# An XML and Component-based IDE for Document Flow Application

Xie Xiaoqin, Li Juanzi, Ma Lu, Wang Kehong*

Contents



Key Words:Workflow, component composition, document flow

## Abstract

Business process in e-government mostly embody as the flow of documents. Constructing a web-based document flow system becomes an critical task for today's digital government. But few of them use an off-the-shelf workflow product. Why? One of the reasons is that most of the workflow systems are heavyweight, monolithic, and package a comprehensive set of features in an all-or-nothing manner. Another reason is that workflow technology lacks the constructs and modeling capability as programming language. It is incumbent on government IT organizations to transform their solution development to component-based computing model. Component technology isolates the computation and communication parts, but how to compose different software components is still a hard nut. An integrated development environment is necessary for CBSD. In this paper we propose a XML and component-based document flow-oriented integrated development environment (DFoIDE) for software developers. By writing some xml configure file, and operate on DFoIDE, developer can construct a workflow application quickly. This method divides system to several components and the activities in process are implemented as business component. Different components are discribed detailedly in this paper, especially one of the core component . Component Integrating Tool. Different perspectives in workflow application are seperated and depicted as different XML files. Correspondly, A component composition method for developing workflow application instead of workflow itself is proposed.

* Department of Computer Science
TsingHua University, Beijing, 100084, China
Email:xxq01@mails.tsinghua.edu.cn

In the other hand, object oriented method tangles data and control flow in its codes, so OO has impeded the software reuse now. Another reason is that workflow technology originates the organizational management and lacks the constructs and modeling capability as programming language.

# I. Introduction

Traditional paper-based documents have caused many related problems: maintaining; customizing, sharing, reusing, tracking and accessing of document. In order to quickly respond to the market needs and keep competitive advantages, companies rely heavily on electronic document and data interchange. Business process in E-commerce and e-government mostly embody as the flow of documents. For example, Commerce documents such as orders or invoices are the core of supply-chain management system. In government, notification, report or statement are presented as document. Previously document distribution is realized through email facility, but this method lacks an efficient process-control and feedback ability. As a typical workflow application, web document flow system can solve such problem and it is also an essential part for office automation system.

Component-based architectures represent a major shift in the IT industry from the traditional software development paradigm. As the IT industry has transformed around this new computing model, it is incumbent on government IT organizations to transform their solution development life cycle processes to gain the promised benefits: shorter time to market, lower rich, modular and adaptive systems [1]. Component technology isolates the computation and communication parts, and solve the problem of tight-coupled code, thus fosters reuse. But a new problem issues. How to composition these component? Connector concept in ADL just gives an abstraction and does not depict how to implement it. Workflow management system is an ideal implementation technology for connector.

Most researchers focus on the workflow meta-model. The workflow schema language covers different workflow perspectives [2]. The standard modeling language includes WPDL and up-to-date XPDL[3]. In most of workflow applications, the following perspectives are present: *functional* perspective, *operational* perspective, *behavioral* perspective, *informational* perspective and *organizational* perspective [2]. WPDL or XPDL integrates all of the perspectives in one language. But all this perspectives are defined in a modular way, i.e., independent from each other. Workflow model ignores those problems that exist when realizing such a system, such as user mapping,

application mapping and allowing strategy of task etc. In this paper, from the application development point of view, instead of workflow modeling, we propose a component integrating solution for developing workflow-ware application-document flow system. We use different xml language to model organizational and informational perspectives, make use of component-oriented methods to integrate these perspectives: functional, informational and organizational perspective. XML format enables the interoperability and reusability of different perspectives. A XML and component-based document flow-oriented integrated development environment (*DFoIDE*) for software developers is proposed. Different components in *DFoIDE* especially the core component - *Component Integrating* Tool (*CIT*) which bridge workflow engine and web applications, are described detailedly. The goal of *DFoIDE* is to help the developer to develop a document flow systems conveniently and quickly. Furthermore, component composition is a bottom-up approach, which is especially suited for large-scaled system. By adding or updating components we can customize the workflow application.

The structure of this paper is as following: section 2 describes the related works. Then Section 3 gives the design principle of *DFoIDE*. The detailed design and different components are depicted in Section 4. Finally is the comparison and conclusions.

# Ⅱ. related works

Many methods and tools have been developed to describe business processes and workflow as described in [2][4][5]. But most of them focus more on the workflow meta-model, namely how to use constructs and notations to describe a process instead of how to acquire the entire application requirements. Contemporary workflow management systems (such as IBM's MQSeries Workflow) consider software programs only as external applications. [2] proposed the use of Object Coanlination Nets for workflow modeling, which integrates software development and workflow management. But our research emphasizes the component-based software engineering.

Currently there exist many object-oriented frameworks, but few of them use an off-the-shelf workflow product. Large numbers of workflow products are available on the market. However current workflow management systems are not completely suitable for developers who need workflow functionality. The mismatch between the type of functionality provided by current workflow systems and the type of workflow functionality software developers need within object-oriented applications proposes several research directions in

workflow management [6]: object and workflow technology, new workflow architectures, workflow for developers and flexible workflow etc. How to build up a flexible, customizable and dynamic workflow application is the final objective. This is the focus of this paper. We put our focus on helping developer to build up a workflow application conveniently and quickly. This method is based on component and XML technology.

# Ⅲ. Design Principles

*DFoIDE* make full use of the process-oriented idea of workflow technology, which provides automated component integration. At the same time, component and component composition are adopted to manage overall complexity and change management for workflow applications. The following principles are considered when designing *DoIDT*:

1) Flexible information exchange and document customization/adaptation for multiple users' needs.
2) Achieving web-based knowledge management and document manipulation.
3) Component reusing and reduce the coding effort. Workflow language lacks the capabilities of modeling a system. We use software architecture description language to model system. Workflow engine is designed as a core system component. Monitor function or other additional module of workflow is designed as alternation component. Thus the core can be as small as possible. By adding new component the workflow application can be extended.
4) Realizing the sharing of document content, document style and document manipulation.
5) Separating the layout, content, and business logic. Each part relates to different component.
6) Workflow-based business process composition. Generative programming is used to integrate core workflow component to final implementation code.
7) XML configurable. Developer can construct a new application just by modify xml configuration file.
8) Generative programming. Most of the core code can be generated automatically.
9) Conforming to international standard, such as XPDL. Based on XPDL, we extend and enhance it by document schema model and organization schema model in document-processing domain.
10) Separation of perspectives of workflow application. In XPDL, organization model are mainly referred to participants. But participant is an abstraction level between the real performer and the activity. During run

time these abstract definitions are evaluated and assigned to concrete humans or programs. Function model is specified as application declaration, which is a list of all applications or tools required and invoked the workflow processes. The real definition of the tools is not necessary in workflow model and may be handled by an object manager. In DFoIDE, we propose different schema editor for creating organization model and document model. At the same time, a component repository is established. Function modules are stored and managed in the repository.

11) Document flow-oriented domain analysis. The commonality is designed as the underlying framework, and the variability guide the design of different editors (which will be depicted in Section 4). With these editors, developer can customize the system.

The task of document processing is almost the same in most document flow systems, including document creating, accessing, updating, subscribing, archiving and so on. While for some applications, along with the extension of functionality, the process and content of document flow must be modified. A web document flow system can be divided to three parts: presentation service, data service, and business logic service. The dynamic feature of this workflow application is composed of several parts as following from the point view of developer as depicted in table 1. The right column gives our solution in the design of DFoIDE to this kind of changes. Detail design can be referred in Section 4.

**Table 1. dynamic feature of document flow system**

| *Dynamical feature* | *solution* |
|---|---|
| The change of business process, such as adding a new document process activity. | Process modeling, WPDL |
| The change of presentation content, including rendering string, image icon. | Component integrating tool. Mapping between document model and abstract UI object. |
| The change of layout of presentation component. | Mapping between abstract UI Object and concrete UI object |
| The change of number of components, such as adding some new data field in the web page for user input. | Dynamically creating web pages in terms of document model |
| The change of relationship between presentation and actions. | Component integrating tool |
| The change of core document model. | Document schema editor |
| The change of organization, including its structure, member, num of member, member rights and so on. | Organization schema editor |

When constructing a similar document flow system for different application, the similar source codes must be written repeatedly. In order to avoid the repeating work and improve the efficiency of developing, a framework that contains the basic components of document flow system is defined. For the developers, they could write the specific XML files for the target application. These specific xml parsers and information extractor for different XML documents will be constructed. A workflow-based composition engine will aggregate those components to form final necessary programs, including jsps, java classes or ejbs. These source codes will be deployed to different server to form an executable application. The commonality of web document flow system is depicted in table 2.

# Ⅳ. System Design

The whole framework of *DFoIDE* is depicted as figure 1.

Within workflow meta-model, process model, organization model and information model are major description objects [7]. In terms of document flow domain, we depict the workflow application from four dimensions: document model, organization model, presentation model and process model. Each model is depicted as a XML schema file. Thus the problem of system design and implementation becomes a XML data-integrating problem. A document flow system always involves the database. For creating or modifying the database schema dynamically, we use generated data

**Table 2. commonality of document flow system**

| Description |
|---|
| Each system belongs to one organization |
| Each document involves two parts: document order and document content |
| Each document has a list of participants who process document in order. |
| Each document has a undertaker who is responsible for the document |
| Each document has a browser screen |
| Each document has a update screen |
| Each document has a processing process |
| Each processing process includes those tasks: document creating, signing, checking, updating, delivering, archiving and browsing. |

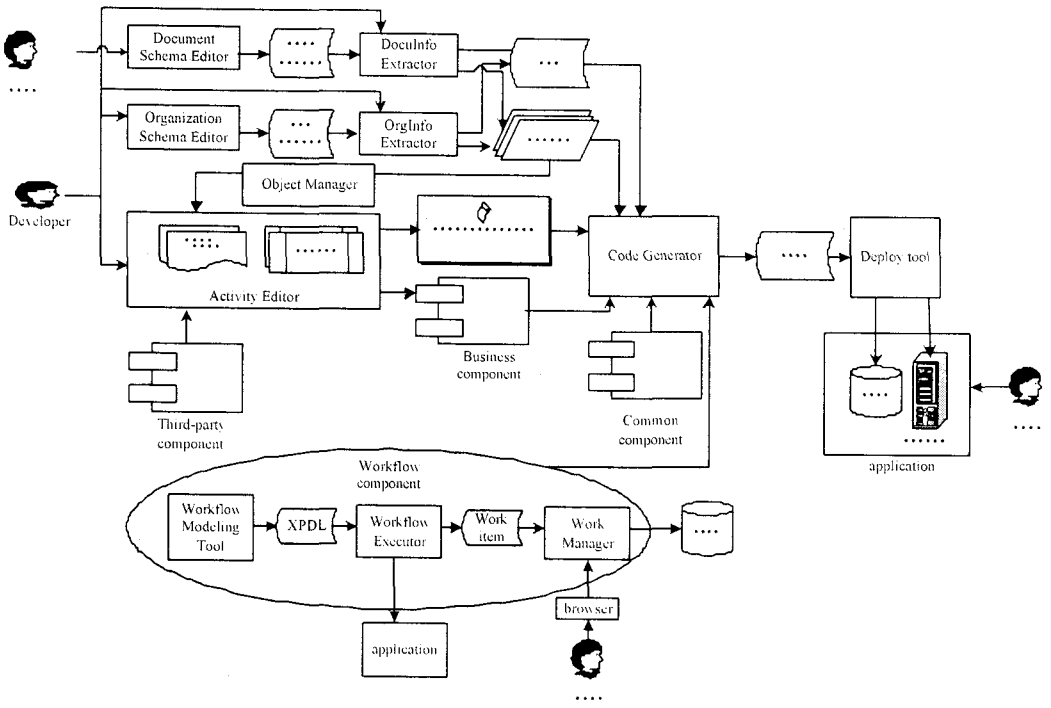
manipulate language (DML) to do it as depicted in figure 1.

The design and develop methodologies facilitate different aspects of J2EE web information system, including conceptual design, hypermedia design, presentation design and adaptation design etc. We offer designers and programmers a design framework based on a model-driven approach, in the following sections we will specify the different aspects of the complete application design in terms of different models following the separation of concerns principle.

### 4.1 Core Component - Component Integrating Tool (CIT)

This module is the core of *DFoIDE*. CIT bridges web applications and workflow engine. It integrates the document model, organization model and function model. As a web-based application, these three models decide the different program at different levels: document model decides the web page in presentation level. The content for presentation comes from the objects extracted from document model and outputs different data forms; Organization model

**Fig. 1. System Architecture**



decides who has the privilege to enter the web page. A mapping table between page and organization object is established in CIT; Function model decides what function is invoked in each page. The output of CIT is page component and business component. This tool provides a visual interface for developers to connect different model. Some page templates are provided. Each page is binding to one or more roles. Role decides the detailed presentation of web pages. In GUI, we can load one page template, and map different document object to page template, namely relate document model to

to real user in organization model( user is taken from organization model, which can be referred in Section 4.2.3). The interface model is depicted in table 3. *role_id* is defined in XPDL files. One task may be allotted to multiple users. *AllottedStrategy* element in interface model is used to define the stragegy. At present, the strategy type can be one element of set: {all, random, rotation, loadbalance, skillbased, appointed}. The *members* element describes the real users, each one is identified by their name.

**Table 3. User Mapping Interface**

```
<UserMap id="role_id" type="ROLE">
        <Members>
            <Member name="Potter"/>
            <Member name="Bob"/>
            <Member name="user"/>
        </Members>
        <AllottedStrategy key="ALL"/>
</UserMap>
```

presentation service. Controls in a web page (or a data form) have such features as visibility, editable, hidden etc. Role decides these features. Another important task in this module is to define the business operation for each page through relating a business component to a button of the page.

CIT implements the following two functions:

- User mapping

This module maps the performer in XPDL

- Application mapping

This module maps *application* element in XPDL to real applications (such as web application). Applications are divided into two categories. One is programs which including exe files, web service, java classes and packages etc. For example, in XPDL there is such a statement:

〈application Id="signDocument"〉……
〈/application〉

In the interface model as depicted in line 1

of table 4, an *application* element is defined also.

Its *ref* attribute specifies the concrete program. The other category is data form. When user needs to interchange with the workflow engine, data form is a preferred selection. The interface model is depicted in table 4.

**Table 4. Aplication Mapping Interface**

```
<application id=" signDocument " type="EXE" ref="D:/utilities/signDocument.exe" />
<application Id="sign_comment" type="dataform"/>
<dataforms location="D:/web/jsp/">
    <dataform id=" sign_comment"/>
</dataforms>
```

By parsing and annotating document schema the data forms will be created automatically. In CIT, we also provide extended interface for plug in user-defined components or third-party component.

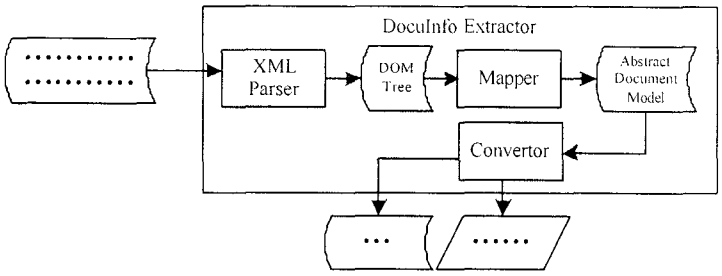
## 4.2 Other Components

### 4.2.1 Document Schema Editor

For a document flow system, the document model is critical. Invoice, order or notification, plan, report are different documents. Document schema editor can create and edit all kinds of document schemas. For example, a government document covers title, keywords, sign-info elements and so on. Some template schema will be provided in the editor. Other standard schema such as xCBL(*refer to xCBL.org*) can be imported to editor. Each schema defines one type of document.

### 4.2.2 Document Information Extractor

**Fig. 2. Document Information Extractor**




This module includes a Schema parser for document schema file. According to the parse result, each element and its property in xml file are conserved as multiple objects such as document content object and document meta-data object that will be referred to when creating a presentation-level page component. Some persistence

objects related to document should be created and stored in data base, so corresponding data manipulate language will be created also. Abstract document model is composed of document content, document order and multiple table object parts. As depicted in figure 2, *Mapper* translate DOM tree to proprietary and uniformed models according to domain ontology, then *Convertor* analysis the uniformed model and output DMLs and objects.

4.2.3 Organization Schema Editor

This editor can help developer to create the organization model. The schema is described as the xml-based organization model markup language (OMML). It consists of the following parts:

1) Entity definition. Entity includes department, user, role, workgroup.
2) Definition of relationship between entities: isPart, isPeer, exclusion, entrust
3) Definition of an organization tree of structure, including tree depth, the number of child of each tree node.

4.2.4 Organization Information Extractor

This module includes a XML parser for OMML. It will extract the information about users, departments, roles and workgroup. The entities are conserved as objects and relationship between entities are reflected in those between objects. Organization information should have corresponding tables in database, so some DMLs will be created. This module also derives the rules of role allocation such as multiple departments can play the same role.

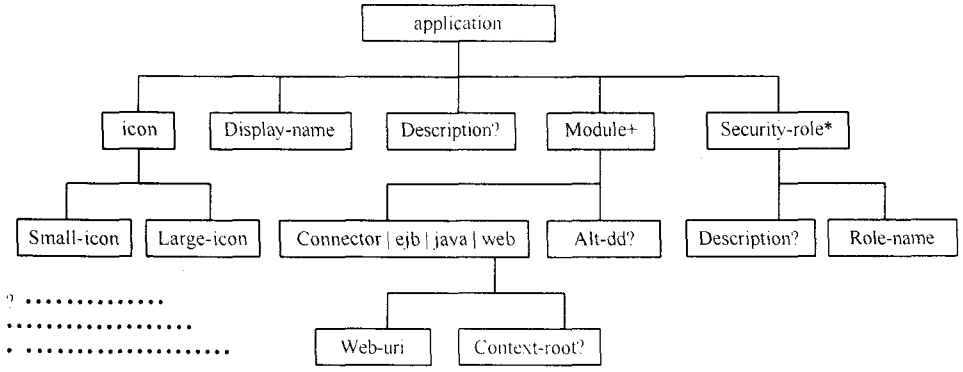
4.2.5 Workflow Engine Component

As depicted in figure 1, workflow component include three core modules: workflow modeling tool, workflow engine (or workflow executer) and task manager. Component idea is reflected in this component also. When we need other optional workflow function, such as log or monitor function, we can add them by composition.

4.2.7 Code generator

Code generator plays the connector role in this IDE. It connects the different component output by CIT. And it makes use of workflow component to navigate page component. In this way, a web-based document flow system is constructed. Code generator generates the final code, including jsp, java class, servlet, ejb etc. figure 4 described how to decompose a J2EE application and how to compose and deploy it. In implementation, we take Struts as our application framework. After generating the final code, we can deploy them to the running environment using apache tool.

Fig. 3. Decomposition of Application




# V. Comparisons and Conclusions

In current workflow productions which conforming to WfMC[7] specification well and powerfully, Shark[8] and OBE[9] are two typical system. But both of them extend WfMC specification, thus decrease the generality. As far as Shark, it just support java class mapping at present. In addition, for using client/server structure, the client application should be installed firstly. As for OBE, it does not provide method to update workflow-related data manually. User must call application to update data. However, in practice most of workflow activity can be handled through data form instead of calling application. While *DFoIDE* simplifies the development of workflow application and can bridge other workflow engine with web application. It can serve as not only a development environment but also a middleware or component. For fasten the

In this paper, a workflow and component-based IDE *DFoIDE* is proposed for development of digital government system especially web document flow system. In this IDE, workflow and component technology are combined to achieve a better solution for workflow application development. Xml file is served as the interface of components. With this tool, developer can construct different workflow-aware document flow system. The paper proposes a new method for developing Hypermedia Information Systems[10] instead of workflow itself.

# References


1. IAC EA SIG Succeeding with Component-Based Architecture in e-Government. Concept Level WHITE PAPER. March 2003
2. Wirtz G, Weske M and Giese H. The OcoN Approach to Workflow Modeling in Object-Oriented Systems. Information Systems Frontiers, 2001, 3(3):357-376
3. WfMC. Workflow Standard-Workflow Process Definition Interface. XML Process Definition Language. http://www.wfmc.org, 2001
4. Davulcu H, Kifer M, Ramakrishnan C R and Ramakrishnan I V. Logic Based Modeling and Analysis of Workflows. In:Proc of ACM Conf on Principles of Database Systems, 1998, 25-33.
5. Knolmayer G, Endl R and Pfahrer M. Modeling Processes and Workflows by Business rules. Lecture Notes in Computer Science 1806, Springer-Verlag, 2000, 16-29
6. Dragos -anton manolescu, micro-workflow: a workflow architecture supporting compositional object-oriented software development, phd thesis,university of Illinois at Urbana-Champaign, 2001
7. WfMC. Workflow Management Coalition: The Workflow Reference Model. TC00-1003, 1995.1
8. http://shark.objectweb.org
9. http://sourceforge.net/projects/obe
10. Heeseok Lee, Woojong Suh. A Workflow-based Methodology for Developing Hypermedia Information Systems. Journal of Organizational Computing and Electronic Commerce 11(2), 77-106(2001)