The representation of human choices in business process models

Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 9 (2013) 54 – 63

Procedia Technology

CENTERIS 2013 - Conference on ENTERprise Information Systems / PRojMAN 2013 - International Conference on Project MANagement / HCIST 2013 - International Conference on Health and Social Care Information Systems and Technologies

The representation of human choices in business process models

Giorgio Bruno*

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

Abstract

Tasks and artifacts are recognized as first-class citizens in business processes but there is still no consensus on how to represent them. This paper proposes a notation, named ENTA, which allows both task-centric models and artifact-centric ones to be defined. The major contribution is the emphasis placed on human-centric processes and, in particular, on the choices under the responsibility of the participants. The consequences are a strong connection between tasks and business entities and the introduction of structures of work encompassing all the tasks that have some common input entities. An example related to an order handling business process is used to present the major features of the notation.

Keywords: business processes; tasks; artifacts; choices.

1. Introduction

A tighter integration between business processes and business entities is being pursued in current research on artifact-centric business models. The emphasis is placed on the identification of the key business entities (called artifacts) and of their life cycles, which show how the artifacts evolve over time through the execution of business operations. In contrast, activity-centric models, such those based on BPMN [1], appear to be mainly an extension to procedural programs, where the extension consists in allowing long-lived activities (human tasks

* *E-mail address:* giorgio.bruno@polito.it.

doi:10.1016/j.protcy.2013.12.006

3. Basic tasks

This section presents a number of basic tasks with the help of an example; since tasks rely on an information model, as discussed in the previous section, the features of information models are also described.

The example is a simplified order handling process and its requirements are kept to a minimum, as follows. A selling organization provides a platform (referred to as the selling platform) through which their customers can place customer orders. A customer order refers to a product type. Customer orders are handled by staff members called account managers. Each customer is served by one account manager. Account managers do not directly fulfill customer orders; instead, they must first place orders with suppliers so as to get the products needed and then they will fulfill the customer orders. In addition, they may assemble several customer orders into one supplier order, provided that the supplier is able to supply the products required. Suppliers fulfill supplier orders.

First the information model is worked out and then the models of the tasks are presented. The information model is shown in Fig. 1 and the tasks in Fig. 2.

From the requirements above, it comes out that three roles are implied, i.e. customer, account manager and supplier. Customers are served by account managers: the relationship between type Customer and type AccountMgr represents the connections indicating which account manager takes care of which customers. Type ProductType represents the products dealt with through the selling platform, and the relationship between type Supplier and type ProductType represents the connections indicating which product types are provided by which suppliers.

Relationships imply attributes (called associative attributes) in the entities involved. Such attributes refer to single entities or collections of entities depending on the cardinalities of the corresponding relationships. The names of the associative attributes may be omitted and in such cases they take the name of the entity type they refer to (with the initial in lower case), in the singular form or in the plural one depending on the cardinality of the relationship. Associative attributes are essential for navigational purposes as will be illustrated in the next sections. For example, if s denotes a certain supplier order, expression “s.customerOrders” returns the customer orders related to the supplier order.

The syntax of navigational expressions in ENTA is based on a simplified version of OCL [2].

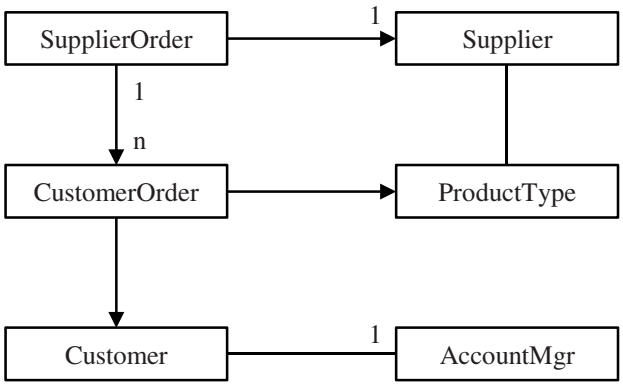


Fig. 1. The information model of the order handling process

Associations may be mandatory on one side and optional on the other; in this case, the relationship is shown as an oriented link whose origin is the entity type for which the association is mandatory. Mandatory

relationships are important in that they determine which associations have to be set when a new entity is generated. Depending on the multiplicity of the relationship, the newly generated entity will be connected to one or more entities of the destination type. On the destination side of a mandatory relationship, the default multiplicity is one; on the source side, it is n (which means zero or many).

The analysis of the requirements above leads to the introduction of the four tasks shown in Fig. 2. The features to be analyzed are the generation of entities, the identification of the performers of the tasks, and the selection of the input entities.

Task placeCO (place customer order) enables customers to place orders when they want to: its effect is the generation of a new customer order, as indicated in the post-condition "new CustomerOrder". The performer is any customer: this is indicated by keyword "any" written after the role name. The mandatory relationships CustomerOrder-Customer and CustomerOrder-ProductType indicate that the newly generated customer order has to be connected with one customer entity and one product type. The customer entity is the one related to the performer of the task; the entities representing the performers are called performer entities. The product type, instead, will be chosen by the performer during the execution of the task. The entities to be selected at run time appear in the pre-conditions of the tasks and are introduced by keyword "with".

When a task results in a new entity, the partner entities implied by the mandatory relationships related to the type of the new entity may correspond to the input entities of the task, to its performer entity or to the entities selected at run time; the correspondences are determined automatically on the basis of the entity types.

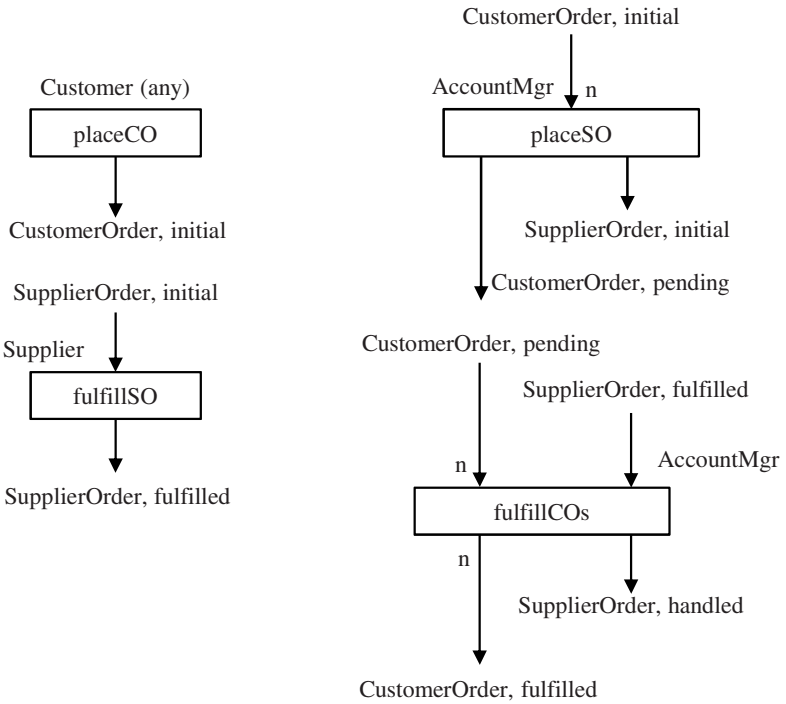
The output flows of a task indicate which results are of interest for the process; the labels show the types of the entities transmitted to the process. Task placeCO has only one output flow consisting of the newly generated customer order in the initial state.

Account managers have to assemble customer orders into supplier orders and this takes place with task placeSO (place supplier order). The performer of the task is not a generic one as it happens with task placeCO; in fact, there is no "any" qualifier added to the role name. He/she is a specific one and the specificity is determined by a relationship with the input entities. Such relationships are established by means of performer rules. Performer rule "CustomerOrder to AccountMgr" maps a customer order to an account manager through the navigational expression "customerOrder.customer.accountMgr". The expression starts from the customer order, follows the link to the customer who issued it and then reaches the account manager associated with the customer.

With task placeSO, the account manager can select a number of customer orders and one supplier so as to assemble the customer orders into one supplier order directed to the supplier. The choice of the input entities is subjected to a pre-condition which guarantees that the supplier is able to provide the products needed. The result expressed by the post-condition is the generation of a new supplier order. Due to the mandatory relationships of type SupplierOrder, the new entity will be connected to the input customer orders and to the supplier chosen by the performer.

Task fulfillSO (fulfill supplier order) acts on a supplier order in the initial state and returns the supplier order in a different state, i.e. fulfilled. The intended modifications are not specified in the requirements above, but they are represented by a different output state. The performer of the task is related to the input entity, as indicated by performer rule "SupplierOrder to Supplier".

Task fulfillCOs is an example of human task with the input entities automatically selected. In fact, it makes an account manager fulfill all the customer orders related to a supplier order that has been fulfilled.



Performer rules:
CustomerOrder to AccountMgr: customerOrder.customer.accountMgr.
SupplierOrder to Supplier: supplierOrder.supplier.
SupplierOrder to AccountMgr:
supplierOrder.customerOrders(1).customer.accountMgr.
Tasks
placeCO: pre: with productType. post: new CustomerOrder.
placeSO: pre: with supplier, customerOrders.productType in supplier.productTypes.
post: new SupplierOrder.
fulfillCOs: pre (auto): customerOrders == supplierOrder.customerOrders.

Fig. 2. Basic tasks

When a human task is meant to operate on two or more input entities, the actual choice of the input entities from the available ones may be carried out by the performer or may be made automatically. The first case will be illustrated in the next section. In the second case, human intervention is needed in the accomplishment of the task but not in the choice of the input entities. The automatic selection of the input entities implies a selection rule, which is expressed as a pre-condition. The way the input tokens are selected is an important issue the model must bear evidence of; for this reason, the pre-conditions of human tasks featuring the automatic selection of the input entities include the qualifier “auto”.

The input entities of task fulfillCOs consist of one supplier order and of all the customer orders associated with it. This is stated by the pre-condition “customerOrders == supplierOrder.customerOrders”, where the major terms, i.e. “customerOrders” and “supplierOrder”, denote the customer orders and the supplier order taken from the input flows. The type name with the initial in lower case is used to denote the input entities; the type name appears in the singular or plural form, if one entity or more than one are needed, respectively. The output flows emit the input entities with their states updated.

Performer rule “SupplierOrder to AccountMgr” indicates that the account manager in charge of a supplier order is the one dealing with the customers who placed the orders assembled into the supplier order.

4. Task-centric process models

This section describes task-centric models and introduces decisional blocks (or simply blocks), which include all the tasks having some common input links.

The requirements of the example introduced in the previous section are slightly modified as follows. Account managers may reject customer orders or may assemble a number of them into a new supplier order, which remains open so they can add additional customer orders to it. Eventually, they close the order, which is then passed to the intended supplier, who may fulfill or reject it. In case of acceptance, the account managers fulfill the related customer orders; in case of rejection, they move the customer orders back to the initial state so they will be handled as the newly generated ones.

The information model remains the same as the one shown in Fig. 1, except for the cardinality of relationship SupplierOrder-CustomerOrder, which becomes 0..1 on the SupplierOrder side because in case of immediate rejection a customer order is not connected to any supplier order.

The task-centric model of the example is presented in Fig. 3.

Basically, a task-centric model results from the concatenation of process elements, a process element being a task or a block. Two process elements are concatenated if an output link of one of them is also an input link of the other. Such concatenation links are labeled with the names of the type and state of the entities flowing through them, and have two weights: one is the output weight for the source element and the other is the input weight for the destination element. If the weight is 1, it is omitted.

The notation is based on the principle that all the tasks having some common input entities are included in the same block. As a consequence, two or more links having the same labels (type and state) and directed to different blocks are not allowed. The collection of the input entities to be acted on by a block forms the scope of the block. Scopes are disjoint: no entity may be part of two or more scopes.

Blocks may be used to represent human choices, i.e. situations in which several courses of action are possible and the choice is made by the performer. The tasks in the block are the starting points of the different courses of action.

A simple choice is characterized by one input entity; an example is given by block decideOnSO. Suppliers can accept or reject the orders directed to them. The input of the block is assumed to be the input of both tasks.

Complex choices operate on heterogeneous input entities, i.e. input entities belonging to different types; an example is provided by block handleOrders. This block enables an account manager to reject a customer order (task rejectCO), to assemble a number of customer orders into a new supplier order (task generateSO), to add additional customer orders to a supplier order (task extendSO) and to issue a supplier order (task placeSO). Since not all the input links of the block are needed by all the tasks, those needed are shown explicitly. The output links of the tasks indicate that: 1) the customer orders are put in states “rejected” by task rejectCO and in state “pending” by tasks generateSO and extendSO; 2) a supplier order is generated by task generateSO, it is put in state “placed” by task placeSO and does not change state with task extendSO.

The scope of this block consists of all the customer orders and supplier orders in the initial state; however, each performer of the block, i.e. an account manager, will receive a distinct sub-scope made up of: 1) all the customer orders that were issued by the account manager’s customers and are in the initial state; 2) all the supplier orders that were generated by the account manager and are in the initial state. The distribution of the entities is brought about by the performer rules mapping customer orders and supplier orders to account managers. They were shown in Fig. 2 and are not repeated in Fig. 3.

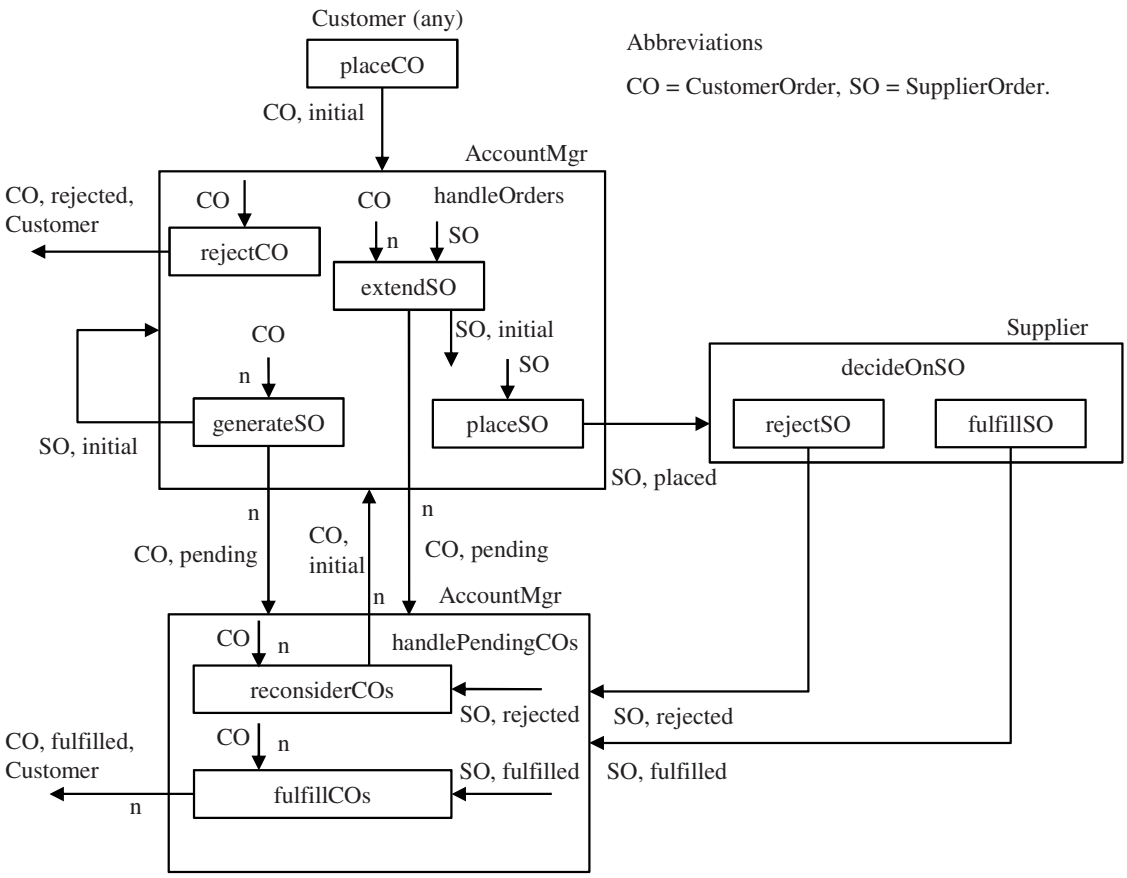


Fig. 3. The task-centric model of the order handling process

Task generateSO produces a new supplier order, which becomes part of the sub-scope for the performer, who may then select it with task extendSO or task placeSO. Task generateSO is similar to task placeSO shown in Fig. 2; pre-conditions and post-conditions are not repeated for lack of space.

A complex choice must include at least one task with heterogeneous input entities. In block handleOrders, there is only one such task, extendSO. In addition to a pre-condition, which is similar to the one of task placeSO, this task also features a post-condition stating that each customer order selected will be associated with the supplier order selected.

In block handlePendingCOs, the selection of the input entities is automatic: the customer orders are those related to the supplier order. If the supplier order has been fulfilled or rejected, the account manager fulfills the customer orders selected or moves them back into the initial state, respectively. In both cases, the supplier order is put into an implicit final state: this is due to the lack of output links related to the supplier order.

The output links that are not connected to any subsequent task represent notifications, i.e. communication sent to the intended recipients. A notification has three labels: the type of the entity which is the payload of the notification, the meaning of the notification, e.g. fulfilled, and the recipient. The recipient is equivalent to a

specific performer and is identified with the same rules. In Fig. 3 there are two notifications and their purpose is to inform customers whether their orders have been fulfilled or rejected.

5. Artifact centric process models

While the task-centric approach leads to monolithic models, those provided by the artifact-centric approach are modular in that they consist of the life cycles of the artifacts involved. The artifact-centric process model of the selling platform is presented in Fig. 4. There are two artifact types, i.e. CustomerOrder and SupplierOrder and their life cycles are shown in the same picture, for convenience.

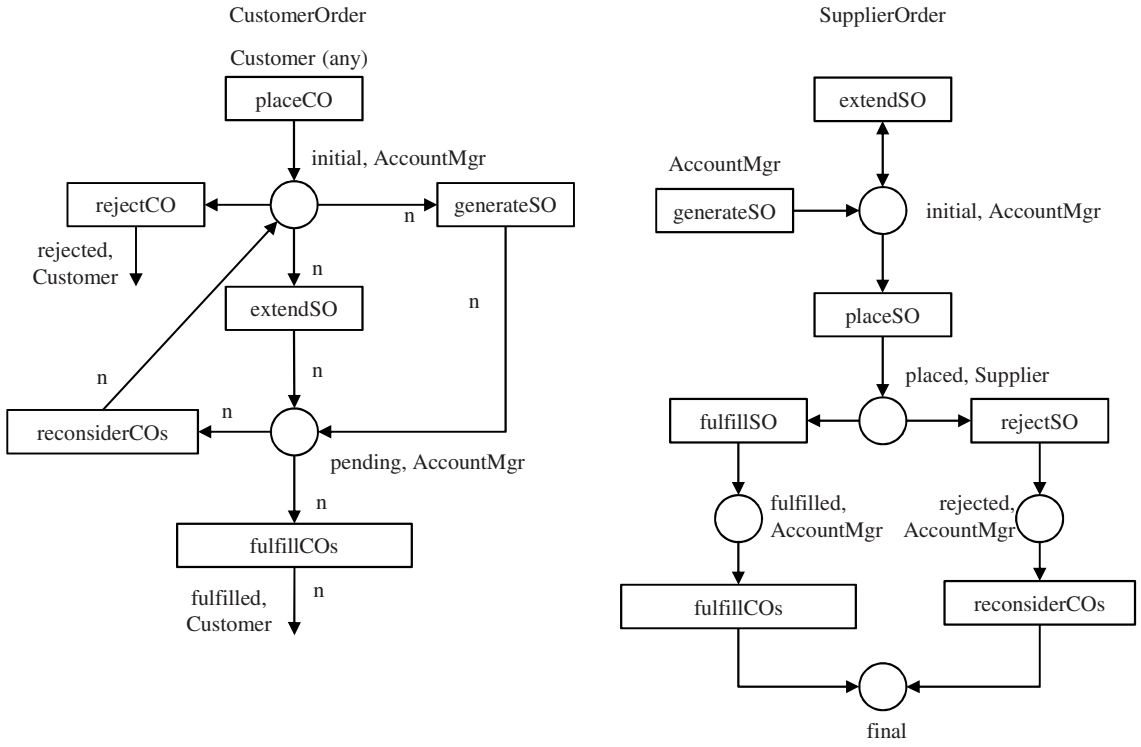


Fig. 4. The artifact-centric model of the order handling process

Artifact-centric models are state-transition models, in which states are shown as circles and transitions as rectangles. Transitions represent tasks but in contrast with task-centric models, role names are associated with states and not with tasks. The reason is as follows: since a state with more than one output transition may represent a human choice, and a choice is under the responsibility of only one role, the role is associated with the state.

A major issue in the artifact-centric perspective is the coordination of the life cycles of the entities involved. Coordination means that there are transitions belonging to different life cycles which have to be performed jointly; such transitions are called coordinated transitions. If they represent human tasks, the input states must be labeled with the same role name.

In the selling process there are four cases of coordination. In the task-centric model they correspond to the tasks generating new artifacts and/or having input entities of different types: the tasks are generateSO, extendSO, reconsiderCOs and fulfillCOs. In the artifact-centric model, coordinated transitions have identical names in the life cycles of the entities involved as they denote the same task. For example, task extendSO corresponds to two transitions affecting a number of customer orders and one supplier order.

The conventions regarding performer rules, pre-conditions and post-conditions are the same in task-centric models and in artifact-centric ones. Moreover, coordinated transitions have the same pre/post-conditions and then they need to be written only once.

6. Related work

What distinguishes knowledge-intensive processes [3] from routines is that the participants are not considered as mere resources needed to carry out tasks which are not automatable, but they can play an active role as they may take decisions which affect the control flow. Several techniques have been proposed in order to improve the flexibility of the control flow [4]. For example, with Declare [5], any task of an ad hoc sub-process may be performed as long as the mandatory constraints are not violated; with the Business Process Constraint Network approach [6], process variants may be built at run-time. However, those proposals lack a clear representation of choices in terms of links between the entities to be chosen and the activities that can be performed.

In the case-handling approach [7], a process is meant to take care of a specific entity type (e.g. an insurance claim), called the process case: the purpose is to improve the flexibility of the control flow as the process evolution depends on the state of the case and not only on the tasks performed [8].

The artifact-centric approach emphasizes the role played by the artifacts in business models. Term artifact has been introduced in [9] to designate a concrete and self-describing chunk of information that business people use to run a business. The analysis (reported in [10]) of the operations of a global financing division resulted in a high-level model consisting of 3 major artifact types, whose life cycles include 18 states and approximately 65 tasks. The major benefit is the right level of granularity, which facilitates communication among the stakeholders and helps them focus on the primary purposes of the business.

A major issue in the artifact-centric perspective is the coordination of the life cycles of the business entities involved. In the GSM (Guard-Stage-Milestone) approach [11], coordination is achieved through events and rules defined in the artifact models. In other approaches, where the relationships between artifact types are explicitly defined, coordination takes advantage of them, in particular of hierarchical relationships (e.g. COREPRO [12]). In PHILharmonicFlows [13], the life cycles of the entities are described by micro processes made up of states and transitions, while coordination is defined with macro processes consisting of macro steps and macro transitions.

An approach for discovering business entities from activity-centric models is described in [14], which also presents a mapping between activity-centric models and information-centric ones.

7. Conclusion

This paper has presented a notation for human-centric business processes: it shows the ordering of the tasks and the information flows exchanged between the tasks and the process. The information flows indicate both the input entities needed by the tasks and the output entities provided by them. Emphasis is placed on two types of choice in which participants may be involved: one is the choice of the input entities for the tasks that need more than one, and the other is the choice between a number of courses of action.

The notation named ENTA has been illustrated in its two flavors, i.e. task-centric and artifact-centric, in order to demonstrate that tasks and artifacts (i.e. business entities whose dynamics matter in the current process) are equally important and must be considered as first-class citizens in business models.

Current work focuses on how the tasks and the structures of work may be presented to their performers through suitable work lists [15].

The content of work lists is influenced by the notation adopted; with ENTA, a work list is a kind of viewpoint that participants are provided with on the artifacts they are in charge of. The viewpoint includes the options currently available and it changes on the basis of the decisions taken by participants. One of the challenges is the identification of the most expressive techniques to help participants work in this way. For example, graphical techniques showing a network of artifacts along with their states, correlations and valid options could be worked out.

References

[1] BPMN, Business Process Model and Notation, V.2.0. Retrieved March 13, 2013, from http://www.omg.org/spec/BPMN/
[2] OCL, Object Constraint Language, V.2.3. Retrieved March 13, 2013, from http://www.omg.org/spec/OCL/
[3] Marjanovic, O., Freeze, R.D., 2011. "Knowledge intensive business processes: theoretical foundations and research challenges", IEEE 44th Hawaii International Conference on System Sciences. Hawaii, USA.
[4] Nurcan, S., 2008. "A survey on the flexibility requirements related to business processes and modeling artifacts", IEEE 41st Hawaii International Conference on System Sciences. Hawaii, USA.
[5] van der Aalst, W.M.P., Pesic, M., Schonenberg, H., 2009. Declarative workflows: balancing between flexibility and support, Computer Science - Research and Development 23, p. 99.
[6] Lu, R., Sadiq, S., Governatori, G., 2009. On managing business processes variants, Data & Knowledge Engineering 68, p. 642.
[7] van der Aalst, W.M.P., Weske, M., Grünbauer, D., 2005. Case handling: a new paradigm for business process support, Data & Knowledge Engineering 53, p. 129.
[8] Künzle, V., Reichert, M., 2009. Towards object-aware process management systems: issues, challenges, benefits, in LNBIP 29, Springer, Heidelberg, p. 197.
[9] Nigam, A., Caswell, N.S, 2003. Business artifacts: An approach to operational specification, IBM Systems Journal 42, p. 428.
[10] Chao, T., et al., 2009. Artifact-based transformation of IBM Global Financing, in LNCS 5701, Springer, Heidelberg, p. 261.
[11] Hull, R., et al., 2011. Introducing the Guard-Stage-Milestone approach for specifying business entity lifecycles, in LNCS 6551, Springer, Heidelberg, p. 1.
[12] Müller, D., Reichert, M., Herbst, J, 2007. Data-driven modeling and coordination of large process structures, in LNCS 4803, Springer, Heidelberg, p. 131.
[13] Künzle, V., Reichert, M., 2011. PHILharmonicFlows: towards a framework for object-aware process management, Journal of Software Maintenance and Evolution: Research and Practice 23, p. 205.
[14] Kumaran, S., Liu, R., Wu, F.Y., 2008. On the duality of information-centric and activity-centric models of business processes, in LNCS 5074, Springer, Heidelberg, p. 32.
[15] Riss, U.V., Rickayzen, A., Maus, H., van der Aalst, W.M.P., 2005. Challenges for business process and task management. Journal of Universal Knowledge Management 0, p. 77.