RC24135 (W0612-044) December 12, 2006
Computer Science

IBM Research Report

Modeling Business Contexture and Behavior Using Business Artifacts

Rong Liu, Kamal Bhattacharya, Frederick Y. Wu
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Modeling Business Contexture and Behavior Using Business Artifacts

Rong Liu, Kamal Bhattacharya, Frederick Y. Wu

IBM T.J. Watson Research Center
19 Skyline Dr. Hawthorne, NY 10532, USA
{rliu, kamalb, fywu}@us.ibm.com

Abstract. Traditional process modeling approaches focus on the activities needed to achieve a business goal. However, these approaches often pose obstacles in consolidating processes across an organization because they fail to capture the informational structure pertinent to the business context or *contexture*. In this paper, we discuss business artifact-centered operational modeling. Artifacts capture the contexture of a business and operational models describe how a business goal is achieved by acting upon the business artifact. Business artifacts, such as Purchase Order or Insurance Claim, provide business analysts an additional dimension with which to model their business. With operational models, they can describe how a business operates by processing business artifacts and adding business value to the artifacts. This approach has been successfully employed in a variety of customer engagements. We summarize our best practices by describing nine operational patterns. Furthermore, we develop a computational model for operational models based on Petri Nets to enable formal analysis and verification thereof.

1. Introduction

Business process modeling is an essential tool for organizations to formalize and reason about how to reach business objectives. A business process model describes actions taken by business (human or system) actors using the resources of an organization to achieve a strategic or operational goal. Business process models convey business intent and serve as the basis of communication amongst a variety of stakeholders in a business, from business management, analysts, process owners, down to system developers. Enterprises of today have often grown through mergers and acquisitions which frequently lead to process redundancies and inconsistencies. Process consolidation efforts when implemented successfully can lead to significant operational improvements and cost savings. In reality, business process consolidation across a large organization is arduous. Part of the problem is cultural, i.e. disagreement over the unified process itself, as the same business process is often implemented in different ways in different organizations. This typically leads to complications in measuring efficiency of business processes and also in setting balanced incentive targets for the process owners.

One may argue that process consolidation is difficult because different stakeholders employ different process modeling languages, and that transforming from one to another is prone to semantic ambiguities and leads to skewed representations of business intent. We believe, however, that the actual problem of communicating intent using business process models is much more fundamental and independent of modeling semantics. In a variety of client engagements across various industry verticals we have noticed that standardizing the representation of business process models does not necessarily facilitate stakeholder agreement on the process. We find that traditional process models often inhibit consolidation of business operations. Business stakeholders face problems in agreeing on a unified process simply because a business process can be executed in different ways and still achieve the same goal. Agreeing on one process versus another is often a matter of taste.

We propose a different approach to understanding and representing business intent using what we call *business artifacts* (or simply *artifacts*) [4,9,13]. The idea behind business artifacts is the following. Traditional process models focus on the actions taken to achieve a certain goal (often referred to as "*verb-centric*"). Hence, business stakeholders describe their business by stating "first we do A, then B, then C, and while doing C we also do D." We propose to focus on what is acted upon, thus describing *business* operations by first identifying the *things* that matter to their business (e.g. Purchase Order, Insurance Claim), and second how these *thing*s are processed to achieve a certain goal. We call the *things* that matter to the business, the *artifacts*. Modeling business operations using artifacts is thus a "*noun-centric"* approach. In our engagements we found it relatively easy for stakeholders to agree on business artifacts. This agreement on the artifacts leads more naturally to consolidating business operations across organizational boundaries.

Business processes describe how work is coordinated to achieve operational and strategic business goals. In Hammer's framework of the Seven Dimensions of Work [7] Hammer requires that business process design respect all seven dimensions to successfully drive operational innovation. Based on our noun-centric modeling approach we re-examined Hammer's framework and separated the classification into two parts, dimensions related to information in the work context and dimensions related to the behavioral aspects of work. We refer to these two different sub-spaces as the *contextural space* and *behavioral space*. The contexture of a business is manifested in the business artifacts themselves; the behavior of a business is manifested in all the activities the business performs. In traditional process modeling, the emphasis is on the behavioral space; the contextural aspects are defined as the data attribute inputs and outputs of the work activities. In operational modeling, contextural and behavioral aspects are given equal emphasis; each work task is defined with respect to the business artifact(s) on which the task operates.

Table 1: Dimensions of Work and Process Modeling Approaches

Dimensions of Work	**Scope**
What results the work delivers	*Contexture* of business
What information the work employs	
How thoroughly the work is performed	
Who performs the work	*Behavior* of business
Where the work is performed (i.e., by which tasks)	
When the work is performed (e.g., before or after which tasks)	
Whether the work is performed	

Over the past few years we have conducted over a dozen case studies with internal and external clients. A recent case study illustrates the operational approach and the value it demonstrated in consolidating business operations. A major health insurance company was struggling to keep the database of physicians in its provider networks up-to-date. The company had to process large volumes of data coming from physicians, such as requests to be added to an insurance network, requests to update physician information (e.g. a new address or phone number), and requests to be terminated from networks (e.g. retiring or relocating). These requests were processed at numerous offices across its geographic service areas, and eventually updated a centralized provider database. Processing these requests frequently requires contacting physicians to ensure the completeness and accuracy of data, and in some cases requires verification of physicians' credentials.

The company had grown by acquisition, and each office had a different way of handling these requests. The main problem faced by this company was that some requests were taking many months to complete, delaying the processing of claims filed by those physicians. Although operations management had attempted to institute monitoring systems to identify problems, the lack of process consistency led to unintelligible measurements from the various regional offices. Management saw an opportunity to consolidate the processes into one standard set, and thus had asked each regional office to model their provider management processes. The result was a set of drawings that appeared to be very different, although all the representatives agreed that they were doing essentially the same thing. However, none of these models could be accepted as a standard one. For example, offices may have different credentialing requirements. One office requires site visits, but another may need other types of credentials. In addition, these models lacked consistent activity granularity, which complicated matters with respect to identifying fundamental business activities.

We approached the problem by asking the business stakeholders to describe the key business documents used to manage their operations. The stakeholders quickly agreed on four such documents (or artifacts): *Add Provider*, *Add Provider with Credentialing*, *Update Provider*, and *Terminate Provider*. These four business artifacts are request types that capture all information to manage the adding, updating and terminating of providers. The second step was to identify how these requests are processed. The resulting business operational models describe the lifecycle of these artifacts. These operational models formed an agreeable basis for all stakeholders in all geographies and were further used to implement a business process management system to monitor the performance of request handling.

The remainder of the paper is organized as follows. Section 2 reviews the graphical notation used for operational modeling. Section 3 gives an overview of operational patterns that have been identified by analyzing a wide range of operational models. Operational patterns are a means for modelers to quickly identify a suitable modeling construct for a given business scenario. Section 4 introduces a computational model for business operational modeling based on Petri Nets. Applying Petri net analysis ensures model correctness against several unique correctness criteria. We will describe the automatic transformation from operational models to their Petri net representations. Section 5 compares the operational modeling with other process modeling approaches. Section 6 concludes with a brief description of future work.

2. Business Operational Modeling Using Business Artifacts

The goal of operational modeling is to identify business artifacts and describe the lifecycle of artifacts from creation to archiving. A business operational modeling engagement typically consists of two main steps, first, business artifact discovery and second, modeling the lifecycle of the discovered artifacts.

Business Artifact Discovery

As any business consulting engagements, operational modeling starts with discussions with different business stakeholders to understand the overall business problem and define the modeling scope. Two types of questions are typically asked during the discussion:

(1) *Scoping:* What are you in the business of producing? What is the outcome of your process?

(2) *Discovery:* How are you measuring that you are doing what you are supposed to do?

The scoping questions help understanding the boundaries of the business operations in terms of the actual product produced by the business actors and the input required for successful production. All of this needs to be captured in information terms.

Once the scope is established the discovery phase will reveal how business stakeholders keep track of their business, i.e. information shared amongst different roles and information recorded within the established scope. Note that the discovery phase is not about the activities taken to achieve the business goal but about the information managed and maintained to produce the end product. Often we have encountered business stakeholders presenting some spreadsheets or even paper-based notebooks as their means of recording relevant information. Such information forms a basis of business artifacts. For example, in a study conducted with Bayer Pharmaceuticals, each lab head uses a specific type of document, which contains a protocol that encodes operational specifications to execute experiments, a placeholder for results and a list that shows the efficacy of a chemical applied against the biological target. We used this document as the basis for designing a business artifact called *experiment record* (or EXP for short). The details of artifact discovery can be found in [4, 13]. Next, we briefly review some fundamental properties of business artifacts and define the semantics for operational modeling prior to defining operational patterns.

A business artifact is an *identifiable, self-describing unit-of-information through which business stakeholders add value to the business* [13]. An artifact has an id which identifies itself uniquely within a given enterprise. This uniqueness property has the most important consequence that an artifact cannot be split in two. For example, the experiment record artifact can be worked on by only one role at a time, meaning the unique artifact cannot be split in two.

A business artifact is *self-describing* in the sense that its attributes are so named that its use in a given business domain is apparent. Information contained in the artifact can be listed as name-value pairs. In principle, the information model of an artifact can be modeled using any suitable information modeling approach such as an ER diagram or an XML Schema.

Modeling the Lifecycle of the Discovered Artifacts

Next, we explain the various modeling primitives used in operational models. Their graphical notations are shown in Fig. 1.

A *business task* (or simply *task*) describes the work acting upon an artifact by which *a business role adds measurable business value to this artifact*. We require a task to generate business value and hence, require an update of an artifact. This condition is necessary in our modeling approach and helps in defining the granularity of a task or the task boundaries. Imagine a simple scenario where two tasks, T1 and T2 work on an artifact consisting of ten name-value pairs. A business stakeholder could determine that completion of T1 will require update of, say attributes 2-5 of the artifact and T2 requires update of attributes 6-10. Therefore, adding business value in this case can be clearly defined by the business stakeholder who thereby determines the boundaries of a task.

Notice that the condition of an artifact update is necessary from a modeling perspective but not sufficient to truly determine the task boundaries. In the example above, the fact that attributes 2-5 are updated in task T1 is a business decision made by the analyst based on his insight into the work, and may be reflected in a condition that guards completion of T1. The approach does not prescribe this in any way, nor does it support modeling the execution of the task. The main reason to enforce artifact updates in tasks is that artifact-centered modeling is designed for creating accountability of work. Any work conducted should be traceable and hence be accounted for in a chunk of information in one or more artifacts.

Ports are the entry and exit points of tasks. We distinguish between input and output ports. A port can be associated with only one artifact type. A port lives in the context of a task and an input port can have a trigger condition that instantiates the task. A task can be triggered by a message, usually when the task is instantiated by an external (e.g. human) agent.

A *Repository* describes a waiting shelf or a buffer for an artifact. Tasks can push an artifact into a repository and pull it out of the repository. A *connector* connects an output port to an input port (task-task) or connects an output port to a repository (task-repository). Task-task connectors carry artifacts or simple messages. We support the use of messages mostly to allow for triggering of tasks by external agents, but do not encourage modeling message flows, as messages are not persistent entities and hence violate the design paradigm for accountability. Task-repository connectors carry artifacts when a task pushes an artifact into a repository. A task can either pull an artifact from a repository or read the content of an artifact in the repository.

Finally, we want to point out that one can embed conditions on ports. Conditions work as guards on input or output ports and can be conceptualized as recognizers for artifacts. In this paper, we will disregard conditions for the sake of simplicity.

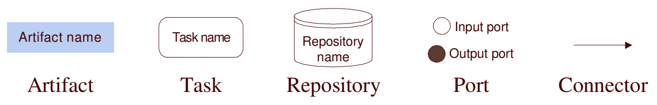

Fig. 1: Graphical Notations of Modeling Primitives

3. Building Operational Models Using Patterns

3.1. Operational patterns

During our practice with operational modeling, we designed nine operational patterns, which describe most common behaviors of business artifacts. These patterns are not exhaustive, but our customer engagements in the past five years convinced us that they are expressive enough to serve as the basic modeling constructs.

Pattern 1 (Pipeline Pattern): In a pipeline pattern, tasks are executed in sequence. An artifact is transported directly from an output port of a task to an input port of another. In this pattern, information processed by all sequential tasks is encapsulated by the same artifact. No new artifacts should be created within or from this pattern. An example of this pattern is shown in Fig. 2. In this example, task T3 "Analyze Results" is triggered right after the receipt of an artifact following the completion of task T2 "Perform Experiment". Each task updates the experiment record artifact (or EXP for short) and thus is considered to be a milestone in this artifact's lifecycle.

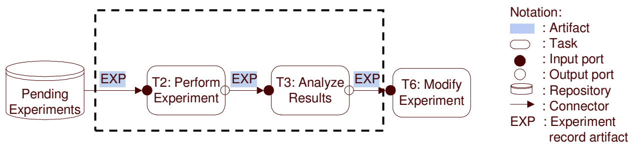

Fig. 2: Pipeline Pattern

Pattern 2 (Repository Pattern): In a repository pattern, tasks are in sequence but execution is decoupled. After being processed by a task, an artifact is sent to a repository. The repository can respond to requests for this artifact. A repository acts as a buffer when an artifact is processed by different business roles consecutively. An example of this pattern is shown in Fig. 3. The main difference of this pattern from a pipeline pattern is that task T1 "Design Experiment" does not directly trigger the subsequent task T2 "Perform Experiments". Rather, task T2 is triggered asynchronously upon accessing artifacts from the repository for pending experiments.

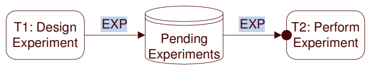

Fig. 3: Repository Pattern

Pattern 3 (Branch Pattern): A branch pattern describes more than one option to process an artifact. Fig. 4 shows an example of this pattern. In this example, the results of an experiment are analyzed by task T3 "Analyze Results". Depending on the analysis result, one of the following three tasks can be executed: (1) T4 "Clone Experiment" (i.e., the experiment is repeated to test the reproducibility of the results), (2) task T5 "Update Protocol", and (3) task T6 "Modify Experiment". These options are exclusive and only one can be chosen, as required by the fact that each artifact is a unique entity and thus cannot be split to more than one location at any time.

Pattern 4 (Convergence Pattern): In a convergence pattern, a task or a repository can accept an artifact which may arrive from different sources. In general, a convergence pattern always happens together with branch patterns. When branch patterns create multiple possible ways to process an artifact, this artifact can follow different paths and then arrive at a common task through a convergence pattern. An example of this pattern is shown in Fig. 5. In this example, the artifact follows either the sequence: task T3 "Analyze results" → task T6 "Modify Experiment", or another sequence task T3 "Analyze Results" → task T5 "Update Protocol" → task T6 "Modify Experiment". Then, there are two possible paths for this artifact to reach task T6. However, this artifact cannot take both paths concurrently, since an artifact is unique entity and cannot be divided into two.

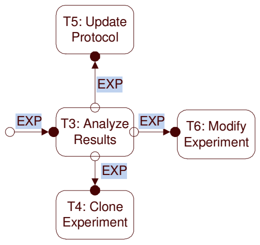

Fig. 4: Branch Pattern

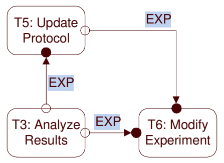

Fig. 5: Convergence Pattern

Pattern 5 (Project Pattern): A project pattern is useful in collaborative scenarios where an artifact is worked on by many role players in an arbitrary order. An example is shown in Fig. 6. In this example, task T1 first creates an experiment and stores it in a repository. Then tasks T2 "Order Raw Material" and T3 "Request Supplies" are done in any order by pulling the artifact from and replacing it in the repository. T2 and T3 can be executed in an arbitrary order but while, e.g. T2 is working on the artifact, T3 has to wait for T2 to release the artifact back into the repository. The task T4 "Start Experiment" can only be executed after both T2 and T3 completed, which would typically be realized by an appropriate guard condition on the input port of T4.

Pattern 6 (Creation Pattern): A creation pattern, as shown in Fig. 7, considers the correlation between different types of artifacts. Through a creation pattern, at least one new artifact is created. In Fig. 7, an HTS (Candidate High Throughput Screening Protocol) artifact is processed through task T1 "Design Experiment". At the same time, new EXP artifacts are created within this task. In general, these two types of artifacts are correlated in some way. For example, an EXP artifact can have references to the HTS artifact.

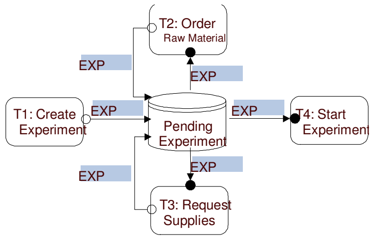

Fig. 6: Project Pattern

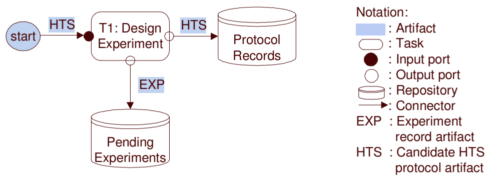

Fig. 7: Creation Pattern

Pattern 7 (Synchronization Pattern): A synchronization pattern considers the coordination between different types of artifacts. Through this pattern, a task acts on more than one artifact. The information content of one artifact is updated based on other artifacts. One example is shown in Fig. 8. In this example, the analysis of experiment results may indicate that the HTS protocol needs to be modified. Therefore, completing task T5 "Update protocol" requires two artifacts: an HTS artifact and an EXP artifact. After task T5, these two artifacts are synchronized and the updated HTS artifact is sent back to the repository. If multiple experiments are created, the HTS artifact may synchronize with each of them. Another example is shown in Fig. 9. When changes are made to a service order, a new artifact RFC (Request for Change) is created through task T2. Task T3 "Approve RFC" needs both the service order artifact and the RFC artifact as inputs. If the RFC is approved, the content of the service order is updated accordingly. In this example, synchronization happens only once.

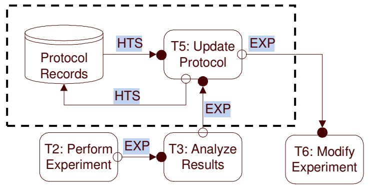

Fig. 8: Synchronization Pattern (Example 1)

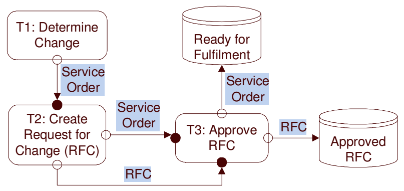

Fig. 9: Synchronization Pattern (Example 2)

Pattern 8 (Rework Pattern): A rework pattern is, in general, a loop. In this pattern, an artifact circulates in a set of tasks until an exit condition is satisfied. An example of such a pattern is shown Fig. 10. In this figure, after an experiment is performed and analyzed, if the results cannot be confirmed, the experiment needs to be repeated. Therefore, the experiment is cloned, sent to the repository "Pending Experiments", and then performed again. Note that task T6 "Modify Experiment" serves as an exit of this rework pattern.

Pattern 9 (Disposal Pattern): In some situations, an artifact may become unnecessary, for instance because of exceptions, and it drops from its lifecycle. An example is shown in Fig. 11. In this scenario, multiple experiments are created and each experiment is performed independently. When desirable results are achieved, all remaining pending experiments are disposed and sent to a repository, say "Disposed Artifacts", because there is no need for them. This pattern can be triggered by conditions, e.g. the attributes for desirable results are filled.

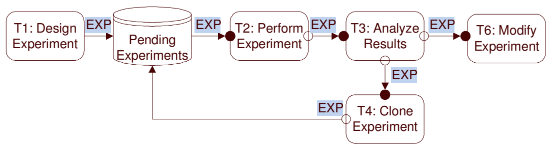

Fig. 10: Rework Pattern

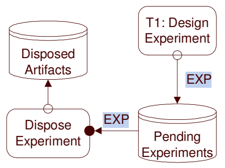

Fig. 11: Disposal Pattern

3.2. Putting Patterns Together – An Example

Having given nine operational patterns, next we continue the case study of Bayer pharmaceutical research and show how to use these patterns to build an operational model for industrializing drug discovery processes [4].

A drug discovery process starts with identifying and isolating the biological target–the biological structure associated with a specific disease. A very large number of chemical compounds that have the potential of inhibiting or neutralizing the malignant biological behavior of this target are selected during a procedure called high throughput screening (HTS). Experiments are conducted to further test chemical characteristics (ease of synthesis, solubility, reactivity, etc.) and biological characteristics (selectivity, toxicity, etc.) of these compounds. A HTS protocol gives precise and detailed instructions of performing these experiments. The protocol is evaluated and perhaps updated through a series of experiments. The target of this process is to generate an optimal HTS protocol which has maximum signal strength in the HTS apparatus in order to obtain unambiguous results. Two business artifacts, *candidate HTS protocol* (HTS) and *experiment records* (EXP), are identified.

Next, we describe several detailed scenarios of this process. Each scenario can be mapped to one or more operational patterns. Some matching patterns have been shown as examples in the previous section. We give the names of matching patterns at the end of each scenario.

(1) *Design experiment*: A lab head creates a candidate HTS protocol along with experiments. The protocol is stored in a repository and experiments are sent to the pending experiment repository (**Creation Pattern** (see Fig. 7)).

(2) *Perform experiment*: A lab technician performs an experiment from the pending experiment repository. Consecutively, the results are analyzed (**Repository Pattern** (see Fig. 3), **Pipeline Pattern** (see Fig. 2)).

(3) *Analyze results*: The lab technician and the lab head analyze the experiment results to determine one of the following options as the next step: (1) the experiment needs to be cloned and rerun; (2) the experiment needs to be modified and rerun; and (3) the protocol needs to be updated (**Branch Pattern** (see Fig. 4)).

(4) *Update protocol*: If a protocol needs to be updated, the protocol is retrieved from the protocol record repository, synchronized with experiment results, and sent back to this repository. After the update, either the experiment is determined to be complete or it needs to be modified and rerun (i.e. option (2) of Scenario (3)). Completed experiments are stored in a repository. Therefore, after result analysis, this experiment can be modified directly, or it may be modified as the protocol is updated (**Synchronization Pattern** (see Fig. 8)**, Branch Pattern, Convergence Pattern** (see Fig. 5)).

(5) *Rerun experiment*: a rerun experiment is first stored in repository "Pending experiments" and then it follows the same processing path as a new one (**Rework Pattern** (see Fig. 10)).

(6) *Prepare candidate protocol*: With experiment results, the lab head evaluates the protocol and archives completed experiments. Later, the lab head prepares to finalize the candidate protocol, requests pre-run, and stores it in a repository called "Candidate protocols". (**Synchronization Pattern**, **Pipeline Pattern**).

(7) *HTS lab*: The HTS lab retrieves the candidate HTS protocol for review. It may return the protocol and suggest further validation. Otherwise, the protocol is finalized and stored in an HTS Protocol repository (**Branch Pattern**).

(8) *Initiate additional experiments*: If further validation is needed, the lab head updates the candidate HTS protocol and creates additional experiments to the pending experiment repository (**Rework Pattern**).

It is very straightforward to formulate these scenarios after identifying their matching patterns. We can get a complete operational model shown in Fig. 12.

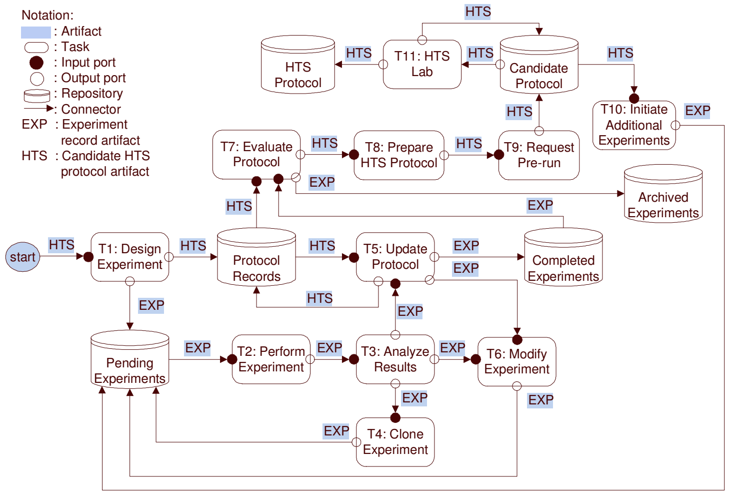

Fig. 12: Operational model of Drug Discovery Process

Operational models are targeted at users at business levels. Similar to other process models, operational models lend themselves to formal analysis, verification, and simulation to ensure successful process execution. Next, we describe how to verify an operational model through Petri nets. We start with a brief introduction to Petri nets.

4. Verifying Operational Models Using Petri Nets

4.1. Petri Net Preliminaries

Petri nets are a powerful tool for modeling the state transitions of systems in a variety of domains. A Petri net is a directed graph consisting of two kinds of nodes called *places* and *transitions*. In general, places are drawn as circles and transitions as boxes. Directed arcs connect transitions and places either from a transition to a place or from a place to a transition. *Arcs* are labeled with positive integers as their weight (the default weight is 1). Places may contain *tokens*. In Fig. 13(b), place P1 has a token, shown as a small disc. The firing rules of Petri nets are as follows [11]:

(1) A transition t is *enabled* if each input place of t contains at least $w(p,t)$ tokens, where $w(p,t)$ is the weight of the arc from p to t.

(2) The *firing* of an enabled transition t removes $w(p,t)$ tokens from each input place p of t, and adds $w(t,p)$ tokens to each output place p of t, where $w(t,p)$ is the weight on the arc from t to p.

In classical Petri nets, tokens are indistinguishable. A colored Petri net (CPN) is extended from the classical kind by tagging tokens with data values (i.e. *colors*) [8]. Moreover, in a colored Petri net, each place is associated with a type of data values (i.e., *color set*). For example, in Fig. 13(b), EXP is a color set and each color in this set stands for an experiment record artifact. In addition, each arc is associated with an arc expression specifying the number of tokens and colors of tokens removed or added to a place. In Fig. 13(b), variable "exp" means that a token from color set EXP is required to fire transition T1 and after firing, the same token is put into place P3. For simplicity, we omit the details of colors. The details of CPN can be found in [8].

4.2. Representing Operational Models as Petri Nets

We can transform operational models into colored Petri nets easily following several rules. First, each artifact type can be represented as a color set. For example, EXP in Fig. 13(b) is a color set for experiment record artifacts. Accordingly, each artifact is represented as a token with a unique color in a color set. Second, a repository is transformed into a place tagged with a color set since it stores a particular type of artifact. Third, each task is transformed into a transition and each of its output ports is represented as a place. Finally, each connector is converted to an arc and its associated artifact becomes a variable as its arc expression. Fig. 13 shows an example applying these transformation rules. However, there are three exceptions to these general rules as follows.

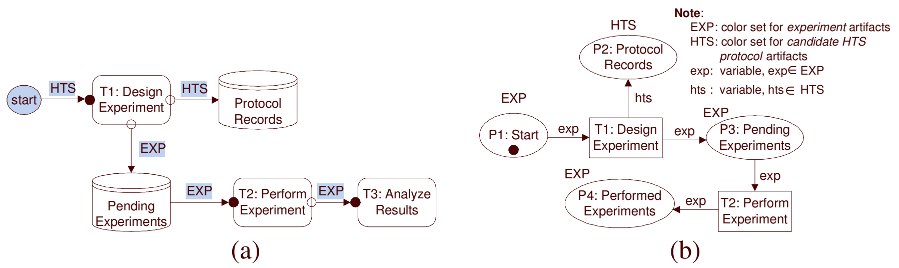

Fig. 13: An operational model and its Petri Net Representation

(1) *An output port is connected to a repository*. For example, task T1 is connected to the protocol record repository in Fig. 13(a). No Petri net representation for this port is needed. In Fig. 13(b), an arc directly connects transition T1 to place P2, which represents the repository.

(2) *Branch pattern*. The output ports of the branch task should be transformed into only one place, as shown in Fig. 14(b). After transition T5, a token is put into place P9 and it can fire either transition T6 or T12. If an output port is connected to a repository, a dummy transition, for example T12 in Fig. 14(b), is added in between two places.

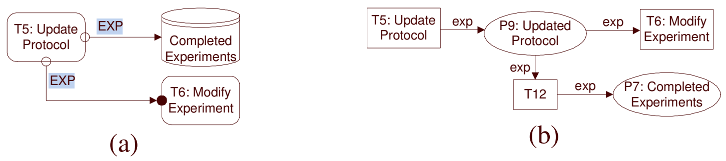

Fig. 14: Petri Net Representation of Branch Pattern

(3) *Convergence pattern*. The convergence task, for example T6 in Fig. 15(a), is duplicated so that each of its input ports belongs to one copy. Its Petri net representation is shown in Fig. 15(b).

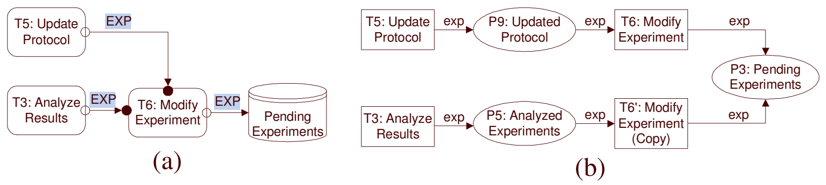

Fig. 15: Petri Net Representation of Convergence Pattern

Following these rules, the operational model of Fig. 12 can be transformed to a Petri net shown in Fig. 16. Next, we describe how to use this Petri net to analyze and verify operational models.

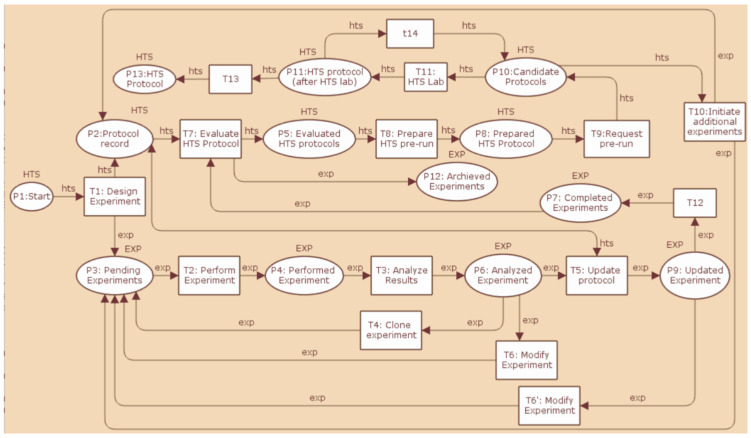

Fig. 16: Petri Net Representation of Operational model in Fig. 12

4.3. Operational Model Verification

Operational models emphasize the uniqueness of business artifacts. Therefore, the objective of verification is to ensure the following important properties of artifacts:

(1) *Persistence*: Once created, an artifact cannot disappear.
(2) *No duplicate*: A business artifact is a unique entity and cannot be duplicated.
(3) *No split*: A business artifact can be at only one place at a time.
(4) *Reachability*: It is possible for an artifact to reach any of its states (i.e. tasks or repositories associated with this artifact).

These properties can be verified using a Petri net reachability graph [11]. A reachability graph shows the development of markings of a Petri net from an initial marking. A *marking* is denoted by a vector M, where $M(p)$ denotes the tokens in place p. For example, suppose the initial marking of Fig. 13 is: M_0(P1)=1`"hts1" and $M_0$(P2) = $M_0$(P3) = $M_0$(P4) = 0 as only P1 has a token (denoted as "1`") with a color "hts1". After transition T1 fires, a new marking, say M_1, is generated: M_1(P2) = 1`"hts1", $M_1$(P3) =1`"exp1" (suppose a new experiment "exp1" is created), and M_1(P1) = M_1(P4) = 0. [11] gives an algorithm for generating reachability graphs.

Since an operational model describes an artifact's lifecycle and all artifacts of the same type have exactly the same lifecycle, to verify the above artifact properties, imagine that we put one token of each color set (i.e., one artifact of each type) into its Petri net representation to see its behavior. If the operational model is correct, for every marking *M* in the reachability graph of this Petri net, there is no more than one token in any place *p*, i.e., this Petri net is *safe* [11]. Moreover, since an artifact can be at only one state at a time, the maximum number of markings is $\prod n_i$, where n_i is the number of states of artifact type *i*. Therefore, the verification is very efficient. The artifact properties can be translated in terms of reachability graphs as shown in Table 2.

Table 2: Verifying Artifact Properties Using Reachability Graphs

Artifact Properties	**Properties of Reachability Graph**
Persistence	Any token in a marking must exist in all of its subsequent markings
No duplicate	Any place of a marking can have no more than one token
No split	Places of a marking cannot have tokens with the same color
Reachability	Each transition is fireable, and for each place *p*, there exists at least a marking *M* such that $\|M(p)\|=1$.

Therefore, using the algorithm in [11], we can get a reachability graph, shown in Fig. 17, for the Petri net of Fig. 16. Note, in this reachability graph, we use a simple marking notation. For example "P1, P2" denote the marking where only places P1 and P2 each have a token with the right color. Also, the transition from node "P10, P12" to "P2, P3" shows a new experiment is initiated by task T10. The existing experiment has reached its final state and is removed. Minor modifications have been made to this algorithm to accommodate such a situation. Obviously, we can verify that these four artifact properties are guaranteed in this operational model.

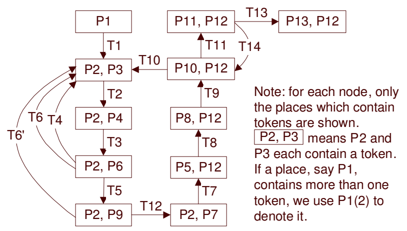
P1
T1
P2, P3
T2
P2, P4
T3
P2, P6
T5
P2, P9
T12
P2, P7
T7
P5, P12
T8
P8, P12
T9
P10, P12
T10
T11
P11, P12
T13
P13, P12
T14
T4
T6
T6'
Note: for each node, only the places which contain tokens are shown.
P2, P3 means P2 and P3 each contain a token.
If a place, say P1, contains more than one token, we use P1(2) to denote it.

Fig. 17: Reachability Graph of Petri Net Representation of Fig. 16

In addition, Petri nets allow simulation and other formal analyses. Simulation can be done using CPN Tools [14]. Moreover, we can perform theoretical analysis to study the performance of an operational model, such as artifact lifecycle length and throughput. The detailed analysis techniques are outside the scope of this paper.

5. Discussion and Related Work

Business operational modeling incorporates the contexture of a business as a first-class modeling primitive as manifested in business artifacts. The behavior of a business, described in the context of artifacts, models how business roles process artifacts to produce measurable business results. We found that the operational modeling approach can reduce the complexity of business problems significantly for two reasons. First, the artifact dimension tends to be manageable because there are typically just a few artifacts in any given business process. For example, we analyzed how IBM manages financial contracts, from the signing of a deal through creating and managing the contract. In spite of the complexity of IBM's business, we identified only 7 distinct business artifacts. Second, complexity of business processes is often exacerbated by a lack of guidelines for the granularity of activities. Standard process modeling approaches provide no criteria that prevent business analysts from including execution details. This lack of guidelines also leads to inconsistent granularity of activities, with some very detailed activities and some large chunks of the process represented by a single activity. Business artifacts, on the other hand, provide a context for the scope of a business task, which should be a distinct functional entity that updates one or more artifacts and produces a measurable business result.

We have shown the fundamental difference between operational modeling and traditional activity-centric process modeling, mainly workflow approaches with a focus on control flows, [1,3,6] throughout this paper. Besides control-flow based workflows, recently, data-flow driven workflows have attracted increasing attention [10, 15]. The data-flow driven approach concerns the dependencies between data used by activities and derives control flows based on such dependencies. However, often the dependency information is insufficient for the generation of process models [10]. Moreover, it could be very difficult to determine the dependencies of a large number of data objects. For example, [10] proposes a data-driven approach which defines data objects and their lifecycles, specifies their dependencies, and then derives control flows from data dependencies and lifecycles. However, the resulting processes may be complicated when considering a large number of data objects. Operational modeling provides a framework to group data logically into a few unique and self-describing entities and the modeling complexity is then greatly reduced.

Accordingly, operational patterns are also different from workflow patterns [2], which describe styles of control flows in workflow tools. Operational patterns should be understood as the styles of artifact behavior. We introduced several operational patterns, such as the creation and project patterns, that are unique in the context of operational modeling. Also, it is easy to understand that some workflow patterns such as "parallel split" cannot happen in operational models since an artifact is undividable.

6. Conclusion and Future Work

In this paper, we presented business artifact-centered operational modeling. Artifacts capture the contexture of business, and operational models describe how a business goal is achieved by acting upon the business artifacts. We also showed how this approach fundamentally differs from traditional activity-centric process modeling.

This approach has been tested by a number of successful customer engagements. We summarized our best practices as nine operational patterns. These patterns can serve as basic constructs for developing operational models. Further, we transform operational models into colored Petri nets and verify the correctness of operational models through Petri net reachability analysis. Using operational models, a company is able to develop business process management IT solutions that are well aligned with its business intent. The MDBT Toolkit [15], which automatically generates IT solutions from operational models has been developed and tested in practice. In addition, a verification and analysis tool based on Petri nets is also under development.

As a future exercise, we plan to explore how operational models enable organizations to develop solutions based on Services-Oriented Architecture (SOA) [5]. Today's enterprises recognize the importance of SOA but struggle with the methodologies to implement SOA solutions. Operational models provide insights into defining business relevant services at the appropriate level of granularity.

References

1. Aalst, W.M.P. van der, “The application of Petri nets to workflow management,” *The journal of Circuits, Systems and Computes*, 7(1):21-66, 1997.
2. Aalst, W.M.P. van der, Hofstede, A.H.M. ter, Kiepuszewski, B., and Barro, A.P. “Workflow patterns,” *Distributed and Parallel Databases*, 14(3):5-51, July 2003.
3. Basu, A. and Kumar A. "Workflow Management Issues in e-Business," *Information Systems Research* 13(1): 1-14, 2002.
4. Bhattacharya, K., Guttman, R., Lyman, K, Heath III, F. F., Kumaran, S., Nandi, P., Wu, F., Athma, P., Freiberg, C., Johannsen, L. and Staudt, A. “A model-driven approach to industrializing discovery processes in pharmaceutical research,” *IBM Systems Journal*, 44(1):145-162.
5. Ferguson, D. F. and Stockton M. L. "Service-oriented architecture: programming model and product architecture," IBM Systems Journal archive 44(4) 753 – 780, 2005.
6. Georgakopoulos, D. and Hornick, Mark “An Overview of Workflow Management From Process Modeling to Workflow Automation Infrastructure,” *Distributed and Parallel Database*, 3:119-153, 1995.
7. Hammer, M. "Deep Change: How Operational Innovation can transform your Company", *Havard Business Review*, page 84-93, April 2004.
8. Jensen, K. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, Volume 1, Springer-Verlag, Berlin Heidelberg, 1996.
9. Kumaran, S. “Model Driven Enterprise,” *Proceedings of Global Integration Summit 2004*, Banff, Canada, 2004.
10. Müller, D. and Reichert, M.U. and Herbst, J. "Flexibility of Data-driven Process Structures," In *Business Process Management Workshops, BPM 2006 International Workshops*, Lecture Notes in Computer Science 4103. pp. 179-190, Springer Verlag.
11. Murata, T. “Petri Nets: Properties, Analysis and Application,” In *Proceedings of the Institute of Electrical and Electronics Engineers*, 77(4): 541-580, April 1989.
12. Nandi, P. and Kumaran S. "Adaptive Business Objects - A new Component Model for Business Integration," In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, Miami, USA (ICEIS 2005), C. S. Chen, J. Filipe, I. Seruca and J. Cordeiro, Eds, pp. 179-188, Miami, USA, 2005.
13. Nigam, A. and Caswell, N. S. “Business artifacts: An approach to operational specification,” *IBM Systems Journal*, 42(3):428-445, 2003.
14. Ratzer, V. A., Wells, L., Lassen, M. H, Laursen, M., Qvortrup, F. J., Stissing, S. M., Westergaard, M., Christensen, and S., Jensen, K. “CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets,” *Applications and Theory of Petri Nets 2003*, Lecture Notes in Computer Science, 2679: 450 - 462, Springer-Verlag, 2003.
15. Sun, S. Zhao, J. L. and Nunamaker, J. "On the Theoretical Foundation for Data Flow Analysis in Workflow Management", *2005 Americas Conference on InformationSystems*, 2005, Omaha, Nebraska, USA.