RC24845 (C0908-016) August 19, 2009
Computer Science

IBM Research Report

An Application Framework for Business Developers to Create Service Based Business Applications

Jie Cui, Jing Min Xu
IBM Research Division
China Research Laboratory
Building 19, Zhouguancun Software Park
8 Dongbeiwang West Road, Haidian District
Beijing, 100193
P.R.China

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

An application framework for business developers to create service based business applications

Jie Cui, Jing Min Xu

IBM China Research Lab, Beijing, P.R.C. 100193
{cuijie,xujingm}@cn.ibm.com

Abstract. Enabling business developers to create business applications greatly reduces development cost and increases business adaptiveness. However, lacking of an application framework that is amenable to business people makes it hard for business developers to create decent business applications. This paper presents such a business application framework that consists of application components primarily including business artifacts, forms and tasks that business developers can easily work with. The different types of components are automatically composed into a service based application with Model-View-Controller (MVC) paradigm. The advanced business developers are allowed to write scripts to manipulate the behavior of the components and mash-up with external services for better customization and integration. The design considerations for the supporting tools and run-time are also discussed in detail in the paper. At the end, the feedback we got and consolidated from practitioners is analyzed for identifying potential enhancement and major future work.

1 Introduction

Nowadays, many companies are looking for an end-to-end solution for building business applications so that people can operate on business data across organizations. These applications have much more sophisticated interaction patterns than data centric applications since they usually support collaborative business processes with different roles. Traditionally, such applications are developed by professional IT teams with extensive knowledge and a set of middleware and development software. Moreover, the unanticipated or the frequency of changes makes software reengineering cycles too clumsy to address all needs of the application. Additionally, the lack of business domain knowledge of IT developers prevents the solution design and implementation to fully satisfy the requirement.

With the emerging and adoption of Web2.0 in business, Mash-up shows the technique of easy, rapid and on-the-fly integration of functionality and services using the browser in a visual, point and click manner, compared to the formal and costly enterprise application development. For example, a business person could build a "dashboard" to see how weather is affecting sales at retail outlets, by aggregating widgets from the web sites with mapping and weather services. Though there are lots of mash-up tools around the web, with the support of simply composing content from two or more sources; we have the vision of constructing more complex, enterprise wide, collaborative business application in this usable model of application creation by business people, rather than IT professionals.

From the business people perspective, they focus and manipulate the information entities called business artifact [2], such as Sales Order, Insurance Claim, to produce physical goods or services. Business application automates the business processing in a way to track the status along with a

specific artifact lifecycle, from creation to completion and archiving. Although artifact modeling methodology in business process management has been successfully applied in [3, 5, 9], the application framework of the artifact centric business is relatively less than that [18, 19] of the traditional flow centric approach, which focuses a coordinated set of work tasks (or activities) according to a predefined sequence of execution.

Our proposed approach takes the artifact notion from business operation point of view, and the openness of Web enables the artifact to interact with the high value Web based services. We define business developer here as those who has limited IT knowledge but knows business requirements and problems best, this ensures the solutions they created for themselves are rich with detailed features. Additionally, the serious business developer may possibly write simple script for enhanced business logic, just like they did in editing formula upon spreadsheet or defining email-filters. We aim at laying a framework emphasizing simplicity with effectiveness in building service based business application, and provide a proof-of-concept implementation, for automating and customizing the web application through browser. The main contributions of the paper are the followings:

1. We present the model of service based business application through identifying building blocks from the viewpoint of business people. Business artifact lifecycle and services operated on artifact are associated in MVC paradigm..
2. We propose both design time and runtime environment of the application framework, where application model creation is transparent to business developer, but can be updated and executed by developer's action.
3. We have implemented a Web based prototype which has been experimentally used to develop a number of applications by business developer. The tool has been proved to greatly lower the bar of building business application.

The structure of the paper is as follows: Section 2 presents a reference scenario, and the service based business application model is described in Section 3. Section 4 describes our architectural design of the proposed framework. Section 5 discusses the evaluation study and analysis result to identify further exploration. Section 6 presents the related work. Finally, we conclude the paper.

2 A Reference Scenario

To understand the problem and the need for such a framework, consider a typical e-commerce application where people can buy goods items electronically. We refer to this example as "Web Order" application. There are several actors cooperated in the process to reach the business goal of final successful service delivery. Figure 1 shows the high level process model which shows the sequence of tasks executed by different roles. It starts from a customer submits an online order after adding or removing items. A credit card authentication is automatically processed by system if the customer has successfully signed in and a credit check failure notification would be sent to customer. Then the Stockroom Rep receives the order and physically picks items from inventory. If the required item's quantity is under limit, he then places picked items into ship-unit boxes and marks them as picked. If the item is not physically available or insufficient, mark it as "BackOrdered". If all items are picked, mark the order as "ready to ship", otherwise, the order would be "BackOrdered" and still requires Stockroom Rep's regular checking if there are supplier's refills. The order could only be "ready to ship" if all the items are placed in ship-units. Here, we assume the customer wouldn't prefer the split

orders of in-stock and out-stock items, and can wait until all the items are arrived. Finally, the sales people charges the card and creates bill according to the order, marks the order as “shipped” after sending the packages to delivery staff. The order could only be “closed” once customer signs the order after getting the package.

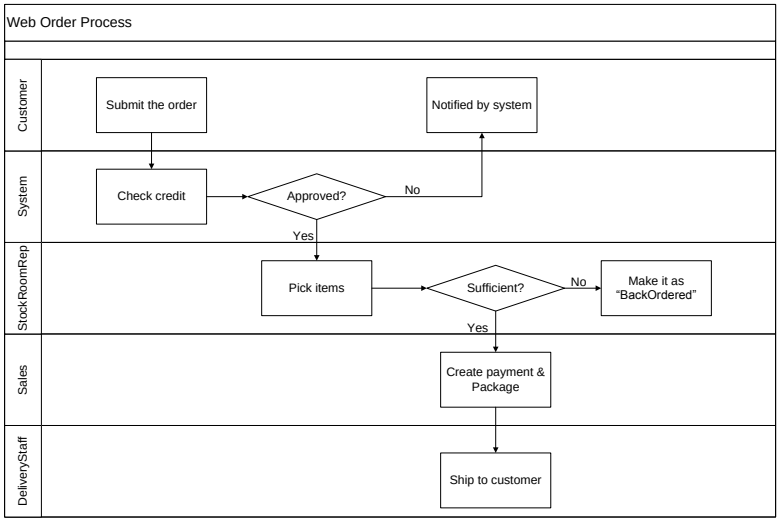

Fig. 1. Process model in Web Order application

We employed artifact centered approach to model the process operation which focuses on the core information entities that transform in business. Tasks initiated by business roles are intuitively considered as services which require input and produce output. The service invocation triggers the artifact state transition or exchanges data with other application. The artifact centered approach is not new, which is introduced by IBM research in [2], and there has been considerable research and development based on that work, e.g., [3, 5, 6, 8, 10, 11] to investigate techniques to support design model. Our work focuses on the general abstraction of artifact model with service association to hide the complexity of traditionally designing data and control flow. In the example, Web Order is the major business artifact. The Web Order persists data across the process, such as the date created, customer name and address, payment info, order items etc. The Web Order exists in different states or stages such as Submitted, Credit check approved or Canceled, Ready to ship or BackOrdered, Shipped and Closed. Fig. 2 illustrates the Web Order lifecycle showing the state transitions.

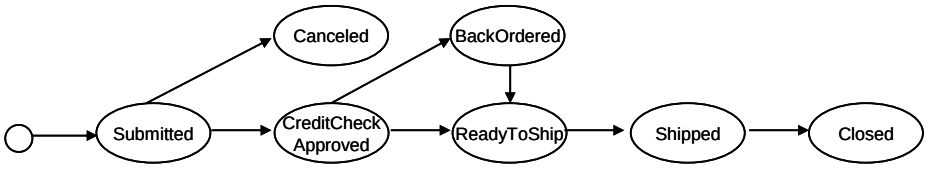

Fig. 2. Web Order artifact lifecycle

3 Business Application Model

Founded on the conceptual ideas of separation of concerns, we employ MVC paradigm, as the best practice for application partitioning. There are several building blocks in the application. Business artifact and its lifecycle are the entity objects in the Model layer. Services are associated with state transitions by a finite state machine, which plays the role as Controller. The presentation of artifact as

form and human task user interface (UI) inherits the view component of MVC. Figure 3 illustrates the high level business application model.

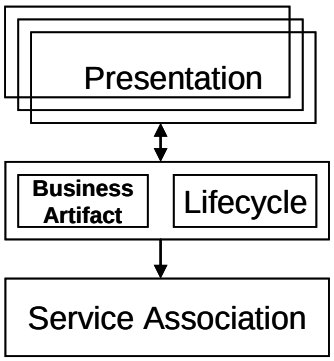

Fig. 3. Business application model

Business Artifact. Business artifact holds all the information needed to reach the business goal, records the decision making in the process execution. It may have a set of data fields with associated attributes. Data values are created, modified or exchanged with other business artifacts during its lifecycle. The progress of the business process is also tracked in an identifier in the artifact.

Lifecycle. In business terminology, lifecycle can be described as business stages throughout the evolution of artifact from creation to completion. "State" in the artifact indicates the changing or transition from one stage to another, as the result of the invocation of business services, either automatically executed or operated by human activity.

Service association. Services are units of work and tasks which would transform the business process toward the final business results. The concept of services is taken from the Service Oriented Architecture (SOA) [15] so that they can be the reusable components or business functionalities which are shared in the application. We distinguish the services between the following two categories.

1. Provided service: services exposed by the process and can be called by other applications. Usually they are implemented by Web Service or Rest Service identified by "type" attribute. In our example, once the order is shipped, the status can be queried by customer to get the progress of the processing order. It is up to application developers to decide what data items are external accessed.
2. Application services: services whose invocation can cause the artifact entries another states. Those services are executed by roles (either system or human) defined in the application. This mechanism makes human as "encapsulated services" which is allowed to invoke at runtime. Furthermore, transitions between states can invoke provided services exposed by other artifacts to interact with them through "InputMapping" and "OutputMapping" attributes.

Both kinds of services are intended to change the data in the artifact and application state by associating them with transitions. Guard can be attached to the transition as the conditional business logic for the entry of one stage. Figure 4 shows the detailed structure of the lifecycle component with service association.

Presentation. Presentations transform the artifact in different state into human perceivable representations. Generally, there are two level presentations in the application. Firstly, form view is the basic layout of the data items, which can be rendered as client side code (such as HTML) in the application. Secondly, as we know the human tasks are connected by a set of UIs for different users,

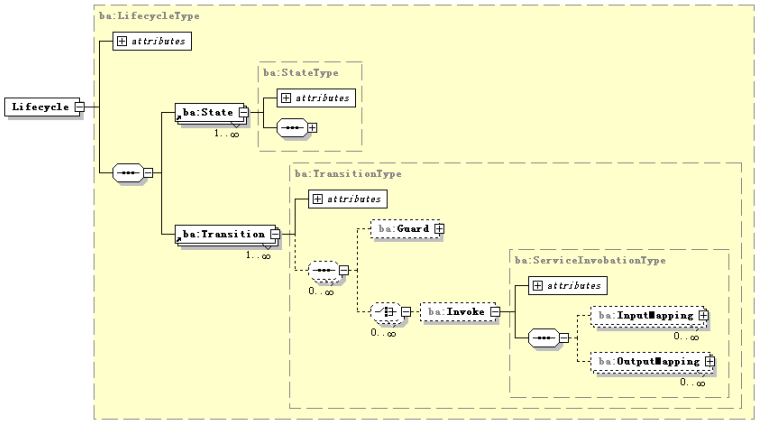

Fig. 4. Detailed structure of lifecycle

task views are dynamically constructed by embracing the form view and user actions from application services for possible transition at runtime. When the views are displayed to business roles staying in different stage of business process, the visibility and authorization of the data item are restricted by role's privilege. Therefore, different users can only operate the valid data and resource through user interface, e.g. ship-units field is only accessible by Stockroom Rep when he picks items, but is never showed to other roles.

To better illustrate our concept, we use the following XML fragment in Figure 5 which contains a list of descriptors (artifact, lifecycle, services) to describe our reference scenario. We omit the details of the definition of application services, roles and views, but more focus on their reference relationship.

```
<WebOrderApplication xmlns="http://magis.ibm.com/model/BA" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <ProvidedService uri="http://magis.ibm.com/Artifacts/WebOrder/{id}" type="restService">
        <Operation id="getStatus" operationName="getStatus" type="GET">
        </Operation>
    </ProvidedService>
    <ApplicationService id="Submit" roleId="Customer"/>
    <ApplicationService id="CreditCheck" roleId="System"/>
    <ApplicationService id="ItemsAllinStock" roleId="StockroomRep"/>
    <ApplicationService id="WaitforRefill" roleId="StockroomRep"/>
    <ApplicationService id="Ship" roleId="Sales"/>
    <ApplicationService id="SendToCustomer" roleId="DeliveryStaff"/>
    <ApplicationService id="InvoiceCreateService" roleId="System"/>
    <Artifact id="WebOrder" rootDataItemId="WebOrder">
        <DataItem id="WebOrder" schemaUri="xsd/WebOrder.xsd" rootElement="WebOrder"/>
    </Artifact>
    <Artifact id="Invoice" rootDataItemId="Invoice">
        <DataItem id="Invoice" schemaUri="xsd/Invoice.xsd" rootElement="Invoice"/>
    </Artifact>
    <Lifecycle id="WebOrderLifecycle" stateXPath="/WebOrder/Status">
        <State id="Submitted"/>
        <State id="CreditCheckApproved"/>
        <State id="Canceled"/>
        <State id="ReadyToShip"/>
        <State id="BackOrdered"/>
        <State id="Shippped"/>
        <State id="Closed"/>
        <Transition id="Submitted_to_CreditCheckApproved" sourceStateId="Submitted" targetStateIds="CreditCheckApproved" serviceId="CreditCheckPass"/>
        <Transition id="Submitted_to_Canceled" sourceStateId="Submitted" targetStateIds="Canceled" serviceId="CreditCheckFail"/>
        <Transition id="CreditCheckApproved_to_ReadyToShip" sourceStateId="CreditCheckApproved" targetStateIds="ReadyToShip" serviceId="ItemsAllinStock"/>
        <Transition id="CreditCheckApproved_to_BackOrdered" sourceStateId="CreditCheckApproved" targetStateIds="BackOrdered" serviceId="WaitforRefill"/>
        <Transition id="BackOrdered_to_ReadyToShip" sourceStateId="BackOrdered" targetStateIds="ReadyToShip" serviceId="ItemsAllinStock"/>
        <Transition id="ReadyToShip_to_Shippped" sourceStateId="ReadyToShip" targetStateIds="Shippped" serviceId="Ship">
            <Invoke serviceDefinitionId="InvoiceCreateService">
                <InputMapping>
                    <Source refType="artifact" sourceId="WebOrder" xPath="/WebOrder/Customer"/>
                    <Target refType="serviceRequest" targetId="CreateInvoiceRequest" xPath="/Customer"/>
                </InputMapping>
                <InputMapping>
                    <Source refType="artifact" sourceId="WebOrder" xPath="/WebOrder/Items"/>
                    <Target refType="serviceRequest" targetId="CreateInvoiceRequest" xPath="/Items"/>
                </InputMapping>
                <OutputMapping>
                    <Source refType="serviceResponse" sourceId="CreateInvoiceResponse" xPath="/generatedId"/>
                    <Target refType="artifact" targetId="WebOrder" xPath="/WebOrder/InvoiceId"/>
                </OutputMapping>
            </Invoke>
        </Transition>
        <Transition id="Shippped_to_Closed" sourceStateId="Shippped" targetStateIds="Closed" serviceId="SendToCustomer"/>
    </Lifecycle>
</WebOrderApplication>
```

Fig. 5. The description of Web Order application

In this example, the key business artifact “WebOrder” and the associated artifact “Invoice” are defined by XML Schema Definition (XSD). Each application service triggers the state transition from source state to target state by updating “state” field. Some services also affect the data of other fields, e.g. the Submit service creates an instance of the WebOrder artifact, updates the artifact with customer input (customer name, address, required goods items etc). Specially, the transition can invoke one service to correlate current artifact with other artifacts. E.g. When “ReadyToShip_to_Shipped” transition happens, an invoice would be created through mapping the appropriate data items from the original “WebOrder” artifact. Additionally, the “generatedID” from service response can be referenced by “InvoiceId” of “WebOrder” to ensure the traceability of “Invoice” from “WebOrder” object.

4 Application Framework Architecture

Based on the business process application model, we propose in this section the architecture design of application framework to facilitate business application creation. Figure 6 describes the architecture of the proposed architecture.

There are two environments in the architecture, application designing tools and runtime, designed for developers and end users respectively. The designing tools provide visual environment that makes it easy to enter data proprieties and privileges, related service innovation or script described actions for the artifact. The application model described in Section 3 is generated automatically by the system and can be updated by developers’ actions. The runtime environment provides ready to use application with friendly front-end by loading the application model and interpreting it after application starts up.

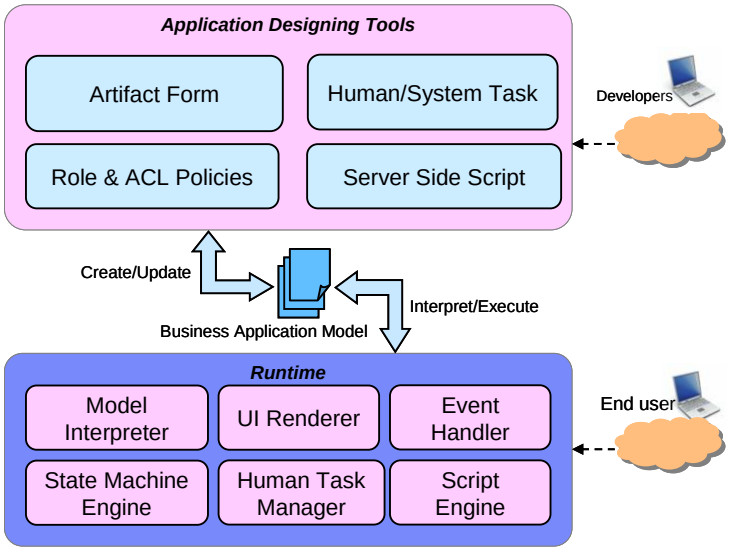

Fig. 6. Architecture of proposed framework

Artifact Form. As we discussed in Section 3, form is the presentation of artifact and interacts with users for business operation. We employ the form based approach for business developer to either create a brand new form or customize industry specified templates available. Following the wizard, developer can create different type of fields such as Text Area, Radio Button, Drop Down Selection, and Check Box, etc, easily set propriety on respective fields such as define field width, enter tips, and advanced options mapping to database features. Developer can also rearrange fields sequence in a

layout editor (Figure 7) to display them at front-end accordingly. In the back-end, the form view is stored as code for the runtime rendering, which reduces the effort for UI reconstruction.

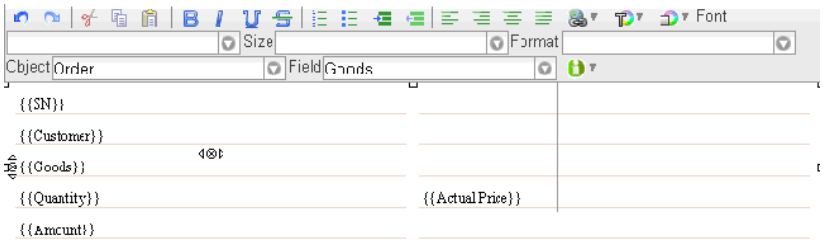

Fig. 7. Artifact form layout editor

Human/System Task. Task expresses the real business operation participated by users. Business developer can define the task flow with assigned participants. Meanwhile, artifact lifecycle with associated application logic is automatically generated on the fly by system. Each task is associated with an application services executed by system automatically or a human. Human actors can be resolved through various policies based on role, organization structure or specified email users. Each task could also invoke an external service to exchange data with other applications.

ACL Policies: In the architecture, we employed Role-based access control, which is an effective mechanism for establishing higher level organizational policy [7]. Developer can associate roles with fine grained permissions to certain data or resource, such as fields in artifact, form and task view, and then assign users to appropriate roles on the basis of job responsibility and business rules.

Server Side Script. Developers can use server side script syntax to write logic for their customized business needs, incrementally, making it more powerful and robust. The components defined in the application, such as artifacts, fields and views, are abstracted so that developer can reuse them as defined or system variables to generate logic while scripting. E.g. customer might want to receive email notifications once the order is shipped, thus developer can write the following scripts in the "state change" event. These scripts can be programmed to send email at runtime.

```
If (WebOrder.state=="shipped")
{
  SendMail
 [From: systemId
  To:   WebOrder.customer
  Subject: "Your order " + WebOrder.SN + " has been successfully shipped"
 ]
}
```

Model Interpreter. The model interpreter translates the application model described in Section 3 to the real code, which creates database schema and application logic automatically, persists the model variables, process and application context, and prepares for the instance initiation.

UI Renderer. This module extracts UI code, HTML in our prototype, and composes them to a page flow according to the policies and customized requirements, finally presents to user for interaction. The mediation is relative complex especially for web application, it has to maintain dialog states if there are multiple pages corresponding to a single task instance and support view switch from back and forth.

State Machine Engine. Artifact lifecycle is inherently finite state machine based, and we implemented a lightweight event driven state machine engine to support process choreography. Each state can receive a certain set of events, and service operation responses to the event to make state transition.

Event handler executes the data transformation logic that maps event parameters to service parameters, which provides the automatic invocation of states transition in response to event.

Human Task Manager. Human task manager provides support of staff resolution, ensures human activities are done in time, notifies others or administrator the task execution status, allows to "invoke" human as application services, and replaces manual task services with automatic system services when time out or according to business policies.

Script Engine. Script engine is able to load, compile and run script code. Unlike the client side script, the engine parses and executes the script block on the server, and allows code to inject data in the response to the client. Meanwhile, developers can still code client side script like JavaScript in the view editing mode.

5. Evaluation study

We implemented the prototype consisting application designing tools and runtime environment. We chose the web-based model for our prototype and the web browser as client interface, which relieves the developer from maintaining the development infrastructure. As our aim was to support business developers who are not experts in programming to create business application, a preliminary user study was conducted and the details is described in this section.

5.1 Participants

The participants were 10 people forming 3 groups from different domains. Group 1 had 4 university students from management science; Group 2 consisted of 3 staffs from business operation department and participants of Group 3 were sales and marketing people from a trading company. They had basic understanding of programming language, which may be gained in college's introductory course. 2 students knew native HTML with JavaScript programming from computer courses. However, they hadn't practiced or further studied programming for their own purpose of developing applications. The 6 company employees had 4 years working experience on average in their current role and evolved in 3 or more business processes daily.

5.2 Applications

The applications they created come from representative business scenarios. Group 1 selected Web Order application described in Section 2. Group 2 would like to develop OA functions living in expense management and project tracking applications. A simplified ERP application including order, product and customer management would be created by Group 3. The requirements and functions of the applications were defined by the business people themselves since they had the best knowledge and had involved in those business operations from their day to day practice.

5.3 Procedure

Each participant took part in one session that oriented by a facilitator. After an overview and short demonstration, the participants were given 2 to 3 hours to develop application. They had access to the usage manual of the prototype and the recorded video describing a basic development scenario. They were allowed to take notes for the identified bugs, advantages and inconvenience, and finally were asked to fill out a questionnaire.

5.4 Results

The questionnaire probed benefits and costs, and the interview for each participant was conducted to gather additional requirements and feedback. These ratings on a 1= low to 5 = very high scale for development efforts were collected from the questionnaire. The results are summarized in table 1. Development efforts were estimated by usability of each functional component and time spent on the learning and actual usage.

Table 1. Feedback of the development efforts

Development efforts	Group 1	Group 2	Group 3
Form development	1	1	3
ACL configuration	3	2	3
Human task development	2	2	2
Service integration	3	4	4
Scripting	3	4	4

The participants all respond they had little learning burden in form development through defining artifact data items and its layout, in contrast with other tasks. Group 3 had minor issues in designing the relationship and data constraints among different artifacts, asking for more support from us. Group 1 also spent longer time in ACL configuration since the application they developed has multiple roles associated with distinct privilege. Business process was quite easy to define by assigning tasks with different roles and users through a wizard. The advanced functions which require additional learning such as service integration and scripting seemed complex to Group 2 and 3 whose members had no programming experience, but they were exercised by Group 1 and used in several places inside the application to enhance business logic.

Feedback from the interviews also indicated training was critical to reducing those costs, as well as technical support to help users with development and testing. The other requirements from different groups are summarized as below.

1. Some requirements from the application couldn't be implemented by the prototype, such as complex statistic reporting across artifacts and printing functions in simplified ERP.
2. Because UIs are automatically generated, they were not able to be fully customized or leverage graphical designing tool to create specialized style for the application.
3. It didn't support debugging of web applications well, only displayed the error and the line number in the web browser only, which made it very inconvenient to find and fix bugs.

The issues raised by participates give us enlightenments for further exploration of our architecture design to improve data intensive processing and user experience. Firstly, the layout mechanism currently is based on storing and retrieving HTML fragments. We are investigating the mechanism

which allows importing the generated code into a desktop IDE for further development and customization. Secondly, although there are provided services defined in business process, which could be invoked by other applications, it's not sufficient for a wider range of application integration both from data and process perspectives. A more generic service creation and integration support should be generated, e.g, once an artifact is created, provided services of CRUD (create, retrieve, update, and delete) operation of the artifact are automatically generated which could be invoked from external applications. Finally, to ensure the quality of developed application, we will create a console for script pre-running and automatic testing which only requires developer to input boundary data but notifies them errors without manually executing the application.

6 Related work

There have been many model-driven development methodologies of Web-enabled process aware application, e.g. WebML [4] used extensions of the data and hypertexts models such as Workflow Management Coalition (WFMC) [1] activities to represent process model. Many process centric applications are still organized around relatively flat process-centric models and implemented in BPEL [17] flow for the coordination of tasks, but that places a rather steep learning curve and initially a negative impact on developer productivity, because it is quite difficult for the developer to be proficient in all involved technologies such as BPEL, Web services, XML, and J2EE. In contrast, our framework employs the artifact-centric approach which is inherently natural to business application and an agile development methodology which greatly lowers the technical barrier for business developer.

Recently, there are increasing websites on providing online Web application creation services. Examples are Wufoo[24], FormAssembly[13], CogHead[21], Zoho Creator[22] and Dabble DB [14]. These websites allow developer to create data centric web application using drag and drop interface in a development mode, execution environment mainly supports basic CRUD data operation. However, they can't fully support building complex application logic such as multiple role involved business process. Salesforce [16] provides additional libraries and programming language for developers to customize their application, but they are still target for the professional developers

Another trend in programming model and tools it to simplify application development, ranging from high level programming languages such as PHP [20] and Ruby on Rails [24] to visual programming tools such as VisualBasic [12] and Dreamweaver [25], but they are still complex for users with limited programming knowledge and require much learning and practice effort.

7 Conclusion

In this paper, we examined the business application from business artifact perspective, and proposed an application framework to facilitate the creation of service based business application for business developer with limited IT knowledge. The implemented prototype provides both design tooling and runtime environment of the framework, where developers can specify the abstract components such as artifacts, lifecycle, tasks and views as well as interaction logic by script. The evaluation study analyzed the feedback from participants and identified directions for further improvement.

Reference

1. The Workflow Management Coalition, http://www.wfmc.org/
2. Nigam, A., Caswell, N. S.: Business artifacts: An approach to operational specification. IBM Systems Journal, 42(3), 428–445 (2003)
3. Bhattacharya, K., Caswell, N., Kumaran, S., Nigam, A., Wu, F.: Artifact-centered operational modeling: Lessons from customer engagements. IBM Systems Journal, 46(4), 703-721 (2007)
4. Brambilla, M., Ceri, S., Comai, S., Fraternali, P. : (2002). Specification and Design of Workflow-Driven Hypertext. Journal of Web Engineering, Vol. 1, No. 2, 163-182.
5. Bhattacharya, K., Guttman, R., Lymann, K., Heath III, F. F., Kumaran, S., Nandi, P., Wu, F., Athma, P., Freiberg, C., Johannsen, L., Staudt, A.: A model-driven approach to industrializing discovery processes in pharmaceutical research. IBM Systems Journal, 44(1), 145–162 (2005)
6. Bhattacharya, K., Gerede, C., Hull, R., Liu, R., Su, J.: Towards formal analysis of artifact-centric business process models. In: 5th International Conference on Business Process Management (BPM) (2007)
7. Kang, M. H., Park, J. S., Froscher, J. N.: Access Control Mechanisms for Inter-Organizational Workflow. In: SACMAT'01, Chantilly, Virginia, USA (2001)
8. Hull, R.: Artifact-centric business process models: Brief survey of research results and challenges. In: On the Move to Meaningful Internet Systems: OTM 2008, OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008 (2008)
9. Strosnider, J., Nandi, P., Kumaran, S., Ghosh, S., Arsanjani, A.: Model-driven synthesis of soa solutions. IBM Systems Journal, pages 415–432 (2008)
10. Kuster, J., Ryndina, K., Gall, H.: Generation of BPM for object life cycle compliance. In: 5th International Conference on Business Process Management (BPM), 2007.
11. Liu, R., Bhattacharya, K., Wu, F. Y.: Modeling business contexture and behavior using business artifacts. In: CAiSE, volume 4495 of LNCS (2007)
12. Visual Basic, http://msdn2.microsoft.com/en-us/vbasic/default.aspx/
13. FormAssembly, http://www.formassembly.com/
14. DabbleDB, http://www.dabbledb.com/
15. Service-Oriented Architecture and Web Services, IBM, http://www.ibm.com/services/us/imc/html/soa.html
16. SalesForce, http://www.salesforce.com/
17. Business Process Execution Language for Web Services Version, http://www.ibm.com/developerworks/library/specification/ws-bpel
18. McCarthy, D., Sarin, S., Workflow and Transactions in InConcert. In: Bulletin of the Technical Committee on Data Engineering, IEEE Computer Society, Vol. 16, No.2 (1993)
19. Eckerson, W.: Case Study: The Role of IS in Reengineering. Open Information Systems, Patricia Seybold Group, Vol. 9, No. 2 (1994)
20. PHP, http://www.php.net.
21. CogHead, http://www.coghead.com.
23. ZoHo Creator, http://creator.zoho.com/
24. Wufoo, http://wufoo.com/
25. Ruby on Rails, http://www.rubyonrails.org/
26. Adobe Dreamweaver, http://www.adobe.com/products/dreamweaver/.