МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2009
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2009
Proceedings of the Thirty Eighth Spring Conference of
the Union of Bulgarian Mathematicians
Borovetz, April 1–5, 2009

CONVERSION OF PROCESS FLOW DIAGRAMS TO ACTIVITY DIAGRAMS*

Vladimir Dimitrov

Process flow diagrams are used by DG TAXUD for specification of business processes. Software development tools usually support UML activity diagrams for specification of business processes. Conversation of process flow diagrams to activity diagrams is important to be in efficient and flexible way opening doors to Service-Oriented Architecture – the ultimate solution for software development in Grid.

1. Introduction. European Commission's Directorate-General for Taxation and the Customs Union for description of models of Business process model view uses the next notations: 1) Process Flow Diagram, which is providing a graphical representation of the Transit business thread. It shows how the business arranges its processes to respond to external events and to produce results. This specific technique allows for full understanding of the Transit business before detailing the information technology solution supporting those processes. Many businesses have similar processes, but the arrangement of the processes – the dynamics or process flow – may be very different. 2) Textual description of the components of the Transit business thread and shown on the process flow diagram: the (major and minor) events that launch the thread; the processes that are involved in the thread; the (major and minor) results produced by the thread. 3) When applicable, specific assumptions, constraints or remarks will be provided for each of the elements. The Data model view will be limited to the following two models: 1) The model of the State Transition Diagram presents the life cycle of the data, showing the different states and the way it goes from one state to the next one. State Transition Diagrams are provided for the most significant process threads. In practice, State Transition Diagrams are not used. 2) During the execution of the Transit business threads, information is exchanged between processes. The Structure of the information to be exchanged together with its symbolism is not provided here – it is not subject of this paper.

The Commission Enterprise IT Architecture Framework (CEAF) recommends for business process modeling being used UML activity diagrams. This means that Process Flow Diagrams have to be converted to activity diagrams. We will discuss here how this

*2000 **Mathematics Subject Classification:** 68U35.
Key words: Process Flow Diagrams, Activity Diagrams, Service-Oriented Architecture, Rational Unified Process.
This work was supported by the Ministry of Education and Science of Bulgaria under contract VU-MI-109/2005: "Creation and development of Grid infrastructure for research and education at University of Sofia".

conversation has to be done. For this purpose we will first give a short description of Process Flow Diagrams notation.

2. Process Flow Diagrams. Fig. 1 is an example of a Process Flow Diagram which represents the arrival of a Transit Movement at the office of destination, and the handling of the diversion in case the actual office of destination is not the one which has been declared at departure. Process flows are composed of two major elements: the components of the process flow, indicated by an alphabetic enumeration: events, processes, results, process flow breaks, process flow connectors, iterations, locations; the flow between those components, indicated by a numeric enumeration: mandatory flows within a location, mandatory flows between locations, optional flows within a location, optional flows between locations.

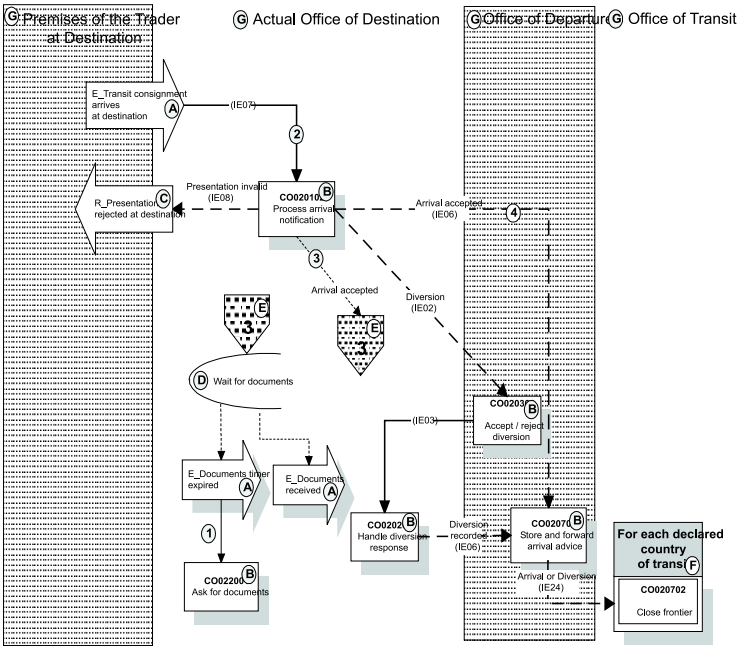

Fig. 1. Process Flow Diagram

An *event* (A) is an occurrence that triggers the business to respond in a predictable fashion. An event causes a sequence of processes to start or to restart after a process flow break, e.g. "E_Transit consignment arrives at destination". The event is represented as a large arrow pointing from left to right and is drawn in the column of the location(s) where it happens. Each event name starts with an E_ followed by the name. An event that can happen at different locations is represented on the diagram as one event overlapping the different location columns where it can happen, e.g. the event "E_Transit consignment arrives at destination" can happen at the 'Premises of the Trader at Destination' or at the 'Actual Office of Destination'.

An *elementary business process (EBP)* (B), also called *process* in the document, is represented as a rectangle containing its identification and its name. Each EBP is

identified by a string of the format 'xxttssss' where: 'xx' identifies the business area, which can be: CO (Core business), GU (Guarantee management), CS (Central services), SA (System administration); 'tt' identifies the process thread in that business area (e.g. CO02 – "Process arrival"); 'ssss' identifies the EBP in the process thread (e.g. CO020102 – "Process arrival notification"). The EBP name is composed by a verb followed by a complement (e.g.: Process arrival notification). A process that can be performed at different locations is represented on the diagram as one process overlapping the different location columns where it can be processed, e.g. the process CO1A1001 – "Verify goods and/or supporting documents" can be performed either at the 'Premises of the Trader at Destination' or at the 'Office of Departure'. A result is a business product put at the disposal of other processes, e.g. "R_Presentation rejected at destination". The result is represented as a large arrow pointing from right to left and is drawn in the column of the location(s) where the result will be used. The result name starts with R_ followed by the name.

A *result* (C) that can be used at different locations is represented on the diagram as one result overlapping the different location columns where it can be used, e.g. the result "R_Presentation rejected at destination" can be used at the 'Premises of the Trader at Destination' or at the 'Actual Office of Destination'.

A *process flow break* (D) occurs when one process in the flow is complete, but the next process cannot start before an event occurs to restart the process flow. When the time interval allowed for the break has elapsed, an internal event is created (in the given example 'E_Document Timer expired') and used as a trigger for the next process. If some condition occurs during the time interval that allows no longer waiting for its end, another internal event is created (in the given example 'E_Document received') and used as a trigger for the next process. The process flow break is represented as a large "U" on its side. It is followed by the events that will cause the process flow to restart. A flow connector is used to clarify the diagram when the next component in the process flow is too far from the previous one. It must be related to process belonging to the same business process thread.

The *flow connector* (E) is represented as a small house on its roof. It is labeled with a number and both components (the previous one and the next one) are in the same diagram, maybe on different pages.

An *iteration* (F) describes the case where sequences of processes need to be repeated a number of times, as if in a loop. The loop may continue a predictable number of times (in this case the statement specifying the number of iterations will start with the word 'For'), or it may continue an unpredictable number of times until some expected verifiable condition is satisfied (starting with the word 'Until').

The process flow diagrams indicate also in background the locations where the processes are executed. Processes are put in the column, or across the columns, whose title identifies the *location* (G) where it is executed.

The above diagram shows that the EBP CO020102 "Process arrival notification" is performed at the location of the "Actual Office of Destination". This process includes, when the arrival is accepted the sending of an information (IE06) to the location of the "Office of Departure". Upon reception of that information, the EBP CO020701 "Store and forward arrival advice" is performed at the location of the "Office of Departure".

The *flows* are represented using a line terminated with an arrowhead which starts

from one component and leads to another component of the process flow: 1) A *mandatory flow within the same location* (1) is represented by a narrow solid line; in this context, 'mandatory' means that the succession of the component is imposed (in the example, the EBP CO020701 "Store and forward arrival advice" always produces the result 'R_Office of Departure notified of arrival (or diversion)'). 2) A *mandatory flow between different locations* (2) is represented by a wide solid line. This flow between locations is always labeled with an identifier of the information to be exchanged in the form IExx. 3) An *optional flow within the same location* (3) is represented by a narrow dashed line; in this context, 'optional' means that the succession of the component depends on a certain condition. So, this flow is always labeled with the name of the condition. 4) An *optional flow between different locations* (4) is represented by a wide dashed line and is always labeled with the name of the condition (because the flow is optional) and the identifier of the Information to be Exchanged.

It must be noted that, depending on the circumstances encountered by an EBP, the latter can produce different sets of flows. As explained above, each optional flow is labeled with the name of a condition. When two or more flows leaving one process are labeled with exactly the same name of condition, which means that they are produced simultaneously (see condition "Arrival accepted"(A)).

An *Information to be Exchanged (IE)* represents the flow of information between two EBP's taking place in different locations. No assumption is made on the communication medium used to carry out this flow of information. The content and structure of all IE's are detailed in Appendix B 'Logical Data Model / Functional Structure of Information To Be Exchanged'. The following remarks apply to all IE's: 1) all IE's are supposed to comply with the structure, the conditions and the rules described in Logical Data Model / Functional Structure of Information To Be Exchanged. This compliance is always checked on the reception side of the IE. In case of non-compliance, the IE is rejected, an advice of non-acknowledgement (NACK) is returned to the sender, the process flow is interrupted, and exception handling measures must be taken; 2) an IE may contain some free text written in the language of the originator.

3. Conversion. In the next following considerations we use UML 2.0 [2] implementation of Rational Software Architect version 7.0.0.6 [3]. Here we use activity diagrams as a tool for specification of business processes, but not as a tool for details specification of use cases. Every process flow diagram we convert to activity diagram. In such a way traversability is supported and it is easy to check conversion correctness. Every location we convert to partition in activity diagram with the same name and same relative arrangements. Usually the leftmost location represents external world for the system (subsystem) modeled on the process flow diagram. From there the starting events are received and there are sent results. The external world can be structured, in which case several locations are used, but it is preferable they to be the leftmost. From implementation point of view, it is possible processing specified in one location to be implemented as a separate subsystem. Such an approach is sensible even if the system is centralized, because every location can be implemented as a service or a package of services in Service-Oriented Architecture (SOA) [4]. But it is possible to implement all the internal locations as an application silo or to combine groups of internal locations. This separation of locations at external and internal ones means that processing in external locations is only outlined at the level of incoming events and out coming results. When the processing

in the external world has to incorporate some regulatory logic, this logic is specified in external world locations. A process flow diagram, usually, specifies processing in several internal locations, and every location can be implemented as separate subsystem. In this case, subsystem of one location is external to the subsystem of the other location. Even more several "truly" external locations could be mixed with internal ones. This note is important if we want more implementation details to put in the diagram. For example, message exchange among subsystems can be modeled with signals exchange, as this is done below for message exchange between external world and the system. In usual manner message exchange is specified as data flows between internal locations.

Events in process flow diagrams start and switch business processes. Based on these roles events can be classified as starting and switching ones. Events represent impact of external world on the business process and results are reported back. Events can be internal and external ones depending of the location type where they occur. Processes are usually started by external events. In some cases, when the subsystem is activated by another one, it is possible starting events to be internal ones. When a business process is started there are two possibilities: a new copy of the process is created and then the process is started or there is only one copy of the process and events are queued for subsequent execution. In the first case is possible very many business processes to be started and executed in parallel. In the second case long queues could be needed to support the only one process. The question is "Is it possible to specify these two cases?" Process flow diagrams do not support any means to specify such details. On the other hand, activity diagrams are Petri nets based, which by their nature are single process approach. Petri nets are extension of state machine, which are used for behavior specification of single class object, but at the same time many objects can exist executing their state machine. Just the same principle is applicable to activity diagrams and business processes. If we want to specify in activity diagram that there are many starting events, then we have to connect single initial node with all of them. In such a way, transition from initial node will be nondeterministically chosen. But if we have many initial nodes, all of them start their own concurrent execution thread – this approach is used for parallelism modeling. Object and control tokens are automatically queued in activity diagrams, but if we want to specify this behavior clearly it is better to use object nodes of types central store and data store to control object tokens and control nodes for both object and control tokens. Event from process flow diagram has to be modeled in activity diagrams by a sequence of constructions. First of all process flow event is a kind of signal and we have to create a signal in activity diagram with the name of message that the event carry. If there is no such a message (the event is simple control event) then the name of the signal is the event name. Second step is to send this signal and then that signal has to be accepted as signal event. The signal event name can be the signal name. This naming convention can be used if there is no semantic conflict with the other events. The event "E_Transit consignment arrives at destination" from Fig. 1 is modeled with Send Signal node with the same name and a signal with the name of the message that has to be delivered – IE07 in that case. This event can occur in two locations: "Premises of the Trader at Destination" and "Actual Office of Destination". We place Send Signal node in the first partition and add manually the second one. Send Signal node creates a signal object in the right partition.

Next we create Accept Event node in partition "Actual Office of Destination". This

node is triggered by signal event with the name “E_Transit consignment arrives at destination”, which we create using the signal IE07. Here we use again the event name as a signal event name. See whole this conversion on Fig. 2. Elementary business processes we represent as Action node with the same name, as it is shown at Fig. 2 with “CO020102 Process arrival notification”. This elementary business process generates result to the external world under condition “Presentation invalid”. This condition can be modeled with Decision node, but instead we prefer transition with guard condition, which style is closer to the original. The result we send back to the external world as a signal in a way analogous to receiving events. One more thing in the process flow diagram is that return of the result is optional – the transition is with dotted line. We have no way to represent such an option and only what can do is to color optional elements. Just a same strategy of coloring can be used to show internal and external elements of modeled system. It is very useful for model verification because attention is on internal elements. For example, the system receives external events in two main steps with Send Signal node and Receive Event node. The first one is external to the system element, but the second one is internal, and can be colored differently. Coloring schema can be used to specify what will be automated and what not.

The next step is to represent message exchanges in the system. We use object flows, again because it is closer to the original. It is possible to use the asynchronous approach with signals exchange, but the diagram will be more complex. If we intend further to specify more details, we can do that – for example, signal exchange is natural way for specifying SOA, but at that time (business modeling) still we have no idea for system architecture. At this time interaction modeling with RSA is impossible. Such a feature can be modeled with Loop node, but currently we only put a Note connected with the loop body. It is possible to model loop with a set of Control nodes, but the diagram will be more complex without any benefit.

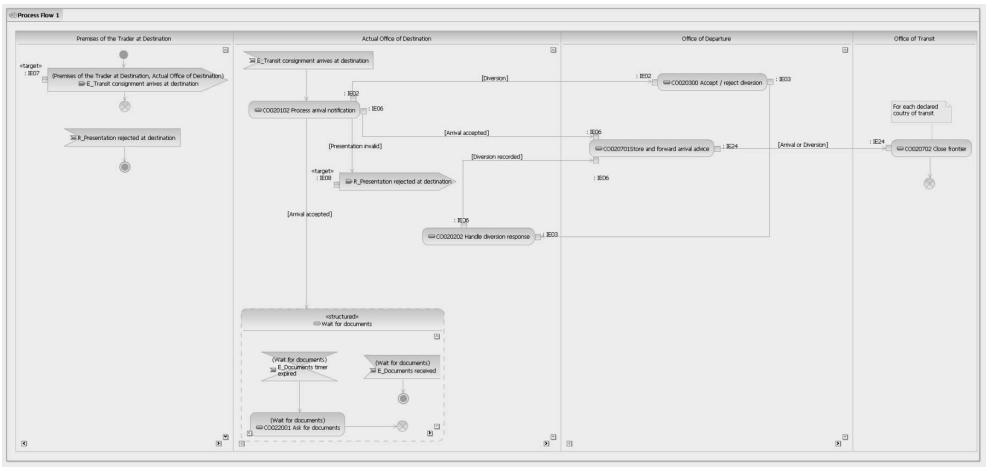

Fig. 2. Activity diagram

The timer we model with Structured Activity node. There is no Initial node in this node which means that exit from that activity is only when documents are received, in

which case the whole business process is stopped. After "CO022001 Ask for documents" the process flow is stopped and we wait for documents again or for timer to expire. Another more direct way for timer representing is to use expansion region, but it is still not available in RSA.

Finally, we use Flow Final node where is possible some control or object token to stay unused.

4. Conclusion. Some topics remain out of scope of the paper, but even this simple example is presented the power of UML 2.0 activity diagrams for business modeling. Using incremental development of Rational Unified Process (RUP) [5], starting from activity diagrams it is possible to specify many architectural decisions in them, before applying other RUP features.

REFERENCES

[1] Taxation and Customs Union DG, FITSDEV Project. Specifications, Development, Maintenance and Support for communication and information exchange systems in the taxation and excise area,
http://www.uni-mannheim.de/edz/pdf/sek/2007/sek-2007-0696-en.pdf

[2] UML 2.0, http://www.uml.org

[3] IBM Rational Software Architect,
http://en.wikipedia.org/wiki/Rational_Software_Architect

[4] Service-oriented architecture,
http://en.wikipedia.org/wiki/Service-oriented_architecture

[5] IBM Rational Unified Process,
http://en.wikipedia.org/wiki/IBM_Rational_Unified_Process

Vladimir Dimitrov
Faculty of Mathematics and Informatics
University of Sofia
5, James Bourchier Blvd
1164 Sofia, Bulgaria
e-mail: cht@fmi.uni-sofia.bg

ПРЕОБРАЗУВАНЕ ДИАГРАМИТЕ НА ПРОЦЕСНИТЕ ПОТОЦИ В ДИАГРАМИ НА ДЕЙНОСТТА

Владимир Димитров

Диаграмите на процесните потоци се използват от DG TAXUD за спецификация на бизнес процеси. Инструментите за разработка на софтуер обикновено поддържат UML диаграмите на дейността за спецификацията на бизнес процесите. Преобразуването на диаграмите на процесните потоци в диаграмите на дейността трябва да става по ефективен и гъвкав начин в посока на архитектурата, ориентирана към услуги, която е решението за разработка на софтуер в Грид.