A data-flow language for business process models

Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 16 (2014) 128 – 137

Procedia Technology

CENTERIS 2014 - Conference on ENTERprise Information Systems / ProjMAN 2014 - International Conference on Project MANagement / HCIST 2014 - International Conference on Health and Social Care Information Systems and Technologies

A data-flow language for business process models

Giorgio Bruno

Politecnico di Torino, c.so Duca degli Abruzzi 24, 10129 Torino, Italy

Abstract

On the basis of the artifact-centric approach to business process modeling, this paper proposes a notation named ACTA that provides two equivalent forms of representation: in one, all the life cycles of the artifacts involved in the business under consideration are shown in a single model, while in the other they are defined in separate models. The latter may be easier to understand when the model is large but requires synchronization points between life cycles: two kinds of mechanisms, i.e., driven transitions and driven tasks are analyzed in this paper. The major feature of ACTA is its nature of data flow language: the activation of tasks depends on the presence of suitable input entities rather than on the precedence relationships between tasks as it takes place in the conventional activity-centric approach. The advantage is the ease with which a number of situations that are difficult to handle with the activity-centric approach can be managed. Such situations encompass the selection of homogeneous entities to be processed in batches and the many-to-many mapping between entities of different types, such as requisition orders and procurement orders.

Peer-review under responsibility of the Organizing Committee of CENTERIS 2014.

Keywords: business processes; data flow; life cycles; tasks.

1. Introduction

In the traditional view [1], a business process is an organizational tool meant to provide a product or service through a number of activities (or tasks) that are performed on the basis of predefined precedence rules. This approach is called activity-centric and its main representative is the BPMN notation [2]. Tasks may be divided into human tasks and automatic ones: the former are carried out by persons playing specific roles, e.g. customers and account managers, and the latter are performed by software components. The precedence rules form the control flow of the process and are based on the completion events of tasks. The implementation of tasks is outside the scope of the process, which acts as an orchestrator of activities: the process elements representing tasks are placeholders of

Peer-review under responsibility of the Organizing Committee of CENTERIS 2014.
doi:10.1016/j.protcy.2014.10.076

external activities and are intended to pass them input parameters and to receive results. Input parameters and results are mapped to process variables.

If business processes are meant to operate on the business entities that form an information system, such as in the case-handling approach [3], a strong correlation between variables and business entities is needed. For this reason, an extension has been made to BPMN: the entities acted on by tasks are shown by means of a new kind of icons called data objects. This extension allows the data flow to be represented in addition to the control flow. However, the precedence rules are still based on the control flow in that the data flow is a kind of "afterthought" [4] which may be used to introduce additional constraints.

The activity-centric approach has been criticized for not giving the right value to business entities, and then an alternative approach, called artifact-centric [5], has been proposed so as to put the major entities (also called artifacts) and their life cycles in the foreground. Several notations have been presented and they share the view that a business process is built on the life cycles of the artifacts related to the business under consideration. Life cycles are defined by means of separate models: however, they cannot evolve independently of each other and then various synchronization mechanisms are provided.

This paper presents the results of a research grounded on the principles of the artifact-centric approach; in particular, it illustrates a notation for business processes, named ACTA, which supports both compact models and compound ones. Compact models integrate the life cycles in one model, while compound models are made up of separate life cycles with synchronization points based on driven transitions and driven tasks. A driven transition is a state change caused by the execution of a task in a different life cycle while a driven task is a task whose activation is decided in a different life cycle.

A process model, be it compact or compound, is accompanied by an information model that shows the entity types involved in the process, their relationships and attributes.

ACTA is a data flow language in that the activation of tasks depends on the presence of suitable entities in their input flows; the control flow of activity-centric notations is not needed. An important consequence of making the activation of tasks depend on the data flow rather than on the control flow is the ease with which a number of situations that are difficult to handle with the activity-centric approach can be managed. Such situations encompass the selection of homogeneous entities to be processed in batches and the many-to-many mapping between entities of different types, such as requisition orders and procurement orders. The difficulty has been placed in relation to the weakness of the notion of process instance [6]. In ACTA, the notion of process instance is not needed in that all the primary information is contained in the entities forming the data flow: process variables, which are the main reason for the introduction of process instances, are not required.

ACTA models are conceptual models that aim at making the operation of business processes understandable by non-specialists. For this reason, a number of patterns and conventions are emphasized: they help readers understand how the data flow is handled and what the constraints and effects of tasks are.

The organization of this paper is as follows. Section 2 explains the major features of ACTA and section 3 illustrates ACTA patterns and conventions with the help of an order-handling process that includes driven transitions. Section 4 is about driven tasks; sections 5 and 6 present the related work and the conclusion, respectively.

2. Process models with ACTA

This section explains the major features of ACTA and the structure of models based on this notation. ACTA is a data flow language in that the activation of tasks depends on the availability of suitable input entities; on the contrary, in the activity-centric approach, tasks are mainly activated on the basis of the completion events of the tasks previously performed.

In ACTA process models, tasks are not directly connected with each other by means of precedence links; they are instead connected to input and output (state) nodes; nodes and connections form the structure of the data flow that feeds tasks and is fed by them.

ACTA assumes a tight integration with information systems and, for this reason, the data flow consists of business entities. The notion of process variable is not needed in that all the information resides in the business

entities. As a consequence, also the notion of process instance is not needed: a process is a structure of work through which all the entities to be handled are entitled to flow.

ACTA handles two types of models, i.e., compact models and compound ones. A compound model is made up of separate life cycle models. A life cycle model is associated with an entity type and then it defines the life cycle of the entities of that type; the initial state is named i. A compact model is an integrated model including all the life cycles of the entities managed by the process: the life cycles may be intertwined in various ways and then each state node is associated with an entity type and a state name.

Process models are accompanied by information models that show the types of the entities along with their relationships and attributes. Entity types can be divided into three categories, role types, managed types and contextual ones as will be explained in the next section.

The choice between compact models and compound ones depends on the nature and complexity of the process under consideration. Compound models may be more flexible and easier to understand than large compact models, but they imply the burden of introducing correlation mechanisms between life cycles: two such mechanisms are illustrated in this paper, i.e., driven transitions and driven tasks. A driven transition makes the entities in a life cycle change state as the result of a task performed in another life cycle; a driven task is a task whose activation is decided in another life cycle.

Two examples are provided in the next sections. The first example concerns a process handling a many-to-many mapping between requisition orders and procurement ones, and the second one addresses a process managing the papers submitted to a conference.

3. Patterns and conventions

ACTA models are conceptual models that aim at making the operation of the process understandable by non-specialists. For this reason, a number of patterns and conventions are emphasized: they help readers understand how the data flow is handled and what the constraints and effects of the tasks are.

This section illustrates ACTA patterns and conventions with the help of an order-handling process named HandleRequisitionOrders. The process is first defined with a compact model and then with a compound one that features driven transitions. The simplified requirements of the process are as follows.

The process is run by a certain company and involves external actors (customers and suppliers) and staff members (account managers). Customers may place requisition orders that are made up of lines each one referring to a product type. Each customer is served by a specific account manager. Account managers may reject or admit requisition orders, and may bundle the lines of admitted ones into procurement orders directed to suppliers. Suppliers fulfill procurement orders and then the related lines are considered to be fulfilled. When all the lines related to a requisition order have been fulfilled, the account manager can fulfill the requisition order. Customers are informed of the rejection or fulfillment of their orders.

The compact model of the process along with the companion information model is shown in Fig.1.

The information model and the related conventions are explained first. An ACTA information model is similar to a UML (Unified Modeling Language) [7] class model and shows entity types and relationships. Attributes are defined in annotations; for simplicity, they are not considered in this paper. The main conventions concern the categories of entity types and required relationships.

An information model encompasses the entity types needed in the process under consideration. They are divided into three major categories: role types, managed types and contextual ones. Role types are told apart by their names, i.e., Customer, Supplier and AccountMgr. Managed types represent the entities (artifacts) forming the data flow of the process; their names are underlined. The remaining types denote contextual entities, e.g. product types, which provide background information: the process does not generate contextual entities but may introduce associations between managed entities and contextual ones. In process HandleRequisitionOrders, role types are contextual types as well.

Required relationships represent associations which must be set when new artifacts are generated; they are shown as oriented links, i.e., connections ending with an arrow on one side or on both sides. In the first case, a new artifact of the source type must be connected to entities of the destination type (the number depends on the multiplicity of the relationship); in the second case, the generation of entities of both types is mutually implied.

For example, the relationship between ROrder and Customer is required on the ROrder side: it expresses the constraint that a new requisition order needs to be connected to the customer entity representing the customer who placed it. The relationship between ROrder and Line is required on both sides: this means that a requisition order and its lines are generated in the same transaction.

A compact process model consists of tasks, state nodes and connections (also called links). Tasks are depicted as rectangles and state nodes as circles. Connections are oriented links drawn from tasks to state nodes or from state nodes to tasks. The input state nodes of a task indicate which entities may cause its execution. Tasks may take one or more entities from each input state node and may deliver one or more entities to each output state node; the number of entities is shown next to the corresponding link and is called weight of the link. The default weight is 1 and is omitted. Weight n indicates that the number of entities is not predefined but is determined during the execution of the task.

Tasks are divided into human tasks and automatic ones; the former are told apart because they have a role label. The role label indicates the role of the process participant who is entitled to perform the task; the actual participant is called task performer and may be specific or generic. A generic participant is any member of the role; in this case, the qualifier (any) is written after the role name. In most cases, task performers are specific participants who bear relationships with the entities to be acted on. The basic rule (called performer rule) is as follows: if there is a unique path based on required relationships from an artifact type, say, A, to a role type, say, R, and the result is one entity, then this entity denotes the specific performer of any task having A as one of its input types and R as its role label. Therefore, the account manager who can admit or reject a requisition order is the one associated with the customer who placed the order, as indicated by the path ROrder-Customer-AccountMgr. The relationship between Customer and AccountMgr is assimilated to a required relationship from Customer to AccountMgr in that it has the multiplicity 1 on the AccountMgr side.

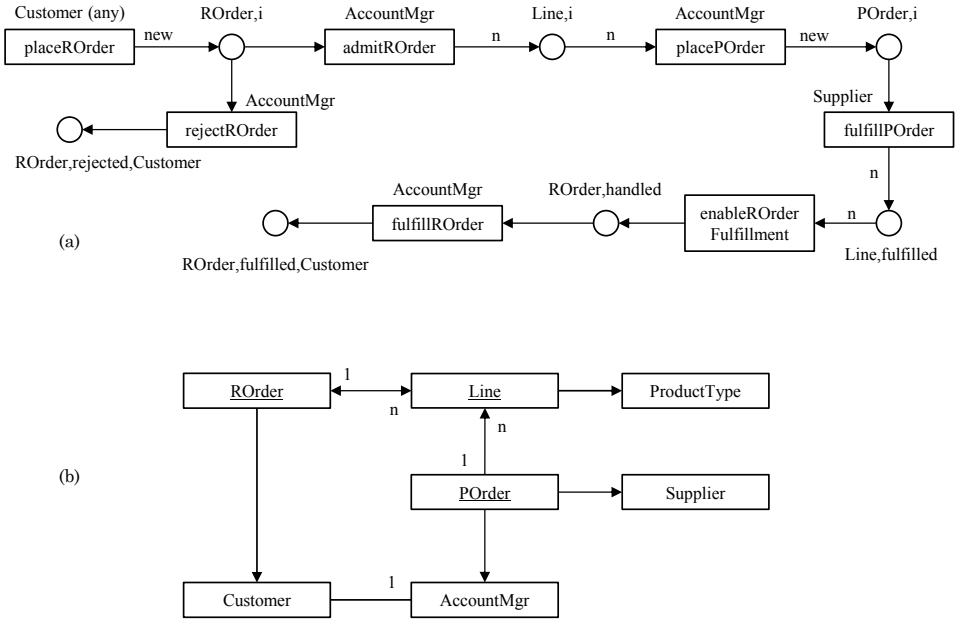

Fig. 1. The compact model of process HandleRequisitionOrders (a) and the companion information model (b)

Tasks may be given descriptions that encompass pre-conditions, post-conditions and output directives. In a first phase of development, the descriptions may be omitted and the operation of tasks may be inferred from the model on the basis of a number of data flow patterns as follows.

The data flow may be fed, transformed or evolved by tasks. It is fed by source tasks, i.e., tasks having no input links, such as task placeROrder; it is transformed by tasks whose output types do not match the input ones and it is evolved by tasks whose output types match the input ones. If a task delivers a newly generated entity to an output node, then the "new" label is placed on the corresponding output link and the state of the destination node must be the initial one; this is called generative effect.

Task placeROrder is a source task, while task admitROrder is a flow-transformer one. As a basic convention, flow-transformer tasks have one input node and one output node and there is a required relationship from the input type (say, T1) to the output one (say, T2) in the information model. If the cardinality of the relationship is one-to-many, the task takes one input entity of type T1 and delivers all the related entities of type T2 to the output node. Therefore, the effect of task admitROrder is to deliver all the lines of the input requisition order to the output node. On the contrary, if the cardinality of the relationship is many-to-one, the task waits until the input node contains all the entities of type T1 that are associated with the same entity of type T2, and then it takes all of them and delivers the T2 entity to the output node. The effect of the automatic task enableROrderFulfillment is to output a requisition order when all its lines have been fulfilled. Then, the account manager can perform task fulfillROrder knowing that all the products needed by the requisition order have been received. This task evolves the life cycle of the input entity from state handled to state fulfilled. The output node of task fulfillROrder is a final node as it has no output links. If a final node has an additional label (the third one), which is a role name, then the node is a notifier: when an entity enters the node a notification is sent to the specific participant identified by means of the performer rule that has been illustrated in the previous section. Therefore, when a requisition order has been fulfilled, the customer who placed it is informed.

The compound model of process HandleRequisitionOrders is shown in Fig.2. The information model is not included as it is identical to the one in Fig.1.

Life cycle models are state transition diagrams where states correspond to nodes in compact models; the state labels are the state names. Transitions are divided into primary transitions and driven ones. The former are associated with tasks while the latter are shown as simple links because they are caused by tasks (called drivers) belonging to other life cycles. The initial transition, i.e., the one entering the initial state of a life cycle, may be a driven transition, too. Basically, a driver makes entities (related to the input ones) follow a driven transition; this is shown by output links (called correlation links) whose labels indicate the name and the state of the target life cycle. If the state is missing, the initial state of the target life cycle is implied. The entities affected are identified on the basis of the data flow patterns above.

Two initial transitions are driven by tasks admitROrder and placePOrder: the former enters the lines associated with the input requisition order in the initial state of the Line life cycle and the latter starts the life cycle of a newly generated procurement order. Two non-initial driven transitions are brought about by tasks fulfillPOrder and enableROrderFulfillment.

A life cycle is a partial process in that it is not self-contained because of the correlation links included. However, as a process, it is in charge of handling all the entities of the type it takes care of: there are no instances of life cycles as there are no instances of compact processes. As a matter of fact, lines are handled in batches as indicated by the weight n of the transitions in their life cycle.

Compact models and compound ones are equivalent in the sense that a compact model can be turned into a compound one and vice versa. For lack of space, the mapping algorithm is not presented in this paper.

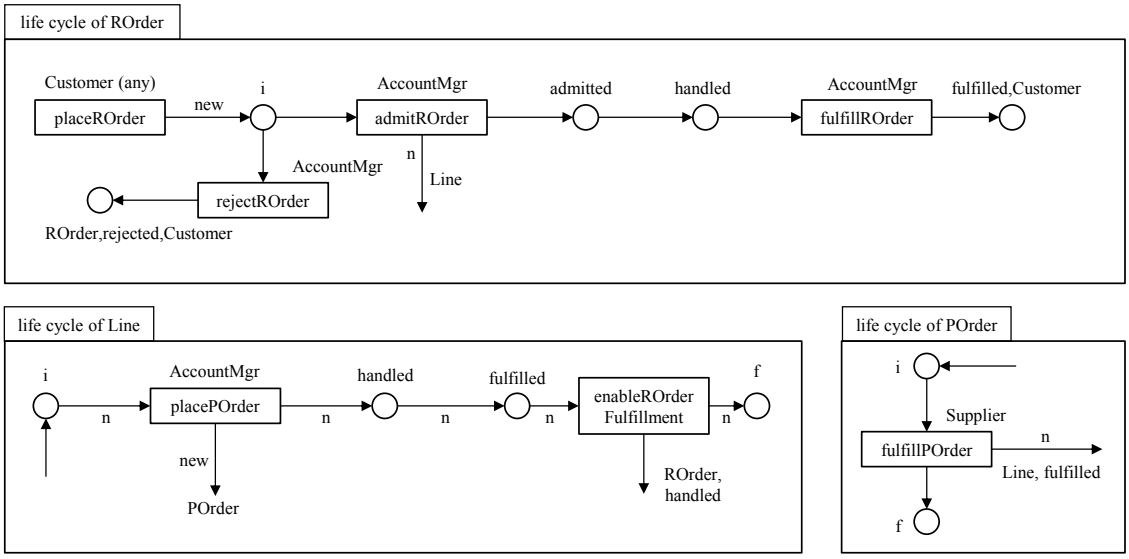

Fig. 2. The compound model of process HandleRequisitionOrders

4. Driven tasks

While a driven transition is a state change caused by the execution of a task in a different life cycle, a driven task is a task whose activation is decided in a different life cycle. Therefore, the activation of a driven task not only depends on the presence of suitable entities in its input states but also requires an enabling action to be performed in a different life cycle. Driven tasks are used to represent situations in which the life cycle of an entity type, say, B, is controlled by the life cycle of another entity type, say, A; A is then the master of B and B is subordinate to A. An example is provided by a process handling papers submitted to conferences; its simplified requirements are as follows.

The process named ManageConference enables a chairman to enter a conference entity containing some information about a forthcoming conference; the attributes of interest for the current analysis are a series of deadlines, d1, d2 and d3. The submission period starts at d1 and ends at d2; during this period, authors may submit papers and they also may update or withdraw the papers submitted. After d2, the chairman closes the conference if there are too few papers (this condition is not further detailed) or else he/she assigns each paper to three referees. Then, referees review papers and, when all of them are finished, for each paper the chairman decides whether to accept or reject it. After that, all the authors are informed and those having their papers accepted must provide the final versions within deadline d3.

Process ManageConference is based on three artifact types, i.e., Conference, Paper and Assignment. Moreover, Conference is the master of the other two artifacts, which are processed in batches. In fact, the life cycle of a conference is a sequence of stages in which papers are collected, assigned to referees, reviewed by referees, evaluated by the chairman and finalized by authors. For this reason, the representation of the Conference life cycle is based on master tasks, i.e., tasks that are meant to operate on the life cycles of subordinate entities, such as papers and assignments. The Conference life cycle is shown in Fig.3 along with the information model of the process. Master tasks are shown as rectangles with rounded angles; they include the task name, e.g. collectPapers, an optional time limit, e.g. (d1..d2), and the name of the target life cycle preceded by the “with” keyword. The actual subordinate entities affected by the master task are those identified by a unique path of required relationships starting from the master entity type and ending in the subordinate entity type; therefore, the subordinate papers are

those directly connected to the conference entity under consideration and the subordinate assignments are those indirectly connected via papers.

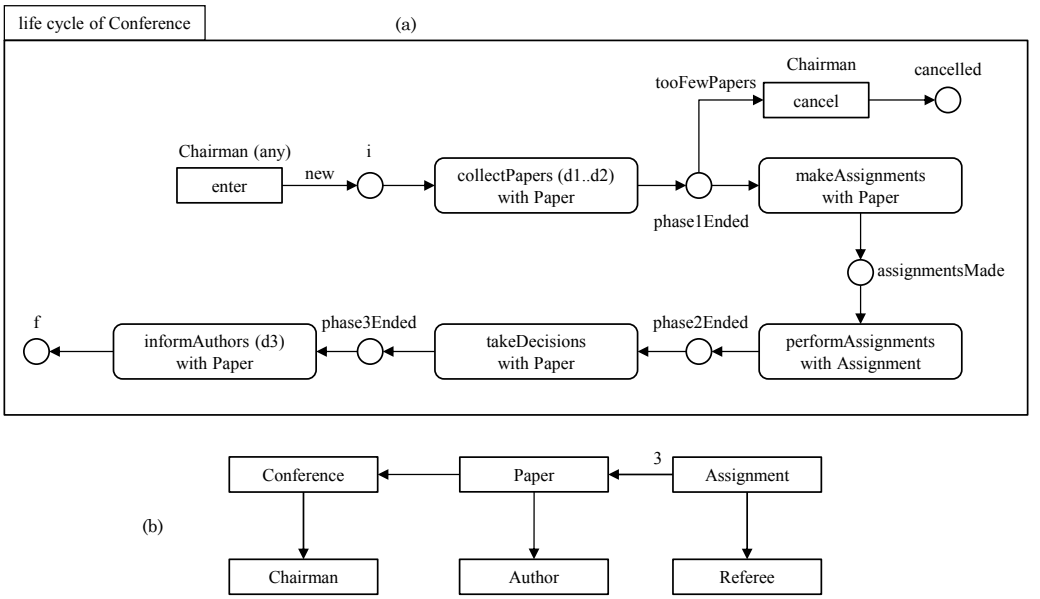

Fig. 3. The Conference life cycle (a) and the companion information model (b)

The life cycles of papers and assignments are shown in Fig.4. Driven tasks are shown in swim lanes whose headings match the names of the master tasks in the Conference life cycle. All the tasks operating on papers and assignments are driven ones.

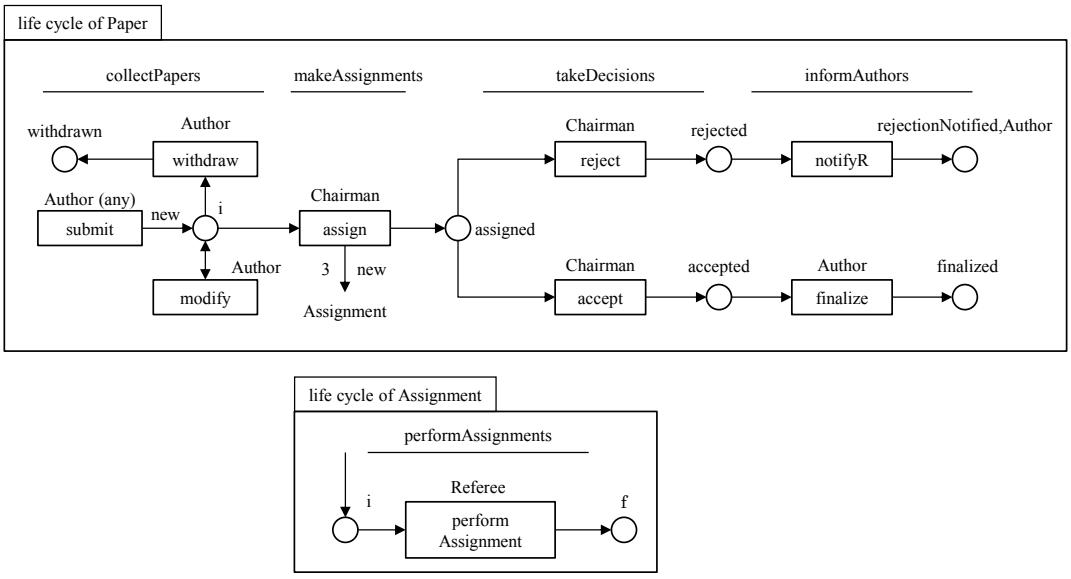

Fig. 4. The life cycles of subordinate entities

When a conference enters the initial state, master task collectPapers is activated; it enables the corresponding swim lane in the Paper life cycle in the interval from d1 to d2. In that period of time, any author may submit a paper for the conference or they may modify or withdraw a paper previously submitted. At d2, the conference entity enters state phase1Ended and an automatic choice takes place. Such choices may be indicated by conditions on output links; condition tooFewPapers needs to be refined and this will take place in a subsequent stage of development. In case there is a sufficient number of papers in the initial state, master task makeAssignments is activated and then the chairman can produce three assignments for each paper; the overall effect is to bring papers into state "assigned" and assignments into the initial state (with driven initial transitions). Papers remain in state assigned until the conference life cycle enables swim lane "takeDecisions": this occurs after master task performAssignments has been carried out. This master task drives task performAssignment for all the assignments subordinate to the conference. The overall effect of master task takeDecisions is to make papers accepted or rejected. When the next master task is performed, rejections are notified to the authors of rejected papers, while the authors of accepted papers are expected to provide the final versions with task finalize within deadline d3.

The master tasks of the Conference life cycle end when their upper time limits elapse, if they are time-driven, or else when all the subordinate tasks have been carried out. A time-driven master task includes optional tasks, i.e., task that are not required but are subject to the will of their performers; tasks modify, withdraw and finalize are optional tasks. The compact model of the process is shown in Fig.5.

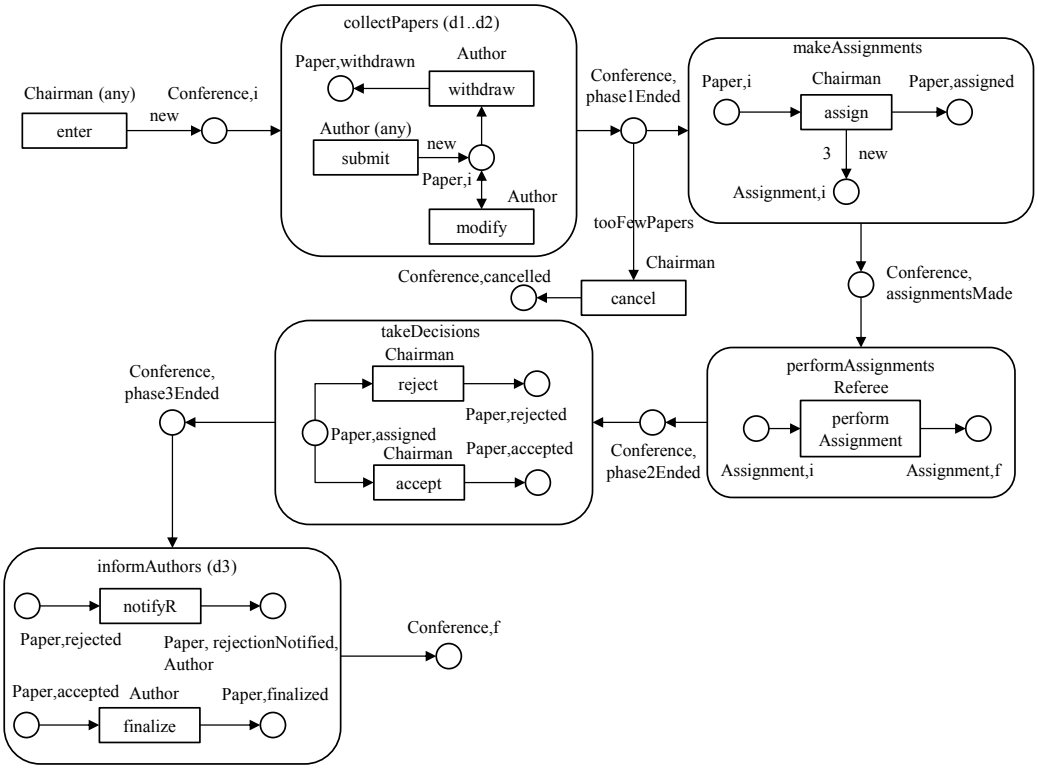

Fig.5: The compact model of process ManageConference

Basically, the compact model is obtained from the compound one: the primary structure is the one of the Conference life cycle with the master tasks turned into compound tasks including the driven tasks of the compound model. The resulting model, because of its size, may appear more difficult to understand and to manage than the corresponding compound one.

5. Related work

The artifact-centric approach emphasizes the role played by artifacts in business processes: the main benefit is the right level of granularity, which facilitates communication among the stakeholders and helps them focus on the primary purposes of the business [8]. Several notations have been proposed, such as GSM (Guard-Stage-Milestone) [9], COREPRO [10] and PHILharmonicFlows [11]. They differ in the representation of the life cycles and in their coordination. Coordination is achieved through events and rules in GSM, is based on hierarchical (container-components) relationships in COREPRO and is obtained by means of macro processes acting on the states of the life cycles in PHILharmonicFlows.

In ACTA compound models, there is no need of additional coordination models since life cycles may include correlation links that affect other life cycles; the identification of the entities affected is based on the relationships defined in the companion information model.

ACTA also provides compact models, which make the coordination of life cycles easier to represent; however, the difficulty of understanding a compact model grows rapidly with its size.

Compact models have also been presented in other papers [12,13,14]. The purpose of ACTA, however, is to leverage the data flow so as to represent situations that are not adequately supported by the functionalities of typical workflow systems [6]: an example is the many-to-many mapping between requisition orders and procurement ones in build-to-order processes. That issue has also been addressed with an extension [15] to BPMN; the solution, however, presents some limitations due to the rigidity of the control flow in BPMN.

While the data flow is a kind of “afterthought” in business process models [4], such as BPMN, in ACTA the entity flow supersedes the control flow in order to improve flexibility: the life cycles may evolve independently of each other until activities calling for synchronization are needed.

ACTA can also easily represent human choices concerning the selection among different courses of action, as shown in a previous version of the notation [16].

Data flow patterns have also been addressed in [17]: the purpose, however, is to analyze the exchange of information between process variables and tasks. On the contrary, data flow patterns in ACTA mainly concern the mapping between the input and output entities of tasks.

6. Conclusion

This paper has presented the ACTA notation, which addresses the artifact-centric approach to business process models by providing two equivalent forms of representation, referred to as compact and compound. Compact models integrate the life cycles of the artifacts needed by the process in one model, while compound models are made up of separate life cycles with synchronization points. In both cases, a companion information model is provided: it shows the entity types involved in the process, their relationships and attributes.

The central point is the identification of the best form for the process under consideration. For this reason, the major ways of intertwining between life cycles must be analyzed. Two such ways have been addressed in this paper with the help of two examples: in the first example, life cycles appear to be interspersed, while, in the second one, they are nested. Driven transitions are appropriate in the first case and driven tasks in the second one. As to the choice of the form of representation, in the first case, the compound model makes it more difficult to grasp the continuity of the process because life cycles progress in stages with mutual influence. On the contrary, in the second case, there is a master life cycle that controls the life cycles of the other artifacts and then the compound model turns out to be easier to understand. Current work is dedicated to the analysis of other forms of intertwining such as those

resulting from spanning tasks, i.e., tasks acting on different life cycles. An example is a task matching requests and offers; in such cases, the choice of the life cycle in which to include the spanning task may be difficult.

Another line of research is the study of the impact of the artifact-centric approach on work lists; in particular, the handling of driven tasks in relation with their master tasks must be worked out.

References

[1] Davenport TH, Short JE. The new industrial engineering: information technology and business process redesign. Sloan Management Review 1990; Summer:11-27.
[2] BPMN, Business Process Model and Notation, V.2.0.2. Retrieved February 6, 2014, from http://www.omg.org/spec/BPMN/2.0.2/.
[3] van der Aalst WMP, Weske M, Grünbauer D. Case handling: a new paradigm for business process support. Data & Knowledge Engineering 2005; 53:129-162.
[4] Sanz, JLC. Entity-centric operations modeling for business process management - A multidisciplinary review of the state-of-the-art. In: 6th IEEE Int. Symposium on Service Oriented System Engineering. New York: IEEE Press, 2011. p. 152-163.
[5] Nigam A, Caswell NS. Business artifacts: an approach to operational specification. IBM Systems Journal 2003; 42:428-445.
[6] Sadiq S, Orlowska M, Sadiq W, Schulz K. When workflows will not deliver: the case of contradicting work practice. In: 8th International Conference on Business Information Systems. 2005.
[7] UML, Unified Modeling Language, V.2.4.1. Retrieved February 6, 2014, from http://www.omg.org/spec/UML/2.4.1/.
[8] Chao T, et al. Artifact-based transformation of IBM Global Financing. In: Lecture Notes in Computer Science, 5701. Heidelberg: Springer; 2009. p. 261-277.
[9] Hull R, et al. Introducing the Guard-Stage-Milestone approach for specifying business entity lifecycles. In: Lecture Notes in Computer Science, 6551. Heidelberg: Springer; 2011. p. 1-24.
[10] Müller D, Reichert M, Herbst J. Data-driven modeling and coordination of large process structures. In: Lecture Notes in Computer Science, 4803. Heidelberg: Springer; 2007. p. 131-149.
[11] Künzle V, Reichert M. PHILharmonicFlows: towards a framework for object-aware process management. Journal of Software Maintenance and Evolution: Research and Practice 2011; 23:205-244.
[12] Bruno G. Combining information and activities in business processes. In: Lecture Notes in Computer Science, 7447. Heidelberg: Springer; 2012. p. 481-488.
[13] Kumaran S, Liu R, Wu FY. On the duality of information-centric and activity-centric models of business processes. In: Lecture Notes in Computer Science, 5074. Heidelberg: Springer; 2008. p. 32-47.
[14] Kucukoguz E, Su J. On lifecycle constraints of artifact-centric workflows. In: Lecture Notes in Computer Science, 6551. Heidelberg: Springer; 2011. p. 71-85.
[15] Meyer A, Pufahl L, Fahland D, Weske M. Modeling and enacting complex data dependencies in business processes. In: Lecture Notes in Computer Science, 8094. Heidelberg: Springer; 2013. p. 171-186.
[16] Bruno G. The representation of human choices in business process models. Procedia Technology 2013; 9:54-63.
[17] Russell N, ter Hofstede AH, Edmond D, van der Aalst WM. Workflow data patterns: Identification, representation and tool support. In: Lecture Notes in Computer Science, 3716. Heidelberg: Springer; 2005. p. 353-368.