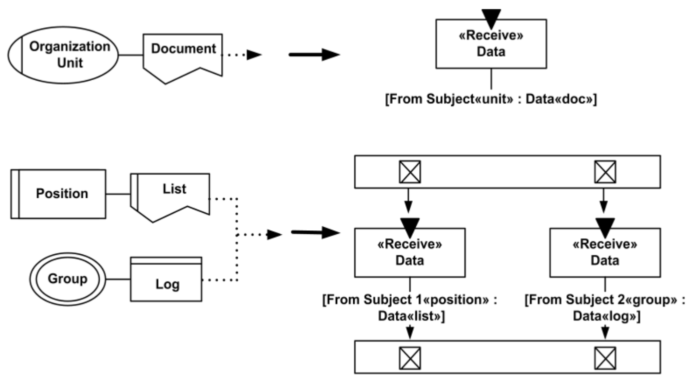
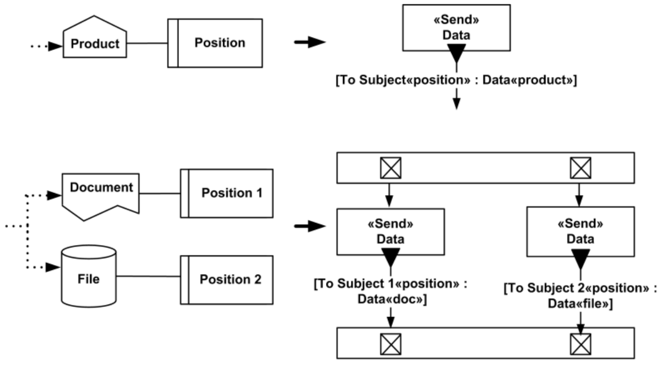
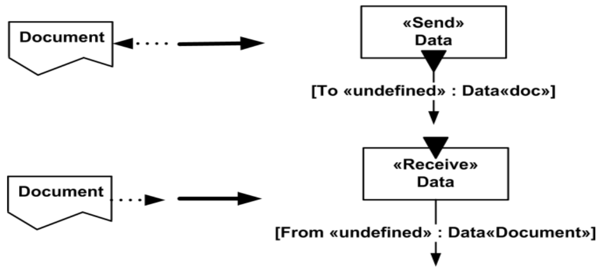

Application
·····[Used «app»:Application]
Reference
·····[Used «ref»: Reference]
Business Rule
·····[Used «rule»: Business Rule]

Fig. 6. Resource object transformation rules

Receive/Send Data. In the eEPC, there are data objects which are used by functions (input) or produced by functions (output). Direction of the information flow between data object and function shows whether it is a receiving or sending data. Data objects do not affect the process flow. In SBD, data objects are stated on control flow as a label. The value of that label is set according to direction of the flow (sent or received data). "Receiving Message" and "Sending Message" elements are used during the conversion of data-subject pairs. Since the data objects are not ordered in the process flow, it's assumed that all receiving messages take place before the concerning function and all sending messages come after the function in the flow. The order of received and sent messages can be stated by the user before the transformation. In the following patterns for sending and receiving messages and their transformation rules with semantics are explained in Table 4.

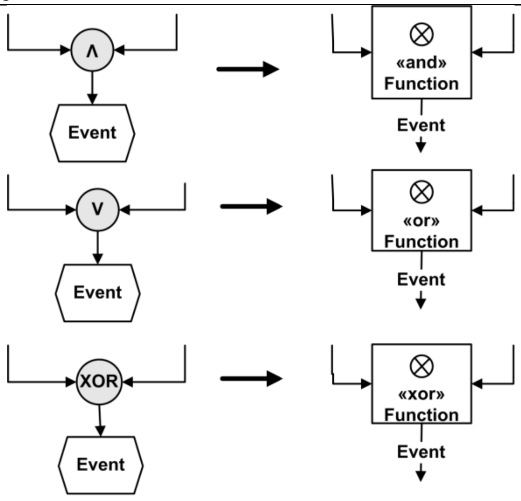
Table 4. Transformation rules for data objects and subjects

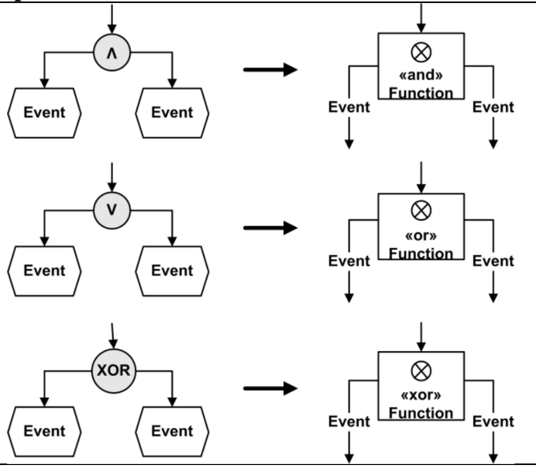
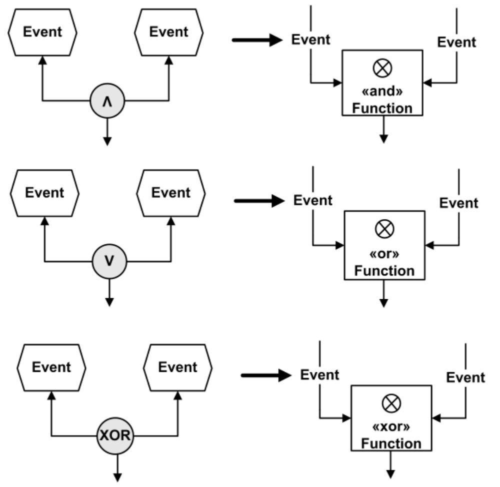
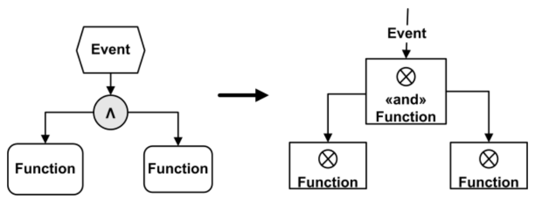
Data with outgoing information flow	
 Organization Unit Document «Receive» Data [From Subject«unit» : Data«doc»] Position List Group Log «Receive» Data «Receive» Data [From Subject 1«position» : Data«list»] [From Subject 2«group» : Data«log»]	Name of the element is given as **"«Receive» data name"** automatically. If there is more than one data object received by a function, choice operator to combine receiving messages is used.
Data with incoming information flow	
 Product Position «Send» Data [To Subject«position» : Data«product»] Document Position 1 File Position 2 «Send» Data «Send» Data [To Subject 1«position» : Data«doc»] [To Subject 2«position» : Data«file»]	Name of the element is given as **"«Send» data name"** automatically. If there is more than one data object sent by a function, choice operator to combine sending messages is used.

Data without any subject relation	
 Document «Send» Data [To «undefined» : Data«doc»] «Receive» Data Document [From «undefined» : Data«Document»]	**«undefined»** annotation is used for missing subjects.

Logical Connectors. Logical connectors show the logical relationships between functions and events in the control flow. They are used to split one control flow into two or more control flows or to concatenate two or more control flows. In the eEPC there are three types of logical connectors; "and", "or" and "exclusive-or". Since there is not any element in S-BPM with the same behavior to map logical connectors, conversion of them is the most problematic part of the transformation process. Three new functions with "«and»", "«or»" and "«xor»" annotations are defined in order to use as logical connectors. In S-BPM functions with more than one incoming control flows and more than one outgoing control flows are possible. Therefore, "Performing Action" element with a respective annotation is used to depict logical connectors. In Table 5, logical connectors grouped by behavior are mapped to S-BPM elements with brief descriptions.

Table 5. Transformationrules for logical connectors

Join Functions	
 Λ Event «and» Function Event V Event «or» Function Event XOR Event «xor» Function Event	Join Functions connectors joins the controls flows that ends with functions and activates the result event. Join Functions is mapped to performing action elements with «and», «or» or «xor» annotations according to logical connectors. These elements have incoming control flows without any label and an outgoing control flow with a label that includes the description of the output event of join function.

Split Function	
 ∧ Event Event «and» Function Event Event V Event Event «or» Function Event Event XOR Event Event «xor» Function Event Event	Split Function connectors activate two or more post-conditions (events) of a function. Split Function is mapped to performing action element with «and», «or» or «xor» annotations. These elements have one incoming control flow and outgoing control flows with a label that includes event description.
Join Events	
 Event Event ∧ Event Event «and» Function Event Event V Event Event «or» Function Event Event XOR Event Event «xor» Function	Join Events connectors concatenate preconditions (events) and activate the following function. Join Events is mapped to performing action element with «and», «or» or «xor» annotations. These elements have incoming control flows with a label that includes the description of precondition (event) and an outgoing control flow without any label.
Split Event	
 Event ∧ Function Function Event «and» Function Function Function	Split Event connector activates two or more functions when the precondition (event) is satisfied. For Split Event, performing action element with «and» annotation is used. It has an incoming control flow with a label that includes the description of precondition (event) and outgoing control flows without any label.

In order to transform of "and" block ("and" connectors using in pairs), choice operator is used. Choice operator leaves the execution sequence of the activities to the discretion of the respective person, can be preferred for transformation. However, usage of choice operator is directly related to structure of eEPC model. In eEPC logical connectors can be used singly or in pairs. If there is only one connector is used for parallel processes in eEPC, there is no way to determine the end of those processes. Therefore, predefined mapping rule is applied. If "and" connectors are used in pairs (logical elements match and parallel paths are opened and closed with the same element), they are transformed into choice operator (Figure 7). In most situations, this goes well with the intended semantics of "and" in EPC process models when the following functions are to be executed by the same person. However, in order to identify the behavioral difference in "and" connectors, modelers have to define whether logical connector starts/ends an alternative path block.

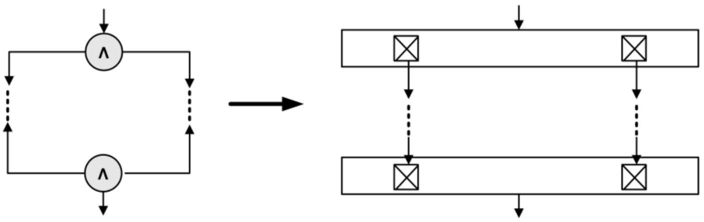


Fig. 7. Mapping rule for "And" connector block

Process Path. Process path navigates the control flow to sub-processes. Transformation rules, defined for function element, are also applicable to process path element. In contrast to function element, process path element is mapped to macro class. In the notation of macro class element, invalid start states, name of the macro and possible output transitions are shown respectively. In transformation, firstly the expansion of the process is sought in the project folder. If the model of the sub-process is in the project folder, start states of sub-process are used as valid start states of macro class and end states of sub-process are used as possible output transitions (Figure 8). If the model of the sub-process is not in the project folder, user can define start states and output transitions manually, or «undefined» annotation is used for missing information.

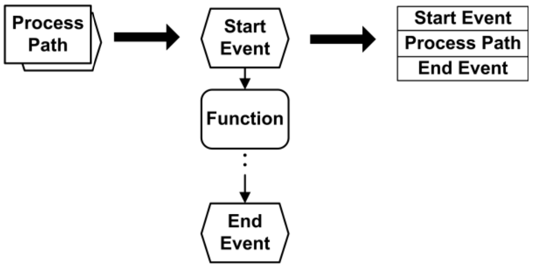


Fig. 8. Transformation rule of Process Path element

6 Case Study

eEPC models generated by Bilgi Grubu in UPROM within a project, which is initiated by Ministry of Development of the Republic of Turkey, are transformed to S-BPM. The aim of the project is to define business processes and to support those processes with IT Systems. Within the scope of the project, main and supporting business processes which belong to development agencies are modeled [15]. Two essential processes of Archive Management System are selected for case study. Figure 9 shows activity and data (sent/received) flow of archiving process. In Figure 10, eEPC model of the Acquiring archive material process is given.

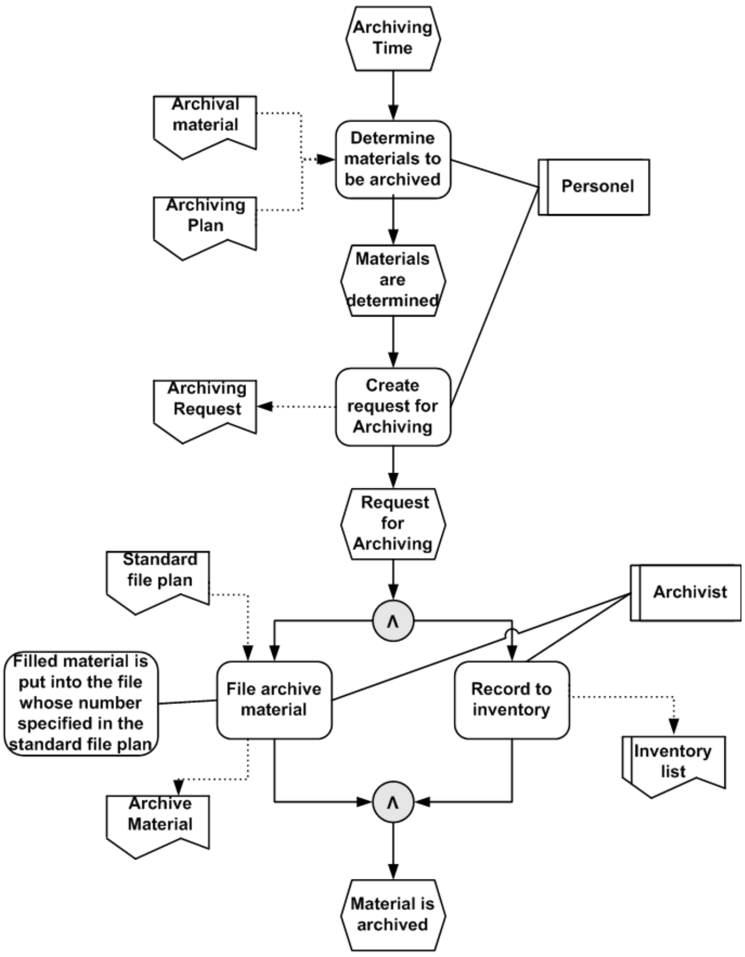


Fig. 9. Archiving Process

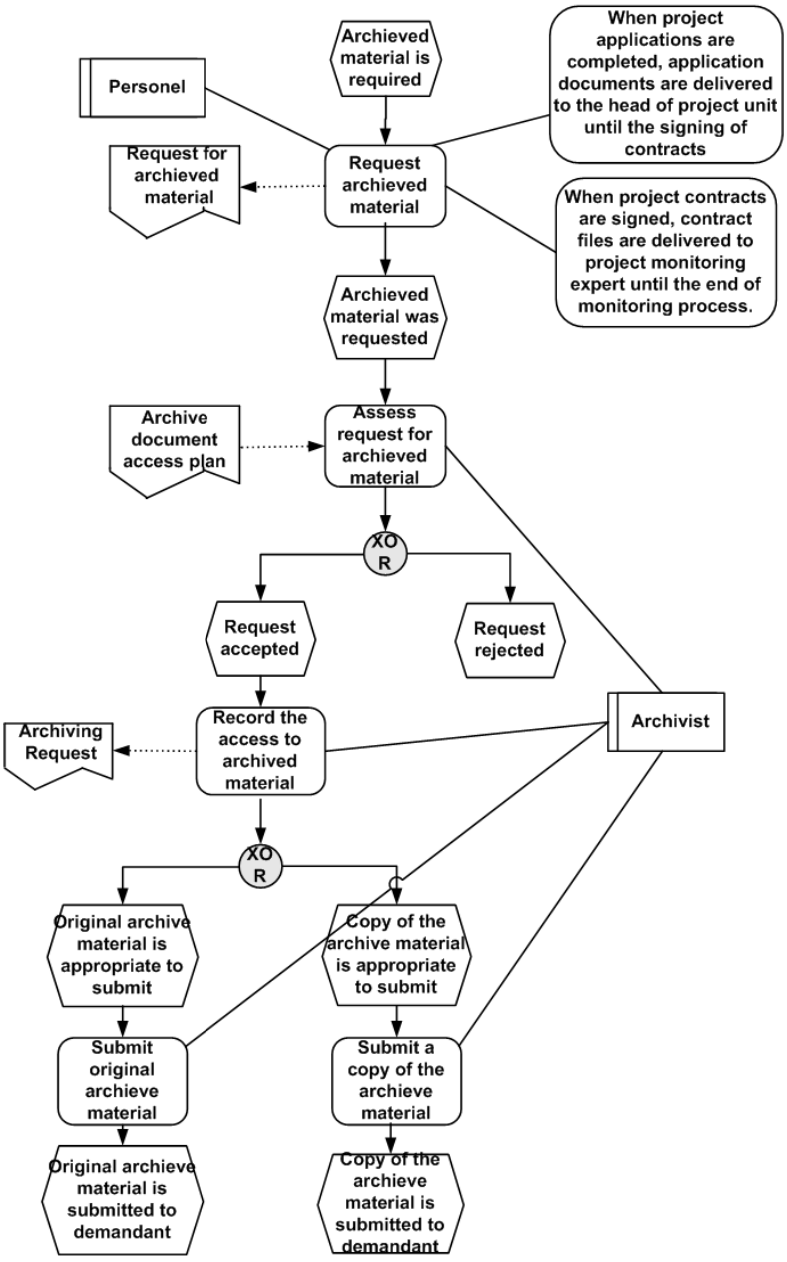

Archieved material is required
Personel
When project applications are completed, application documents are delivered to the head of project unit until the signing of contracts
Request for archieved material
Request archieved material
When project contracts are signed, contract files are delivered to project monitoring expert until the end of monitoring process.
Archieved material was requested
Archive document access plan
Assess request for archieved material
XOR
Request accepted
Request rejected
Archiving Request
Record the access to archived material
Archivist
XOR
Original archive material is appropriate to submit
Copy of the archive material is appropriate to submit
Submit original archieve material
Submit a copy of the archieve material
Original archieve material is submitted to demandant
Copy of the archieve material is submitted to demandant

Fig. 10. Acquiring Archive Material Process

In both of them, two actors Personnel and Archivist takes a role to accomplish the process. Thus in the first part of the transformation, process are divided into two sub-processes by omitting actors which are directly connected to functions from the model. Figure 11.a-12.a shows the sub-processes of Archiving process and Figure 13.a-14.a shows the sub-processes of Acquiring archive material process.

In the second part of the transformation SBD for each sub-process are generated by using the rules defined in section 5.2. Thus, only the name of the applied rule is given in the explanations of transformations. While transforming Personnel_Archiving Process, the rules “Events without incoming control flow”, “Data with outgoing information flow”, “Functions followed by an event”, “Data with outgoing information flow” and “Events without outgoing control flow” are used respectively.(Figure 11.a-11.b).

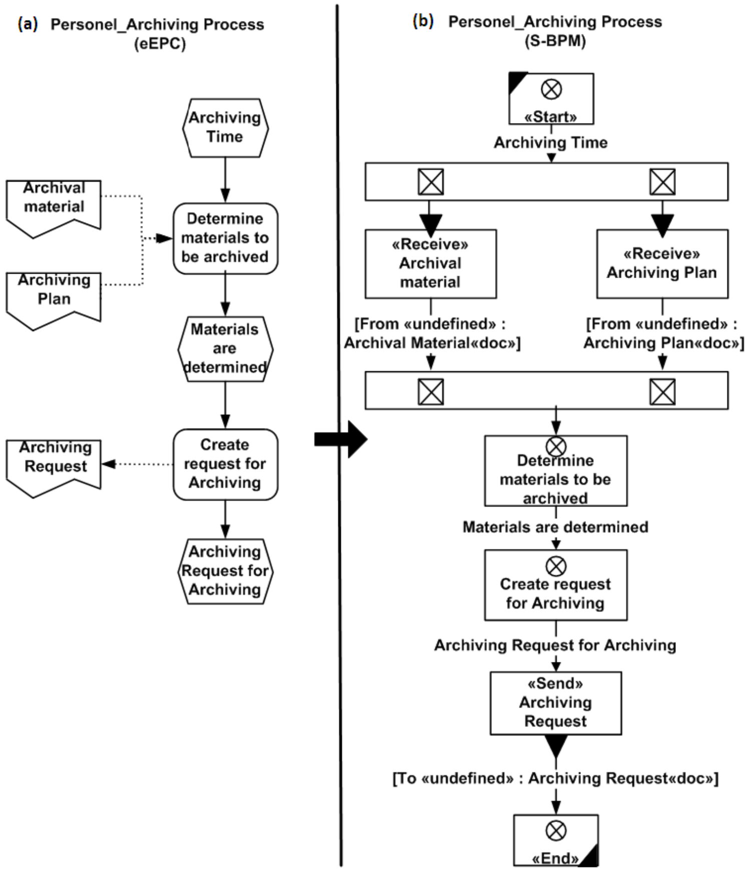

(a) Personel_Archiving Process (eEPC)
Archiving Time
Archival material
Determine materials to be archived
Archiving Plan
Materials are determined
Archiving Request
Create request for Archiving
Archiving Request for Archiving
(b) Personel_Archiving Process (S-BPM)
«Start»
Archiving Time
«Receive» Archival material
«Receive» Archiving Plan
[From «undefined» : Archival Material«doc»]
[From «undefined» : Archiving Plan«doc»]
Determine materials to be archived
Materials are determined
Create request for Archiving
Archiving Request for Archiving
«Send» Archiving Request
[To «undefined» : Archiving Request«doc»]
«End»

Fig. 11. Transformation of Archiving Process for Personnel

In the transformation of Archivist_Archiving Process of Archivist subject, “Events without incoming control flow”, “Split Event”, “Data with outgoing information flow”, “Functions without following event”, “Resource object transformation”, ”Data with incoming information flow”, “Join Functions” and “Events without outgoing control flow” rules are used (Figure 12.a-12.b).

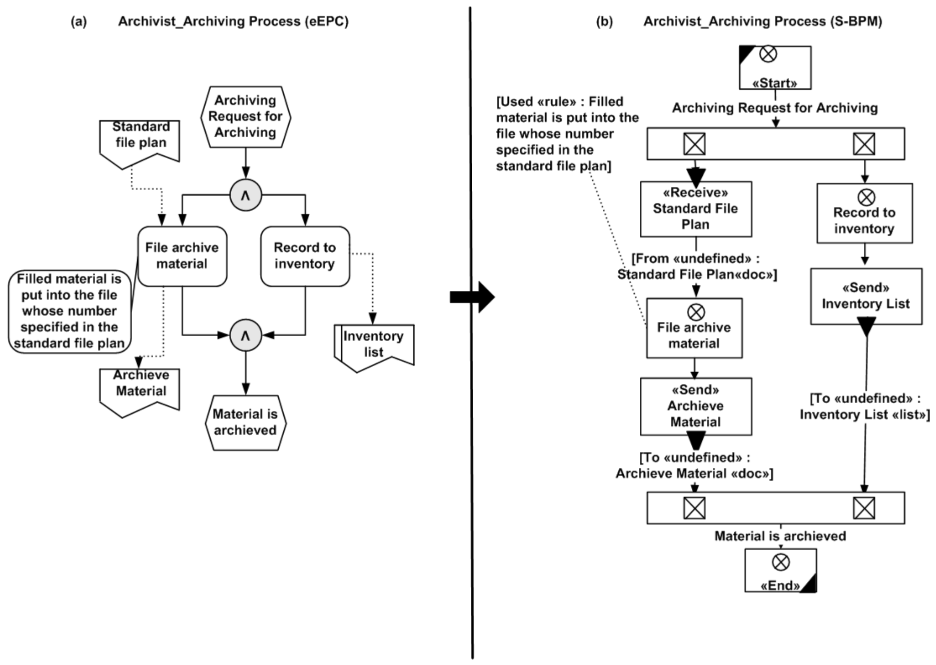

(a) Archivist_Archiving Process (eEPC)
Archiving Request for Archiving
Standard file plan
File archive material
Record to inventory
Filled material is put into the file whose number specified in the standard file plan
Inventory list
Archieve Material
Material is archieved
(b) Archivist_Archiving Process (S-BPM)
«Start»
Archiving Request for Archiving
[Used «rule» : Filled material is put into the file whose number specified in the standard file plan]
«Receive» Standard File Plan
Record to inventory
[From «undefined» : Standard File Plan«doc»]
«Send» Inventory List
File archive material
«Send» Archieve Material
[To «undefined» : Inventory List «list»]
[To «undefined» : Archieve Material «doc»]
Material is archieved
«End»

Fig. 12. Transformation of Archiving Process for Archivist

"Events without incoming control flow", "Data with incoming information flow", "Resource object transformation", "Functions followed by an event" and "Events without outgoing control flow" are used in given order to transform Acquiring archive material process in the view of personnel (Figure 13.a-13.b).

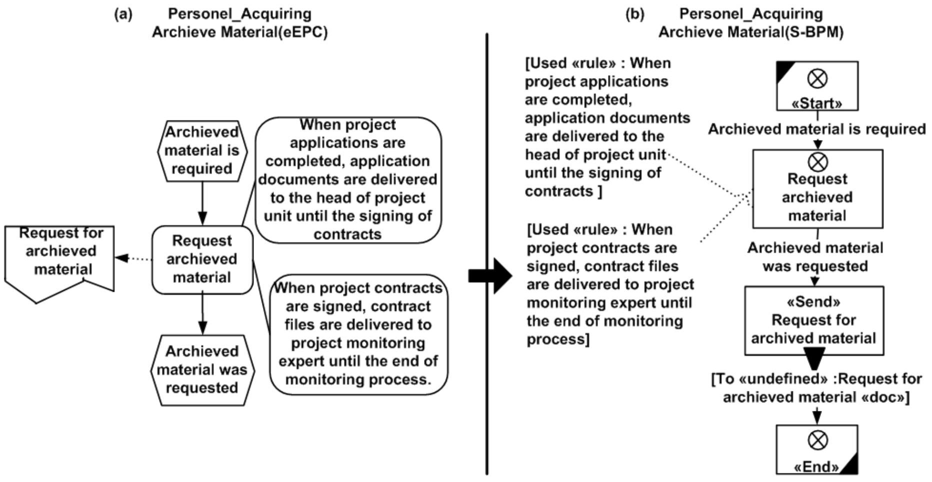

(a) Personel_Acquiring Archieve Material(eEPC)
Archieved material is required
When project applications are completed, application documents are delivered to the head of project unit until the signing of contracts
Request for archieved material
Request archieved material
When project contracts are signed, contract files are delivered to project monitoring expert until the end of monitoring process.
Archieved material was requested
(b) Personel_Acquiring Archieve Material(S-BPM)
[Used «rule» : When project applications are completed, application documents are delivered to the head of project unit until the signing of contracts]
«Start»
Archieved material is required
Request archieved material
[Used «rule» : When project contracts are signed, contract files are delivered to project monitoring expert until the end of monitoring process]
Archieved material was requested
«Send» Request for archived material
[To «undefined» :Request for archieved material «doc»]
«End»

Fig. 13. Transformation of Acquiring Archived Material Process for Personnel

S-BPM model of Achivist_Acquiring archive material process is generated by practicing the following rules respectively; “Events without incoming control flow”, “Data with outgoing information flow”, “Functions without following event”, “Split Function”, “Events without outgoing control flow”, “Functions without following event”, “Data with incoming information flow”, “Split Function”, “Functions followed by an event”, “Events without outgoing control flow” (Figure 14.a-14.b).

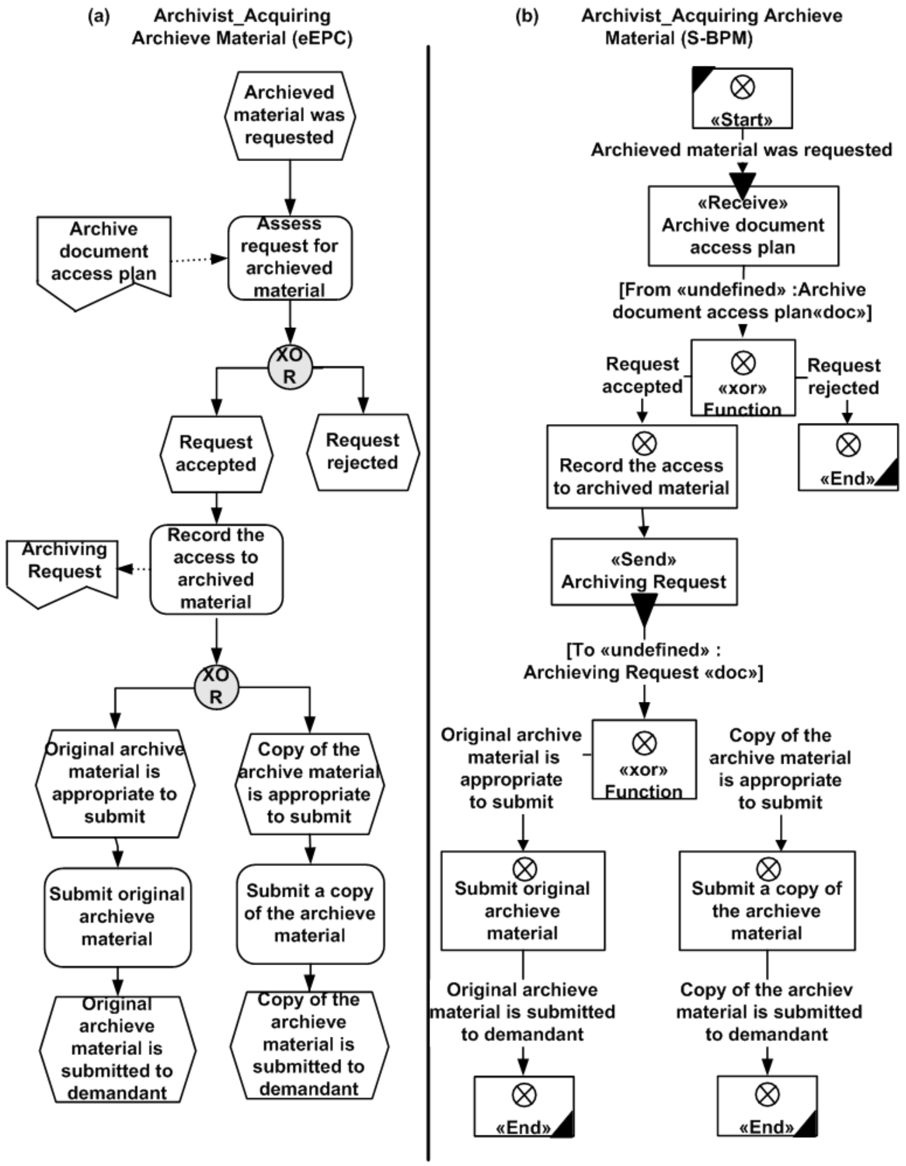

(a) Archivist_Acquiring Archieve Material (eEPC)
Archieved material was requested
Archive document access plan
Assess request for archieved material
XOR
Request accepted
Request rejected
Archiving Request
Record the access to archived material
XOR
Original archive material is appropriate to submit
Copy of the archive material is appropriate to submit
Submit original archieve material
Submit a copy of the archieve material
Original archieve material is submitted to demandant
Copy of the archieve material is submitted to demandant
(b) Archivist_Acquiring Archieve Material (S-BPM)
«Start»
Archieved material was requested
«Receive» Archive document access plan
[From «undefined» :Archive document access plan«doc»]
Request accepted
«xor» Function
Request rejected
Record the access to archived material
«End»
«Send» Archiving Request
[To «undefined» : Archieving Request «doc»]
Original archive material is appropriate to submit
«xor» Function
Copy of the archive material is appropriate to submit
Submit original archieve material
Submit a copy of the archieve material
Original archieve material is submitted to demandant
Copy of the archiev material is submitted to demandant
«End»
«End»

Fig. 14. Transformation of Acquiring Archived Material Process for Archivist

The validity of output S-BPM models are checked manually by traversing the validation rules given in section 3.

7 Discussion and Conclusion

In our case study, eEPC model has been transformed to S-BPM without any information lost by using the defined transformation rules. The generated S-BPM models are valid and semantics of the input model are preserved in output models. As a result of eEPC to S-BPM transformation, individual models for each subject are generated. The output model names give information about subject and related process. However, interactions of the subjects and message flow between them are missing. User can only get this information by tracking the generated models and finding the common points (events) manually. In order to provide traceability, SBD for input process should also be generated. That generation is out of this paper's scope, it is left as a future work.

Duplication is another problem which occurs in the conversion of sending and receiving messages. If there is a function which includes "receive" keyword in its description and an incoming information flow connected to it, information flow part is converted as receive message with «receive» annotation and conversion of function part also includes "receive" keyword which refers to the same data. The duplication problem also occurs in the conversion of outgoing information flow connected to a function that contains "Send" keyword. In order to avoid this problem, description of functions is also taken into account. Information flow and related function is considered as a different pattern and that pattern is mapped to "Receive Message" or "Send Message" element directly. On the other hand, there is no way to show all flow information in a single S-BPM element (e.g. send-receive element) if there are more than one information flow related to the function. Thus, possible duplicates are not handled in order not to complicate transformation.

During the case study, we have had some observations for more understandable model generations. These remarks can be summarized as follows:

- Each function should be followed by an event and each function should be triggered by an event in the input model. Otherwise, null transitions will be occurred in the model.
- Each data object should be related to a subject. Otherwise, subject information will be marked as «undefined».
- In case of eEPC model belongs to only one actor, no information lost occurs in transformation. Otherwise, subject's interactions will be lost.

As a conclusion, in this paper we have defined metamodels (based on MOF) and validation rules for eEPC and S-BPM. The main contribution of the study is defining a transformation approach and it is divided into two phases. In the first phase, individual eEPC models are generated and then eEPC to S-BPM transformation is performed as a second phase. Common eEPC patterns and rules for each of them are defined as guidelines. These guidelines are validated on a real world case study and they are evaluated to be used in the development of the extension to UPROM (work in progress).

References

1. Aguilar-Savén, R.S.: Business Process Modeling: Review and Framework. International Journal of Production Economics (90), 129–149 (2004)
2. Scheer, A.W: ARIS- Modeling Methods, Meta-models, Applications. Springer, Berlin (1998) (in German)
3. Scheer, A.W.: ARIS - Business Process Modeling, 3rd edn. Springer (1999)
4. Fleischmann, A., Schmidt, W., Stary, C., Obermeier, S., Börger, E.: Subject-Oriented Business Process Management (2012)
5. Fleischmann, A.: What is S-BPM? In: Buchwald, H., Fleischmann, A., Seese, D., Stary, C. (eds.) S-BPM ONE 2009. CCIS, vol. 85, pp. 85–106. Springer, Heidelberg (2010)
6. Singer, R., Zinser, E.: Business Process Management — S-BPM A New Paradigm for Competitive Advantage? In: Buchwald, H., Fleischmann, A., Seese, D., Stary, C. (eds.) S-BPM ONE 2009. CCIS, vol. 85, pp. 48–70. Springer, Heidelberg (2010)
7. Sendall, S., Kozaczynski, W.: Model Transformation the Heart and Soul of Model-Driven Software Development. IEEE Software 20(5), 42–45 (2003)
8. Böhme, C.: bflow* Toolbox-an Open-Source Modeling Tool (2011)
9. van der Aalst, W.M.P.: Formalization and Verification of Event-driven Process Chains. Information and Software Technology 41(10), 639–650 (1999)
10. van Hee, K.M., Oanea, O., Sidorova, N.: Colored Petri Nets to Verify Extended Event-Driven Process Chains. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3760, pp. 183–201. Springer, Heidelberg (2005)
11. Nüttgens, M., Feld, T., Zimmermann, V.: Business Process Modeling with EPC and UML: Transformation or Integration? In: Schader, M., Korthaus, A. (eds.) The Unified Modeling Language—Technical Aspects and Applications. Physica-Verlag, Heidelberg (1998)
12. Tscheschner, W.: Transformation from EPC to BPMN. Oryx Research. Potsdam, Germany (2010)
13. Handy, B., Dirndorfer, M., Schneeberger, J., Fischer, H.: Methods of Process Modeling in the Context of Civil Services by the Example of German Notaries. In: Schmidt, W. (ed.) S-BPM ONE 2011. CCIS, vol. 213, pp. 281–295. Springer, Heidelberg (2011)
14. Sneed, S.: Mapping Possibilities of S-BPM and BPMN 2.0. In: Oppl, S., Fleischmann, A. (eds.) S-BPM ONE 2012. CCIS, vol. 284, pp. 91–105. Springer, Heidelberg (2012)
15. Coşkunçay, A., Aysolmaz, B., Demirörs, O., Bilen, Ö., Doğan, İ.: An Approach for Concurrent Business Process Modeling and Requirements Analysis. In: Symposium on Software Quality and Software Development Tools, İstanbul, Turkey (2010)