A Framework for Realizing Artifact-Centric Business Processes in Service-Oriented Architecture

Kan Ngamakeur, Sira Yongchareon, and Chengfei Liu

Faculty of Information and Communication Technologies
Swinburne University of Technology, Victoria, Australia
{kngamakeur,syongchareon,cliu}@swin.edu.au

Abstract. Over the past few years, the artifact-centric approach to workflow modeling has been beneficially evidenced for both academic and industrial researches. This approach not only provides a rich insight to key business data and their evolution through business processes, but also allows business and IT stakeholders to have a single unified view of the processes. There are several studies on the modeling and its theoretical aspects; however, the possible realization of this approach in a particular technology is still in its fancy stage. Recently, there exist proposals to achieve such realization by converting from artifact-centric model to activity-centric model that can be implemented on existing workflow management systems. We argue that this approach has several drawbacks as the transformation, which is unidirectional, poses loss of information. In this paper, we propose a framework for the realization of artifact-centric business processes in service-oriented architecture achieving a fully automated mechanism that can realize the artifact-centric model without performing model transformation. A comprehensive discussion and comparison of our framework and other existing works are also presented.

1 Introduction

To meet the challenges of globalization, business processes demand for technologies that can support more efficient and economical way of automation and collaboration. Promisingly, Service-Oriented Architecture (SOA) shows itself as technology enabler that can support such needs. During the recent years, an artifact-centric approach to business process modeling has been introduced as a propitious paradigm that lends itself well to SOA design principle and model-driven architecture (MDA) design concept [1, 3, 5, 7, 12]. This approach has been evidenced in both academic and industrial researches where it not only provides higher level of flexibility of workflow enactment and evolution, but also facilitates the process of business transformation and helps communicating the business intent for consolidating business operations across organizations [1, 2, 3, 4, 5, 8, 9]. In essence, the approach has a central focus on defining key business entities, so called "business artifacts", which are evolved and manipulated within a process. The controlling mechanism that governs the whole process can be implemented by business rules. So far, there have been several studies on the modeling and theoretical aspects of the artifact-centric approach; however, its

S.-g. Lee et al. (Eds.): DASFAA 2012, Part I, LNCS 7238, pp. 63–78, 2012.

realization and system implementation, especially under SOA and MDA environment, is still in its fancy stage.

One possible and practical approach for realizing the artifact-centric business processes is by transforming an artifact-centric model to a conceptual flow model, which is an activity-centric (control-flow) model. The conceptual flow model is, then, mapped into an executable workflow, e.g., BPEL [5]. The advantage of using this approach is an ease of implementation as workflow technologies and standards based on the traditional model have been developed, e.g., in [11, 12]. In spite of such good point, we argue that this approach has several drawbacks as the transformation, which is unidirectional, poses loss of information. By converting the model, business rules are degraded into control flows; therefore, it is difficult to track and manage the rules based on the converted model. The flexibility of the process is also reduced as business rules are not available to be modified at run-time. Another possible approach is to realize the artifact-centric process model directly without converting the model. This can be considered as more efficient and automatic approach for realizing the model. We claim that the latter approach overcomes the issues of the former approach. In this paper, we propose a framework for the realization of artifact-centric business processes. The framework consists of artifact-centric workflow model and a mechanism that can automatically realize and execute the model under the service-oriented environment. We also provide detailed discussions on technical issues and challenges of our realization framework as well as the comparison with existing activity-centric workflow systems.

The remainder of this paper is organized as follows. Section 2 presents an artifact-centric approach to process modeling. Section 3 discusses the realization framework for artifact-centric business processes. Section 4 shows implementation and evaluation of our framework. Section 5 discusses and reviews the related works. Finally, the conclusion and future work are given in Section 6.

2 Artifact-Centric Approach to Business Process Modeling

In this section, we introduce an example of business processes to illustrate that we can identify business artifacts and use them to construct an artifact-centric business process in order to use it for analysing and capturing the requirement of our prototype system. In the artifact-centric approach, a business process can be constructed using business artifacts. An artifact stores its business relevant information and its lifecycle. The state transition of artifacts is achieved by a service and is controlled by a set of business rules. Our example of business process is adapted from a simple online ordering process. The process starts when a customer places an order including billing information through a web site. Then the order is sent to a manufacturing factory where the ordered product is assembled, tested and packaged. Finally, the product is shipped to the customer. After we examine this process, several business artifacts are identified. Fig. 1 shows *data model* and *lifecycle model* of key business artifacts involved with this business process. For each artifact, the data model represents its data attributes, while the lifecycle model represents its state transition of the artifact.

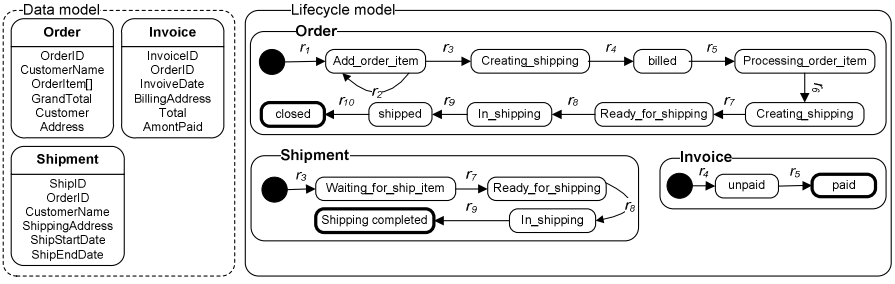

Fig. 1. Artifacts and their lifecycles in product ordering processes

We can see that this process consists of three classes of artifacts: *invoice*, *shipment* and *order*. Apart from the artifacts, in the lifecycle model we can see two components that are essential for constructing a complete business process – those are *services* (a.k.a. tasks) and *business rules*. A service is used to make change on artifacts. An association between services and business artifact(s) is specified by using business rules as to describe on what condition such service is performed on the artifact(s). More details are described in Section 3.2. Based on initial concept of artifact-centric business processes, we analyzed the problem domain to understand basic requirements that needed to be addressed in our framework. Here, we summarized our requirements into three points listed in following paragraphs.

– **A Formal Process Definition of Artifact-Centric Business Process (ACP)**

In artifact-centric approach, the conceptual model of an artifact-centric business process is defined in a declarative manner. The conceptual model provides a high level specification of a business process execution. Normally, it is used to communicate business intents between stakeholders but it can't be executed by a computer system. In order to realize the conceptual model of an artifact-centric business process, we need to develop a process definition that contains all concrete details required by a process execution.

– **A Process Deployment and Execution**

The process deployment has to be developed in such a way that it can parse the model definition, map parsed data to predefined classes, and deploy a process in a web service environment. When a client invokes the deployed process, the process and other related (e.g., artifacts, services, rules) instances need to be created. The concept of executing and managing these instances for artifact-centric business processes are new and relatively challenging. This is because the core constructs of the artifact-centric process model differ from those of the traditional activity-centric model.

– **Business Rule Definition and Evaluation**

In the traditional approach, business logics are defined explicitly using control flows and activities. In contrast, in artifact-centric approach, we use business rules to define an association between artifacts and services. Each rule describes which service is invoked and which artifact(s) is changed under what conditions. This requires an investigation of how rules can be defined in the most expressive and effective manner. In the implementation, the integration of a suitable rule engine to our system to handle the artifact-centric process execution is also challenging.

3 ACP Realization Framework

In this section, we illustrate our framework for automated realization of artifact-centric business processes. The detailed technical discussion on proposed system architecture is also presented. It is quite easy to comprehend the artifact-centric business process model at the conceptual level from our motivating example since it was designed to incorporate information and behaviour aspects of a business process. As a result, we can convey and communicate business intent among a variety of stakeholders. As already introduced, two approaches are observed. The first approach is to convert the artifact-centric model into a conceptual model in a procedural manner. The good side is that the artifact model can be easily to be implemented using traditional workflow technologies. Its drawback is that the flexibility of the artifact model and data may lose in the model conversion. On the contrary, our approach directly realizes the artifact-centric process model. Here, we propose an ACP realization framework based on the direct approach, and it is illustrated in Fig. 2.

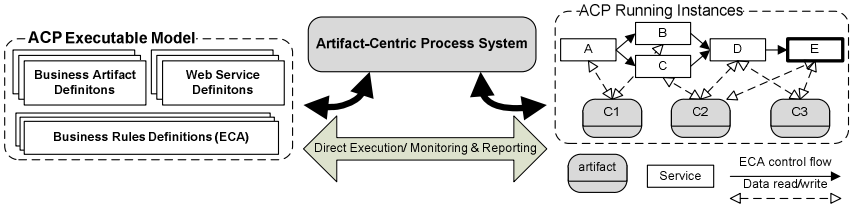

Fig. 2. ACP realization framework

In our framework, we aim at fully automated realization from ACP definition to its execution. This framework does not require additional model transformation (from ACP model to workflow executable model) nor require backward mapping mechanism to validate the running instances with the original ACP model. The framework contains only ACP executable model and the automated realization mechanism that can directly execute such process definitions. In the task-based model, data is defined separately at later time (in most cases after the task has already been defined). While in run-time, workflow systems do not realize the relationship between the current stage of task execution and the state of the data or artifacts that being manipulated. This poses a technical problem when attempting to discover the correspondence and to track run-time instances of those running artifacts directly in the ACP model. In our approach, instances of process, services, and artifacts being manipulated can be directly reported regarding their ACP model. Monitoring a progress of a particular business process can be efficiently achieved at both artifact and process levels. We can see that such direct monitoring and reporting are more efficient as an additional reverse mapping from instances of activity-centric model to artifact-centric model is not needed.

3.1 Artifact-Centric Business Process Model

Here, we introduce an *artifact-centric business process model* (*ACP model*) that has been proposed in our previous work [9, 10]. Our ACP model consists of sets of

artifact classes, *services*, and *business rules*. An artifact, which is a key business entity involved in business processes, contains its relevant attributes and many finite processing states. Let $Z = \{C_1, C_2 \ldots, C_x\}$ be a finite set of artifact classes that are used in a particular process. Each *artifact class* $C_i \in Z$ is defined as a tuple (A, $\boldsymbol{s^{init}}$, S, $\boldsymbol{S^f}$) where set $A = \{a_1, a_2, \ldots, a_y\}$, and each $a_j \in A$ is a name-value pair attribute; set $S = \{s_1, s_2, \ldots, s_z\}$ contains the possible states of the instances of class C_i; $\boldsymbol{s^{init}}$ is the *initial* state, and $\boldsymbol{S^f} \subseteq S$ is a set of its *final* states. A *service* is a task that is used to perform read/write operations on some artifact(s), and it is denoted as $v(C_1, C_2, .., C_y)$ where $C_1, C_2, .., C_y$ are artifacts that are read/updated by service v. A *business rule* is used to associate service(s) with artifact(s). It is defined in a *Condition-Action* style to describe on what *pre-condition* a particular service is executed, and on what *post-condition* after performing such service must satisfy. A *business rule*, denoted as r, is a tuple (λ, β, v) where λ and β are a *pre-condition* and *post-condition*, respectively; v is a service that performs read/update operations on the attributes and the processing states of some artifacts in schema Z. We restrict both pre- and post-conditions to be expressed by a conjunctive normal form. This form can contain two types of proposition over schema Z: (1) *state proposition* (by *instate* predicate) and (2) *attribute proposition* (by *defined* predicate and scalar comparison operators). We write *defined*(C, a) if attribute $a \in C.A$ of artifact of class C has a value; and *instate*(C, s) if state $s \in C.S$ of artifact of class C is active. Initially, *instate*(C, $\boldsymbol{s^{init}}$) implies $\forall x \in C.A$, $\neg$*defined*(C, x). A complete set of business rules defined for a particular process model specifies the control logic (named ECA flow) of the whole process from the beginning to the termination of the process. Table 1 shows an example subset of business rules that are used in our product ordering process.

Table 1. Example of business rules

r1: *Customer requests to make an order O*	
Pre-condition	*instate(O,init) ∧ defined(O,OrderID) ∧ defined(O.CustomerName) ^ defined(O.CustomerAddress)*
Service	*createOrder(O)*
Post-condition	*instate(O,Add_OrderItem) ^ defined(O.OrderID) ^ defined(O.CustomerName) ^ defined(O.CustomerAddress)*
r2: Create Shipment S for an order O	
Pre-condition	*instate(O,Add_Order_Item) ^ instate(S,Init) ^ defined(O. GrandTotal) ^ O.GrandTotal>0 ^ defined(S.ShipID) ^ defined(S.OrderID) ^ defined(S.ShippingAddress)*
Service	*createShipping(S,O)*
Post-condition	*instate(O,Create_Shipping) ^ instate(S,waiting_for_Ship_Item) ^ defined(S.CustomerName) ^ defined(S.ShippingAddress) ^ defined(S.ShipID) ^ defined(S.OrderID)*
r3: Create Invoice I for an order O	
Pre-condition	*instate(I,Init) ^ instate(O,Creating_Shipping) ^ defined(I.InvoiceID) ^ defined(I.OrderID) ^ defined(I.BillingAddress) ^ defined(I.InvoiceDate) ^ defined(I.Total) ^ I.Total= O.GrandTotal*
Service	*createInvoice(I,O)*
Post-condition	*instate(V,Unpaid) ^ instate(O,Billed)*

3.2 ACP Executable Model

Now, we propose to use a serializable and executable version of the ACP model based on the ACP Model definitions described in Section 3.1. Our artifact-centric executable process model is defined by using XML, and it consists of three definitions: *artifact definition*, *business rule definition*, and *service definition*, as shown in Fig. 3. It contains implementation details required by a system to execute a particular business process and they are used for creating running instances.

```
<artifacts>
    <artifact name = "Order">
        <attributes>
            <attribute name = "orderID" structure = "pair" type = "String" />
        </attributes>
        <states>
            <state name = "start" type = "init" />
            <state name = "open_for_item" />
            <state name = "end" type = "end" />
        </states>
    </artifact>
</artifacts>
```

```
<services>
    <service name = "crteateOrderService"
     inputMessage = "createOrderRequest"
     outputMessage ="createOrderRespone"
     operation = "createOrder"
     port = "createOrderSOAP"
     location = "http://localhost:8080/axis2/services/createOrder?wsdl"
     namespace = "www.swin.edu.au" />
</services>
```

```
<businessrules>
    <rule name = "create_order">
        <onEvent type = "inputMessage"/>
        <precon>
            <and>
                <atom type ="state" artifact="Order" id="Order1" value = "init" />
                <atom type="attribute" artifact="Order" id="Order1" attribute="item_quantity" op="==" value="0"/>
                <atom type="input" attribute="a" op=">" value="1"/>
            </and>
        </precon>
        <do>
            <invoke type = "internal" operation = "createOrder" service = "createOrderService" >
                <mapping>
                    .....
                </mapping>
                <transitions>
                    <transition artifact = "order" id = "order1" fromState = "init" toState = "open_for_item" / >
                </transitions>
            </invoke>
        </do>
    </rule>
</businessrules>
```

Fig. 3. Artifact, Service, and Business rule definitions

- **Artifact Definition** composes of a set of attributes and states. An attribute definition provides details of business data (<name>, <type>, and <structure>) that can be stored in each attribute of a particular artifact, such as attribute Name, data type and data Structure. A state definition provides details of each state (<name>, <type>) in a particular artifact life cycle, such as name of state, initial state and final state.
- **Business Rule Definition** is used to define an ECA-like rule description. This rule consists of event, precondition and action (<onEvent>, <precon>, and <do>). Element <onEvent> provides details of which event can trigger a particular rule. Element <preCon> is a condition that needs to be satisfied in order to take a further action. Element <do> is a task or service that needs to be invoked. Element <invoke> provides service name and operation name of a designated web service. Moreover, Mapping rules and transition rules (<map>, <transition>) are defined in this part as well. Mapping rule provides details of data mapping between artifact and message. The transition rule is used to control state transition for each artifact involving in a step of process execution.
- **Service Definition** defines concrete details of a web service. This definition provides information that is necessary for a service invocation, such as service name, operation, WSDL location, and port. In this paper, we consider only inputs and outputs of a web service not including its behaviours.

Due to the fact that the current web service technologies do not support an artifact as an input of a web service. To address this issue, an internal data mapping mechanism is required in order to correlate the passing messages (input/output) and their corresponding data attributes of artifacts. We use a mapping rule to map data between artifact and SOAP message; therefore we introduce mapping rules, as shown in Fig. 4 and Fig. 5 into our framework. We consider that there are two types of data mapping in our framework, which are mapping from message to artifacts and from artifacts to message. We include mapping type to indicate a direction of mapping. The rule is also included details of a source (<from>) and a destination (<to>) for mapping between artifact and message. These two elements contain information that helps the system to locate corresponding artifact's attribute or message's part to be map when a web service is invoked.

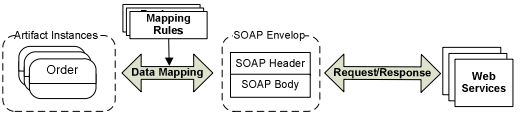

Fig. 4. Data mapping between SOAP Message and Artifacts

```
<map type = "MessageToArtifact">
    <copy>
        <from message = "calTotalMessage" part = "customerID"/>
        <to artifact = "order" attribute="orderID"/>
    </copy>
    <copy>
        <from message = "calTotalMessage" part = "numberOfItem"/>
        <to artifact = "order" attribute="quantity"/>
    </copy>
</map>
```

```
<map type = "ArtifactToMessage">
    <copy>
        <from artifact = "order" attribute="orderID"/>
        <to message = "calTotalMessage" part = "customerID"/>
    </copy>
    <copy>
        <from artifact = "order" attribute="quantity"/>
        <to message = "calTotalMessage" part = "numberOfItem"/>
    </copy>
</map>
```

Fig. 5. Example of mapping rules

3.3 Run-Time ACP Instances

During a process execution, we need to keep track the status of a running process. This allows the system administrator to inspect the status during a runtime and after a completion of a process execution. We classify instances of ACP into four following types where each of which corresponds to individual component of ACP model.

- *Process instance* – When the process is enacted then the system initially creates process instance. Once a process instance is created, it will be given its name corresponding to executed business process and will be given an identifier key. Process instance acts as a container to store other running instances, which are artifact instance, rule instance and service instance.
- *Artifact instance* – In a process, an instance of particular artifact class can be created at the time the process is initialized or after service invocation (that performs a creation of artifact). Newly created artifact instance will be populated with artifact definition data and will be given an artifact identifier key. This instance serves a purpose of storing information including business data and lifecycle during each step of business process execution. Thus, it is a key to indicate progress of a running process.

- *Service instance* – Service is instantiated when it is invoked (as defined by the action in a business rule). It not only stores service invocation information defined in a service definition but it also captures input/output message data as well as timestamp.
- *Rule instance* – An instance of business rule is created when the rule is triggered by event and its pre-condition holds. A rule instance provides information regarding decision making. By inspecting this instance, we will know which rule is fired, what time rule is fired, and what data triggered rule firing.

Based on the above types of instances of ACP, we can gather the complete execution traces by recording every instance type on the log records. These records of a particular process permit the real-time (direct) monitoring of the process and its components without the reverse mapping, which is required for the existing model transformation realization approach, i.e., covert ACP model to task-based model and run it on existing workflow system. It is worthwhile mentioning that with our framework, business rules can be modified/removed/added at run-time while still able to reflect its process model. At run-time, we allow our ACP system to keep different versions of business rule for a particular process model by storing the mapping of every version and its original version. This feature enhances the system ability to be able to track/monitor different versions of (process and rule) instances of the same process. We also claim this feature to be one of the advantages of our realization framework compared with the existing approach.

4 Implementation and Evaluation

4.1 ACP System Architecture and Its Components

In this section we show our proposed architecture of the artifact-centric process system (*ACP System*), as illustrated in Fig. 6. This system architecture ensures that we can address those requirements from the previous section. We adopted the concept of the event driven architecture and service-oriented architecture.

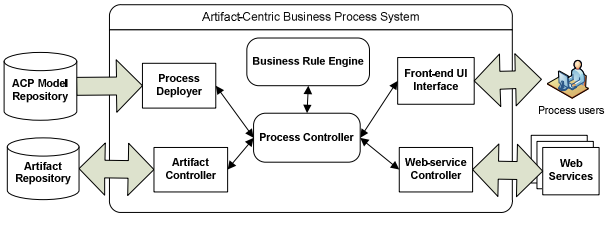

Fig. 6. ACP System

Here, we describe each ACP system's component in more details.

– *Process Deployer* is used to deploy ACP Model definition file. The definition file will be parsed to generate running instances of a particular business process.

- *Business Rule Engine* provides a rule evaluating functionality. For any change in a running process, a rule engine will evaluate an instance in order to determine the next possible action that will be undertaken.
- *Process Controller* is used to manage instances of a process based on rule engine's command. Once the process controller receives a command from rule engine to start a new process execution, it will use a process factory to create running process instance. The factory will identify the corresponding instance and create it. The created process instance will be given an id for identification purpose. The process controller can issue a command to an artifact controller to update artifact instances or to web-service controller to invoke web services. For any changes in a running process, the process controller will consult the rule engine to perform the next possible action.
- *Web Service Controller* is used to invoke web services to process artifact data. To invoke a web service, the controller creates a soap message corresponding to message definition in WSDL. The artifact data is mapped to message data using a mapping rule. Finally, the request message is sent to a designated web service. Once a response message is returned, the service controller processes the message, and returned message data is mapped to corresponding artifact attribute.
- *Artifact Controller* is used to manage and update artifact (which is stored in external repository). After the service controller receives a response message from a web service, a data mapper will extract the message. Then the artifact controller uses such data to update corresponding artifacts.
- *Front-end UI Interface,* proposed in our previous work [10], is used to manage web-based interactions between ACP system and users, which includes automatic generation of web pages and receiving/responding via web form interfaces.

4.2 Run-Time Execution

Here, we discuss operations of our ACP system in more details including instantiation of a running instance, operation of rule engine and how each component works together to coordinate a process execution.

- **Creation of a Running Instance**

The ACP system has the component so called process factory to create running instances from an ACP definition. Once a process execution has started, the process controller will call a process factory method to create a process instance and also other running instances. The factory will identify a correct deployed process and instantiate corresponding instances. During instantiation of running instances, implantation data stored in the definition file will be parsed and mapped to corresponding instances. Upon receiving a process instance, the process controller will register the process instance in order to be able to keep track processes that are currently running.

- **Integration of Rule Engine**

In Artifact-centric business process, business rules are main mechanism to control interaction between artifacts and services. The rule engine is integrated into our system to provide functionality for evaluating business rules. The rule engine will be activated once it receives an internal event generated by the process controller. During

the activation, a process instance will be feed as an input of a rule engine. A process's data is validated against a set of conditions. If conditions are satisfied, an action will be undertaken to make changes to artifacts. A rule instance that keeps track of rule execution is also created in the process as well. In our current prototype, we have integrated Drools rule engine [21] since the engine provides very efficient ways of evaluating business rules. The rule format is also easy to comprehend and can be written in xml format and drool format. Drools engine conforms to JSR94 standard and provide a set of APIs that allows us to integrate it to our system.

– **Coordination of Running Instances**

In order to coordinate running instances, we address this issue using process controller, artifact controller and service controller. Here, we will use our motivation example to describe how these controllers work together to coordinate all instances that are created during a process execution. Once the ACP system receives a request from a user to start an ordering process, the rule engine will evaluate the request. If preconditions of rule *r1* are satisfied, the rule engine will issue a command to the process controller to start a process execution of an ordering process. The rule instance of rule *r1* is also created at this point as well. The process controller invokes the process factory. The factory identifies the deployed process and instantiate a process instance for an *ordering* process. After receiving a corresponding process instance, the process controller will initialize a unique process id and register the process instance to a list of running process instance. Once a process instance is registered, an instance of rule *r1* is added to a list of rule instances. The artifact instance of an *order* artifact is also created by the factory and added to a list of artifact instances. The artifact instance is given a unique id that is used for identification purpose and its state is set to *start*. Next, the process controller orders a web-service controller to invoke the *createOrder* service. The service instance of *createOrder* service is instantiated and added to a list of service instances. The service controller will communicate with the artifact controller to retrieve data from corresponding artifacts. The *createOrder* service is then invoked. Once a response is returned, a service controller will send message data to an artifact controller to update the artifacts. The unique artifact ensures that correct artifact instance is being updated. A new artifact instance will be generated if necessary during this step as well. Mapping rule defined in rule *r1* controls the data mapping between service input-output and artifact instance. Once finishing data updating, the artifact controller will update a state of an *order* artifact from *start* to *open_for_item*. After completion of artifact updating, the artifact controller sends a signal to the process controller. The process controller will generate *Artifact_change* event to trigger rule engine to continue a process until *ordering* process is completed.

As we can see that each step of a process execution, the ACP system creates artifact instances, service instances and rule instances (cf. Section 3.3). These instances can be used to monitor a process since they contain overall information. To prove our concepts, we developed a prototype of ACP system and generated a test case based on the motivating example. After the prototype executed a test case, it is able to process data from running instances and generate a log file. In a log file, we can see detailed information for each step of a particular business process execution. To generate this log file, the system need to capture data from rule instances, artifact instances and service instance at run-time. A rule instance contains identifier keys that

belong to involving service instance and artifact instance. These keys help define a relationship between rule instance and the other instance in each step of a process execution. This enables our system to be able to generate a record for each step during run-time. As shown in Fig. 7, Pre and Post-artifact are also recorded in a log file to show progress of each artifact from initial state to final state. The system records these data before and after service invocation. We can also use these pre and post artifact data to help facilitate process provenance if it is necessary. Therefore, this is a solid proof to illustrate an advantage of artifact-centric business process regarding to monitoring and reporting.

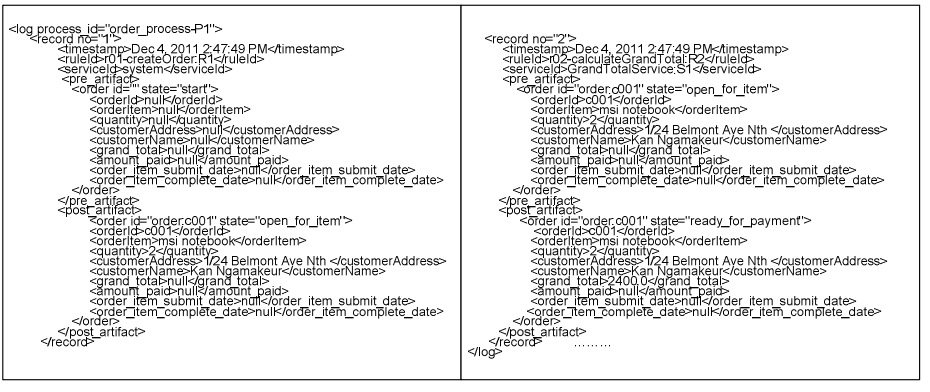

Fig. 7. Log record of a test case based on the motivating example

4.3 Technical Evaluation

In this section, based on the result of our implementation prototype, we discuss on the technical evaluation of our ACP system as well as a detailed comparison between two realizations approaches. After a prototype of ACP system is completed, we have simulated test cases based on our online ordering process. The result shows that our framework can address our requirements. The developed system is able to manage running instances created during process execution. Each running instance, e.g. service instance, stores process execution data and can be used for purpose of monitoring and reporting as shown in Fig. 7. Log record of a test case based on the motivating example. A business rule engine is proved to be able to work solely to provide decision making that affects on running processes. In our current prototype, we centralize all decision making process into a single rule engine. Thus, it simplifies rule management. However, this may raise performance issue of process execution if there are thousands of business rule to be evaluated by a rule engine. Non-deterministic is also an issue since a rule engine fires rules simultaneously. However, their ordering is non-deterministic. Thus, sequence of process execution may be different even with the same business process. A task for evaluating reachability of running processes is needed to address this issue. In our implementation, we assume that there is no issue regarding to non-deterministic. Since business process models are defined implicitly in artifact-centric approach, this would be another issue for a

process modeler. An artifact model doesn't have any explicit control flows as in the traditional process model so this is not an easy task for the process modeler. Thus, an intuitive process designing tool needs to be further developed. As we know that there is another way to implement artifact-centric processes, we compare our system to this existing artifact system implemented using the model transformation approach described in papers [11, 5].

– **Realization Approach**

In our implementation, we used our direct approach to realize an ACP model whereas the opposite approach proposed in [11] attempts the conversion from an artifact-centric process model to a procedural model, e.g. BPEL. We consider that logical information of artifact-centric process model can be lost during the model conversion process since some logical information which defined in a declarative manner, e.g. business rules, is spilt and mapped into several control flows in a procedural model. Moreover, this conversion task is quite cumbersome, error-prone and time consuming. Our approach ensures that there is no loss of data during transformation of the conceptual model since logical information of the conceptual model can be mapped faithfully to the proposed executable model. Without any model conversion, this approach uses less time and reduces chance of making mistake. Thus, direct approach is considered to be more appropriate way to realize the artifact model compared to model conversion approach.

– **Flexibility and Changes Management**

Flexibility is strength of a process model in declarative style. A conceptual model of artifact-centric business process gains this advantage as well since it is specified in the same style. Direct approach that we used to realize the conceptual model guarantees that the executable model inherits flexibility from its conceptual model, whereas the other approach does not since tasks are locked up by control flows. As a result, flexibility is well supported for design-time and run-time for our approach, whereas the other approach partially supports flexibility at run-time as it depends on the functionality offered by a particular workflow system. Thus, Changes can be made directly on the implementation level in our direct realization approach. In contrast, changes have to made at design-time and then convert to the implementation if an artifact model realized in a procedural workflow.

– **Monitoring and Reporting**

As opposed to traditional approach for process modeling, an artifact-centric process model focuses on business artifacts as its first class citizen to model a particular business process. Each business artifact contains business-relevant data and its life cycle. Artifact data and life cycle of each artifact reflect progress of a particular process toward a business goal. Thus, business process monitoring and tracking can be done by inspecting artifacts. Our approach provides a feature of direct and consistent monitoring and reporting at both model and instance level since both data and life cycle are combined at model level and instance level. Our implementation illustrates that a particular process can be monitored by directly inspecting running instances at run-time without any technique involving data gathering and processing. Whereas, the other approach needs a sophisticated mechanism which may include retransformation from the implementation specification back to its model specification and backward mapping for some data to its model to gather and process all process information to provide monitoring and reporting functionality.

– **Verification and Conformance Checking**

Verification and conformance checking is very essential task for both traditional approach and artifact-centric approach for modeling business process to ensure validity of developed model so that it can be realized on an automated system to support decision making for a particular business process. Since we used direct approach to realize artifact-centric business process model, single model verification for both design-time and run-time can be achieved because an implementation level reflects its conceptual level. Thus, conformance checking can be achieved directly, whereas the other realization approach needs to have separate verification on both Artifact model and procedural model. Run-time verification does not reflect the base artifact model because of the conversion. Therefore, conformance checking needs some additional procedures.

– **Standards and Technologies Support**

Although our approach has several advantages, there are some drawbacks regarding to standards and technologies supports. Artifact-centric model realized on traditional workflow benefits from current industry-wide standards and technologies, e.g. OASIS, OMG, W3C and etc. Thus, an implementation of this realization approach is much easier and faster than our approach. Moreover, interoperability and execution in distributed environment are well supported when the artifact model is realized on traditional task-based workflow system. Currently, the developed prototype system only supports execution of an artifact-centric business process model in local environment and need further extension to handle distributed executions.

5 Related Work and Discussion

The notion of a business artifact was originated in [1] where business operational model can be constructed using a collection of lifecycles of all artifacts and their interaction. The operational model based on business artifacts provides the benefits that are flexibility of the representation, ability for analyzing changes, and ability for managing application. Moreover, Rong et al [2] improved the idea of the business operational model by introducing nine operational patterns for constructing the model and the method for verification the model. The concept of business artifact was further adopted in [5] as a business process model can be constructed using four core constructs that are artifacts, artifact lifecycles, services, and association. To realize an artifact-centered model, this paper presented a three-layer framework. The artifact-centric business process model considered as a logical specification sits on the top level. Then, it is converted to a conceptual flow that captures an essence of the top-level model in a procedural manner. Finally, it is mapped into an operational workflow for automation. Gerede and Su [3] focused on the middle layer of the framework, a conceptual flow, as it provides a separation between the logical specification and the physical execution; hence changes can be made freely to the implementation level as long as the logical specification remains unchanged. Therefore, the conceptual flow needs to be verified and optimized to ensure its correctness and performance respectively. This paper presented verification and optimization techniques for a conceptual flow.

There were other works that extend the artifact-centric approach. Yongchareon and Liu [9] introduced a process view framework for artifact-centric business processes followed by the extended version for modeling inter-organizational processes [22].

An (public/private) artifact-centric process view can be used to support participated organizations to have their own freedom to model and implement their own parts of the process while preserving global correctness of the collaboration. Narendra et al. [12] tried to address flexibility and monitoring issues of the service composition using business artifacts, and their lifecycles. The concept of context-based artifact was introduced in this paper. The contexts are not only used to keep track of changes that make on all artifacts but also used in the coordination between artifacts and web services. To support inter-organization, the artifact-centric hubs were proposed by Hull et al. [6]. The hubs provide a centralized, computerized rendezvous point, where stakeholders can access data of the common interest and check the current status of an aggregate process. The framework also incorporates access control mechanisms to cope security issues. Instead of the centralized hubs, Lohmann and Wolf [7] proposed the use of artifacts in a choreography setting. In particular, the artifact-centric business process models were enhanced with the concept of agents and locations. By defining precisely and clearly who is accessing an artifact and where the artifact is located, an interaction model that acts as a contract between the agents can be generated automatically. Liu el al [23] proposed an approach to performance monitoring based on Artifact-centric business. The first step is to create monitoring context skeletons from business-artifact definition and from user inputs. The second step is to derive the executable monitoring models from the monitoring context skeletons. This work may be able to apply to improve our prototype in the future.

As was indicated in [4] by Hull, the implementation of artifact-centric business process is considered as one area of research challenges needed. An artifact process model can be converted to a conceptual flow. Then, it is mapped into an executable workflow. This first approach was adopted in [11]. This paper introduced the conceptual flow, namely ArtiFlow, and showed how ArtiFlow can be mapped to BPEL. Cohn and Hull [8] illustrated that IBM has used BELA tool to map an artifact-centric process model into a workflow that runs on IBM's WebSphere Process Server. In this research, we use a different approach compared with Artiflow. Our realization approach is to generate the executable model from the logical specification of an artifact-centric model based on [9, 10] without any transformation of the model. We also develop our prototype to execute our proposed executable model where the system uses business rules to control each state of process execution. In Siena [15], users can model business artifacts and process as an XML documents in order to create a composite web application. Then, the application is deployed and executed on an execution engine. However, there is no use of business rules. Moreover, processes are still executed in a procedural manner.

6 Conclusion and Future Work

In this paper, we propose a new framework for realizing artifact-centric business processes. Especially, we showed how an artifact-centric process model can be realized in our system. Apart from the proposed system for the realization of ACP model, we also provided a detailed discussion on the advantages and disadvantages of our approach. Our future work will extend our system to support interoperability.

Acknowledgement. This research was partly supported by the Australian Research Council Linkage Project LP0990393.

References

1. Nigam, A., Caswell, N.S.: Business artifacts: An approach to operational specification. IBM Syst. J. 42(3), 428–445 (2003)
2. Liu, R., Bhattacharya, K., Wu, F.Y.: Modeling Business Contexture and Behavior Using Business Artifacts. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 324–339. Springer, Heidelberg (2007)
3. Gerede, C.E., Su, J.: Specification and Verification of Artifact Behaviors in Business Process Models. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 181–192. Springer, Heidelberg (2007)
4. Hull, R.: Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges. In: Meersman, R., Tari, Z. (eds.) OTM 2008. LNCS, vol. 5332, pp. 1152–1163. Springer, Heidelberg (2008)
5. Bhattacharya, K., Hull, R., Su, J.: A data-centric design methodology for business processes. In: Handbook of Research on Business Process Modeling (2009)
6. Hull, R., Narendra, N.C., Nigam, A.: Facilitating Workflow Interoperation Using Artifact-Centric Hubs. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 1–18. Springer, Heidelberg (2009)
7. Lohmann, N., Wolf, K.: Artifact-Centric Choreographies. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 32–46. Springer, Heidelberg (2010)
8. Cohn, D., Hull, R.: Business artifacts: A data-centric approach to modeling business operations and processes. IEEE Data Engineering Bulletin 32(3), 3–9 (2009)
9. Yongchareon, S., Liu, C.: A Process View Framework for Artifact-Centric Business Processes. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010, Part I. LNCS, vol. 6426, pp. 26–43. Springer, Heidelberg (2010)
10. Yongchareon, S., Liu, C., Zhao, X., Xu, J.: An Artifact-Centric Approach to Generating Web-Based Business Process Driven User Interfaces. In: Chen, L., Triantafillou, P., Suel, T. (eds.) WISE 2010. LNCS, vol. 6488, pp. 419–427. Springer, Heidelberg (2010)
11. Liu, G., Liu, X., Qin, H., Su, J., Yan, Z., Zhang, L.: Automated Realization of Business Workflow Specification. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 96–108. Springer, Heidelberg (2010)
12. Narendra, N.C., Badr, Y., Thiran, P., Maamar, Z.: Towards a Unified Approach for Business Process Modeling Using Context-based Artifacts and Web Services. In: IEEE SCC 2009, pp. 332–339 (2009)
13. Hull, R., Kumar, B., Lieuwen, D., Patel-Schneider, P.F., Sahuguet, A., Varadarajan, S., Vyas, A.: Everything Personal, not Just Business: improving user Experience through Rule-Based Service Customization. In: Orlowska, M.E., Weerawarana, S., Papazoglou, M.P., Yang, J. (eds.) ICSOC 2003. LNCS, vol. 2910, pp. 149–164. Springer, Heidelberg (2003)
14. Hull, R., Kumar, B., Lieuwen, D.F., Patel-Schneider, P.F., Sahuguet, A., Varadarajan, S., Vyas, A.: Enabling context-aware and privacy-conscious user data sharing. In: IEEE Intl. Conf. on Mobile Data Management, MDM (2004)

15. Cohn, D., Dhoolia, P., Heath III, F., Pinel, F., Vergo, J.: Siena: From PowerPoint to Web App in 5 Minutes. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 722–723. Springer, Heidelberg (2008)
16. Boley, H., Paschke, A., Shafiq, O.: RuleML 1.0: The Overarching Specification of Web Rules. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) RuleML 2010. LNCS, vol. 6403, pp. 162–178. Springer, Heidelberg (2010)
17. Hollinsworth, D.: Workflow reference model. Technical report, Workflow Management Coalition, TC00-1003 (1994)
18. WebSphere Process Server, http://www-01.ibm.com/software/ integration/wps/
19. BizAgi Business Process Management System, http://www.bizagi.com/
20. ActiveVOS Business Process Management System, http://www.activevos.com/
21. Drools Expert, http://www.jboss.org/drools/drools-expert.html
22. Yongchareon, S., Liu, C., Zhao, X.: An Artifact-Centric View-Based Approach to Modeling Inter-Organizational Business Processes. In: Bouguettaya, A., Hauswirth, M., Liu, L. (eds.) WISE 2011. LNCS, vol. 6997, pp. 273–281. Springer, Heidelberg (2011)
23. Liu, R., Vaculín, R., Shan, Z., Nigam, A., Wu, F.: Business Artifact-Centric Modeling for Real-Time Performance Monitoring. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 265–280. Springer, Heidelberg (2011)