SWIN
BUR
NE

SWINBURNE
UNIVERSITY OF
TECHNOLOGY

An Artifact-centric Approach to Generating Web-based Business Process Driven User Interfaces

Sira Yongchareon, Chengfei Liu, Xiaohui Zhao, Jiajie Xu

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

Abstract. Workflow-based web applications are important in workflow management systems as they interact with users of business processes. With the Model-driven approach, user interfaces (UIs) of these applications can be partially generated based on functional and data requirements obtained from underlying process models. In traditional activity-centric modelling approaches, data models and relationships between tasks and data are not clearly defined in the process model; thus, it is left to UI modellers to manually identify data requirement in generated UIs. We observed that artifact-centric approaches can be applied to address the above problems. However, it brings in challenges to automatically generate UIs due to the declarative manner of describing the processes. In this paper, we propose a model-based automatic UI generation framework with related algorithms for deriving UIs from process models.

1. Introduction

Over the past several years, the use of workflow management system in organizations has been considered as a promising automation approach to enable them to design, execute, monitor and control their business processes. It is conceived that current workflow technologies support organization's own developed web applications for users to efficiently interact with the processes they involve. The interaction of users and these workflow-based applications are through user interfaces (UIs) that are designed and developed when workflows are modelled. This can bring an issue of coupled alignment between business processes and UIs, i.e., changes of the processes that impact on UIs are to be managed in an ad-hoc manner [9]. Several works [9-12] have been proposed the adoption of *Model Driven Approach* (MDA) that specify the association between models that support the propagating changes and control the alignment of business processes and UIs of underlying applications.

   Traditionally, business processes are modelled by identifying units of work to be done and how these works can be carried out to achieve a particular business goal. This approach, so called activity-centric business process modelling, has been recognized as a traditional way of process modelling and it has also been used in many MDA approaches, e.g., in OOWS-navigational model [12], for the semi-automatic generation of UIs by deriving task-based models from business process

models. These approaches require UI modellers to know information that is needed to be inputted from users and then to manually assign it to corresponding UIs. Thus, the changes of the data requirements of any task are still not able to reflect to the UIs if the process changes, so a better approach is required.

We observed the traditional approaches of business process modelling and found that they have some drawbacks and are limited to support only partially automatic UI derivation. Especially, the data model and task model are defined independently and their relation may not be coherently captured in the activity-centric model. In addition, as the limitation of the model, the current derivation approaches can provide only one-to-one transformation, i.e., one task to one page of UI. A mechanism to combine multiple tasks to fit within a single page without losing control or breaking the integrity of the process, e.g. transaction, is not supported. To this end, we consider a new paradigm of process modelling called *artifact-centric approach* [1-4].

By using the artifact-centric approach, UIs can be automatically generated from business processes by deriving both *behavioural aspect* (navigational control flow relations between UIs) and *informational aspect* (related/required data in each UI) from the underlying processes. In this paper, we propose a web-based business process driven user interface framework. It comprises two models, *Artifact-Centric Business Process* (*ACP*) *model* and *User Interface Flow* (*UIF*) *model*, and a mechanism to derive UIF model from ACP model. The UIF model describes the constitution of UIs and their navigational control flows which can be derived from the underlying ACP model. In summary, our work makes the following contributions to the research in business process modelling and web engineering areas:

- Analyze the relations between artifact-centric web-based processes and UIs
- Facilitate the UIs derivation for processes with UIF models and algorithms

The remainder of this paper is organized as follows. Section 2 provides the formal model for artifact-centric business processes. Section 3 presents the approach for UIF model generation. Section 4 reviews the related works. Finally, the concluding remarks are given in Section 5 together with our future work.

2. Artifact-Centric Business Process Models

The concept of modelling artifact-centric processes has been established under the framework proposed in [4] with the formal model [5]. Our artifact-centric business process model (ACP model) extends their work. The model consists of three core constructs: *artifacts*, *services*, and *business rules*. An *artifact* is a business entity or an object involved in business process(es). A *service* is a task that requires input data from artifact(s) or users, and produces an output by performing an update on artifact(s). A *business rule* is used to associate service(s) with artifact(s) alike in a *Condition-Action-Role* style. To explain the model, we use a retailer business scenario consisting of two business processes: product ordering and shipping. The ordering process starts when a customer places an order to the retailer for a particular product and ends when the customer pays the invoice. The shipping process starts when the retailer creates a shipment and ends when the item arrives to the customer.

Figure 1 illustrates artifact lifecycle diagram of each artifact used in our business processes. We denote *l.v*[*m.s*] for the transition to be triggered by a performer with role *l* invokes service *v* if artifact *m* is in state *s*.

Figure 1: Lifecycle diagram of each artifact used within our business scenario

2.1 Syntax and components of ACP model

Definition 1: (Artifact class). An *artifact class* abstracts a group of artifacts with their data attributes and states. An artifact class *C* is a tuple (*A*, *S*) where,

- $A = \{a_1, a_2, …, a_x\}$, $a_i \in A (1 \leq i \leq x)$ is an attribute of a scalar-typed value (string and real number) or undefined value
- $S = \{s_1, s_2, …, s_y\} \cup \{s^{init}\}$ is a finite set of states, where s^{init} denotes *initial state*

Definition 2: (Artifact schema). An *artifact schema* $\mathbb{Z}$ contains a set of artifact classes, i.e., $\mathbb{Z} = \{C_1, C_2, …, C_n\}$ where $C_i \in \mathbb{Z} (1 \leq i \leq n)$ is an *artifact class*.

From our business scenario, we define a primary set of artifact classes as below.

- *Order = ({orderID, customerID, grandTotal}, {open_for_item, ready_for_shipping, in_shipping, shipped, billed, closed})*
- *Shipment = ({shipID, customerID, shipDate, shipCost}, {open_for_shipitem, ready_to_dispatch, in_shipping, completed})*
- *OrderItem = ({orderID, productID, shipID, qty, price}, {newly_added, on_hold, ready_to_ship, added_to_shipment, in_shipping, shipped})*
- *Invoice = ({invoiceID, ordereID, invoiceDate, amountPaid}, {unpaid, paid})*

We also define two predicates over schema $\mathbb{Z}$ (1) *defined*(*C, a*) if the value of attribute $a \in C.A$ in artifact of class *C* is defined and (2) *instate*(*C, s*) if the current state of artifact of class *C* is *s*, where $s \in C.S$

Definition 3: (Service). A service or task provides a particular function. A service may involve with several artifacts of classes $C_1, C_2, …, C_y$, where $C_i \in \mathbb{Z} (1 \leq i \leq y)$.

Definition 4: (Business Rule). A business rule regulates which service can be performed by whom, under what condition, and how artifacts' states change accordingly. Rule *r* can be defined as tuple (*c, v, Γ*) where,

- *c* is a conditional statement defined by a quantifier-free first-order logic formula (only AND connective (∧) and variables are allowed).
- $v \in V$ is a service to be invoked, and *v* can be *nil* if no service is required
- *Γ* is a set of *transition functions* where $Γ = \{\tau_1, \tau_2, …, \tau_y\}$, each $\tau_i \in Γ (1 \leq i \leq y)$ denotes a function *chstate*(*C, s*) to assign the state $s \in C.S$ to the current state of an artifact of class *C*

Table 1 lists some business rules in our business scenario.

Table 1: Examples of business rules

r1 : *Customer c* requests to make an *Order o*	
Condition	$instate(o, init) \land \neg defined(o.orderID) \land \neg defined(o.customerID) \land defined(c.customerID)$
Action	$createOrder(c, o),$ $chstate(o, open\_for\_item)$
r2: Add *OrderItem oi* of *Product p* with a quantity *qty* to *Order o*	
Condition	$instate(o, open\_for\_item) \land instate(oi, init) \land defined(p.productID) \land defined(oi.productID) \land \neg defined(oi.orderID) \land defined(oi.qty) \land \neg defined(oi.price)$
Action	$addItem(o, oi, p),$ $chstate(o, open\_for\_item), chstate(oi, newly\_added)$
r3: Complete *Order o*	
Condition	$instate(o, open\_for\_item) \land o.grandTotal > 0$
Action	$completeOrder(o),$ $chstate(o, ready\_for\_shipping)$
r4: Pay *Invoice v* for *Order o*	
Condition	$instate(o, billed) \land instate(v, unpaid) \land defined(v.orderID) \land o.orderID = v.orderID \land o.grandTotal = v.amountPaid$
Action	$payInvoice(v, o), chstate(v, paid)$

2.2 Artifact system for artifact-centric processes

In this section we define *artifact system* as the operational model for capturing the *behavior* of artifact-centric processes. The artifact system is modeled by adopting the concept of state machine for describing behaviors of objects in a system [6].

Definition 5: (Artifact Machine). An *artifact machine* defines state transitions of an artifact class. An artifact machine *m* for an artifact of class *C* can be defined as tuple (S, s^{init}, T), where *S* is a set of states of *C*, $s^{init} \in S$ is the *initial state*, and $T \subseteq S \times V \times G \times S$ is a 4-ary relation of a set of states S, services *V*, and guards *G*. A transition $t = (s_s, v, g, s_t) \in T$ means that the state of the artifact will change from s_s to s_t if service *v* is invoked and condition *g* holds.

Definition 6: (Artifact System). An *artifact system* Λ is a tuple $(\mathbb{Z}, V, R, M)$ where $\mathbb{Z}$ is an artifact schema, *V* and *R* are sets of services and business rules over $\mathbb{Z}$, respectively, and *M* is a set of artifact machines, each for a class in $\mathbb{Z}$

For an artifact class $C \in \mathbb{Z}$, given service set *V* and rule set *R*, its artifact machine $m \in M$ can be generated by deriving from corresponding business rules that are used to induce state transitions of *C*.

3. User Interface Flow Model Generation

In this section, we formally describe the constructs in *User Interface Flow Model* (UIF model), and propose an approach to derive UIF models from underlying artifact-centric process models. The model comprises (1) a set of web pages and (2) relations between these pages. A page may contain a single or multiple input forms. Each form contains input fields that user must fill in data to make a form completed. There are

two abstract aspects of the UIF models: *behavioural aspect* (navigational control flow relations between UIs) and *informational aspect* (related/required data for each UI).
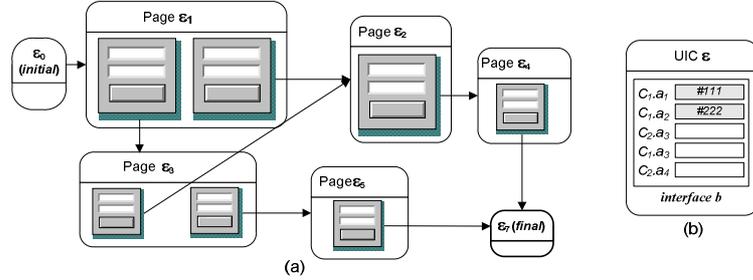
Figure 2: (a) UIF Model, (b) UIC with an interface and its required attributes of artifacts

Figure 2 shows the components and structure of the *UIF model*. The round-rectangle depicts a *User Interface Container* (*UIC*), which represents a single web page of UIs. An *Interface* represents a form comprising a set of required attributes of corresponding artifact that is used in the form. A single UIC may contain either empty interface (for the *final* or *initial* UIC), or a single or multiple interfaces (for *normal* UIC). The *Navigational Control Flow* (*NCF*) is used to indicate that once the interface with all required data has been submitted, then the action, e.g., service, corresponding to such interface is performed and the following UIC then becomes active. The UIF starts at the *initial* UIC and terminates when it reaches the *final* UIC.

3.1 Syntax of UIF model

Definition 7: (Interface). An *interface* represents a form of web page. It contains a required set of attributes of artifact, as well as a role of users and a corresponding service that will be invoked if users complete the form. Let b denote an *interface* and it is defined as tuple (O, ∂, Δ, v), where

− $O \subseteq \mathbb{Z}$, is a finite set of artifact classes used in the interface
− $\partial \subseteq \bigcup_i o_i . A$ is a required attribute set, which can be inputted/edited by users
− Δ defines a set of current states of each artifact of class in O when they are in the interface b, i.e., $\forall s_j \in \Delta, \exists o_i \in O, \exists s_j \in o_i . S$, such that $instate(o_i, s_j)$. We use $s_{o_i}^j$ to denote the s_j state of artifact of class o_i.
− $v \in V$ is a corresponding service which can be performed after attributes in ∂ are all completed by users

Note that an interface may contain nothing, called *empty interface*, where $O, \partial, \Delta = \varnothing$ and $v = nil$. It is only used in the *initial* and the *final* UICs.

Definition 8: (User interface Flow Model or UIF Model). The *UIF model*, denoted as θ, represents UI components and their relations, and it is tuple $(\sum, \Omega, B, F)$ where,

− $\sum = \{\varepsilon_1, \varepsilon_2, ..., \varepsilon_x\}$, $\varepsilon_i \in \sum (1 \leq i \leq x)$ is a UIC
− $B = \{b_1, b_2, ..., b_z\} \cup \{b^{nil}\}$, $b_i \in B (1 \leq i \leq z)$ is an *interface,* where b^{nil} denotes an *empty interface*
− $\Omega \subseteq \sum \times B$ defines the relation between UICs and interfaces

– $F \subseteq \Omega \times \sum$ is a finite set of *Navigational Control Flow* (*NCF*) relations. A flow f = $((\varepsilon_s, b_x), \varepsilon_t) \in F$ corresponds to a NCF relation between the source UIC ε_s and the target UIC ε_t, such that when ε_s is active and every attribute in ∂ of interface b_x is completed then ε_t is enabled (activated) and ε_s is disabled (deactivated).

According to two aspects of the UIF model, the *behavioral aspect* is represented by its UIs components and their NCF relations, while the *informational aspect* is represented by internal information of artifacts required for each interface. Once we defined ACP and UIF models, then the next step is to derive UIF models from underlying ACP models. Two main steps are required: (1) generating the interfaces and their NCF relations for constructing the behavior of the model and (2) mapping the required artifacts and their attributes for constructing the information for each interface. These steps are described in Section 3.2 and 3.3, respectively.

3.2 Constructing the behavior of UIF models

Every machine in the system is required to be composed into a single machine as to generate the entire behavior of the system, i.e., *behavioural aspect* of UIF models according to the control logic of underlying business processes. In this section we define *artifact machine system* for the completed composition of all machines in the artifact system by adapting the compositional technique presented in [6].

Definition 9: (Artifact system machine). Let $m_i \oplus m_j$ denote the result machine generated by combining artifact machine m_i and machine m_j. For an artifact system with machine set M for its artifacts, the combined artifact machine, i.e., $\oplus_i m_i$, is called *artifact system machine*. Combined machine $m_c = m_i \oplus m_j = (S_c, s_c^{init}, T_c)$, where set of states $S_c \subseteq m_i.S \times m_j.S$, initial state $s_c^{init} = (m_i.s^{init}, m_j.s^{init})$, transition relation $T_c \subseteq S_c \times V \times G_c \times S_c$, and G_c is guards, such that G_c contains no references to states in m_i and m_j.
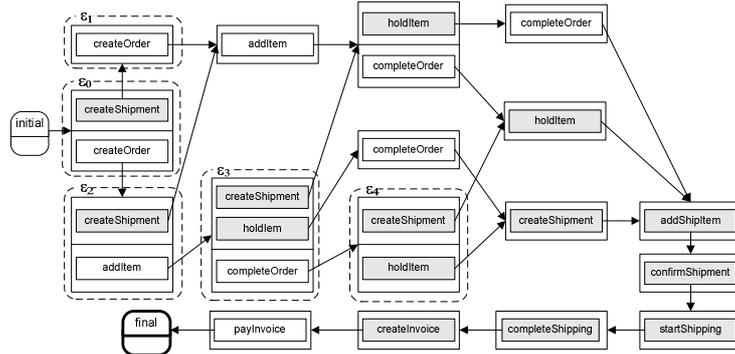
Figure 3: The behavioral aspect of UIF model

After completing the composition for *artifact system machine*, we need to process a mapping from such machine to the UIF model. The mapping contains two steps: (1) states to UICs mapping and transitions to interfaces mapping. (2) NCF relation generation. There can be multiple interfaces in a single UIC if such state has multiple

exit transitions. The result of mapping shows the *behavioral aspect* of the model. Figure 3 shows the result of applying this mapping to our business processes.

3.3 Mapping information of artifacts to interfaces

Once we completed *behavioral aspect* mapping of UIF model, then we need to generate its *informational aspect* by assigning artifacts onto interfaces. In this step, we need to find all the corresponding artifacts required for each interface. We can classify the information needs for a particular interface into: (1) a set of artifacts to be read or updated, and (2) a set of required values to be assigned to attributes of such artifacts. We can simply find both sets by extracting them from every condition of business rules that corresponds to a service to be invoked of such interface. Note that the required attribute set and artifacts for each interface are minimal and sufficient. They can be extended if users would like to incorporate other related artifacts by adding them into the interface; however, these additional artifacts need to be validated as to ensure that the behaviour consistency between UIF and ACP models is preserved. Here, we can say that our proposed information mapping explicitly overcomes the drawbacks of current approaches in which activities, data and their relation are treated separately.

4. Related Work and Discussion

In the context of business process modelling, Küster Ryndina, & Gall [7] established a notion of business process model compliance with an object life cycle. They also proposed a technique for generating a compliant business process model from a set of given reference object life cycles in forms of state machines. Redding et al. [8] conducted a similar work, where they proposed the transformation from objects behavior model to process model by using the heuristic net for capturing the casual relations in the object model. Compared with our work, their transformations use an object behavior model as input, while our work uses the artifact process models. In addition, these approaches are different from ours in such way that they do not consider state dependency between artifacts but we do.

In the area of web engineering in user interfaces, both Sousa et al. [9] and Sukaviriya et al. [10] presented a model-driven approach to link and manage software requirements with business processes and UI models. With their approaches, a process model is mapped to a UI model, thus change propagation can be managed more efficiently. Guerrero et al. [11] and Torres et al. [12] applied the similar concept for developing UIs corresponding to workflow models. All these approaches considered traditional activity-centric process models and proposed approaches to define the internal components and functionalities of the UIs at different levels, e.g., task-base model, abstract UI, and concrete UI. In comparison with these approaches, we considered the artifact-centric model to capture data requirements and their relation with tasks, and propose an automatic generation framework to provide a highly-cohesive bridge between the operational back-end system of business

processes and the front-end UI system. The generated UIs can be further customized by UI modelers without a concern of the integrity of business logic. Moreover, changes of data requirement that specified in the model can be reflected on UIs.

5. Conclusion and Future Work

This paper has proposed a model-based automatic UI generation framework for web-based business processes based on artifact-centric process modeling approach. In the framework, the ACP model and the UIF model are defined with a mechanism to derive the UIF model from the ACP model. The UIF models reflect the logic of business processes and intuitively represent what information is required during the processes. In the future, we plan to improve the model for supporting wider user interface requirements e.g., optional data elements, role-based configuration.

References

1. Nigam, A. and N.S. Caswell, *Business artifacts: An approach to operational specification.* IBM Syst. J., 2003. 42(3): p. 428-445.
2. Liu, R., K. Bhattacharya, and F. Wu, *Modeling Business Contexture and Behavior Using Business Artifacts*, in *CAiSE* 2007. p. 324-339.
3. Bhattacharya, K., et al., *Artifact-centered operational modeling: Lessons from customer engagements*, in *IBM SYSTEMS JOURNAL*. 2007. p. 703-721.
4. Hull, R., *Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges*, in *On the Move to Meaningful Internet Systems: OTM 2008*. p. 1152-1163.
5. Bhattacharya, K., et al., *Towards Formal Analysis of Artifact-Centric Business Process Models*, in *BPM* 2007. p. 288-304.
6. Lind-Nielsen, J., et al., *Verification of Large State/Event Systems Using Compositionality and Dependency Analysis.* Formal Methods in System Design, 2001. 18(1): p. 5-23.
7. Küster, J., K. Ryndina, and H. Gall, *Generation of Business Process Models for Object Life Cycle Compliance*, in *BPM* 2007. p. 165-181.
8. Redding, G., et al., *Generating business process models from object behavior models.* Information Systems Management, 2008. 25(4): p. 319-331.
9. Sousa, K., et al., *User interface derivation from business processes: A model-driven approach for organizational engineering.* in *ACM SAC 2008*: p. 553-560.
10. Sukaviriya, N., et al., *Model-driven approach for managing human interface design life cycle.* in *MoDELS 2007*. p. 226-240.
11. Guerrero, J., et al. *Modeling User Interfaces to Workflow Information Systems*. in *ICAS 2008*.
12. Torres, V. and Pelechano, V., *Building Business Process Driven Web Applications*, in *BPM 2006*, pp. 322-337.