2011 Eighth IEEE International Conference on e-Business Engineering

iWeb: A Service-oriented Web Application Framework with Service Selection over QoS and Context

Sen Luo, Bin Xu, Kewu Sun, Yin Bai, Peng Zhang, Jiuhong Hu, Zonghui Li
Department of Computer Science and Technology
Tsinghua University, Beijing, China
{*luos, xubin, kewusun, baiy, zp, hjh, lzh*}*@keg.cs.tsinghua.edu.cn*

***Abstract*—We propose a service-oriented web application framework named iWeb, which enables web application adaptive to both context and QoS. In iWeb, a context model is established and context information is collected systematically according to the context model. An innovative service selection approach based on context and QoS is proposed in the framework as the leverage of implementation of application adaption. This approach can select the best services respect to QoS and context. For the usability of iWeb, thousands of available web services are collected and grouped by functionality in order to make service selection practicable. By providing development tools, service-oriented web application can be developed easily, and can fulfill its functionality using the selected services. The framework makes service-oriented web application more adaptive and flexible.**

I. Introduction

Web service and SOA have been widely adopted in recent years due to their inter-operation, reuse and globalization. With prosperity of web service, services with similar functionality start to emerge. It becomes possible to make SOA more flexible by making use of services diversity. But SOA development may become even more tedious and time consuming, as finding the best services is a challenging work.

Service selection is a prevalent way to find the best services and we adopt it as our strategy to find the best services in the framework. However, existing service selection approaches usually focus on QoS, they aim to find the QoS-optimal services. QoS-driven service selection approaches can only increase the flexibility of SOA to some extent, as they are totally ignorant of the complex and constantly changing environment. They can't provide different solutions under different environments.

In this paper, we propose a service-oriented web application framework, which using service selection over both context and QoS to enhance application adaption. In the framework, thousands of available web services are collected and grouped by functionality. A service group consists of web services with the same functionality, and during the development of web application service groups are used instead of individual services. Context information is a variety telling something about the environment. So far, most of the researches about context in web service and SOA involve user intention deduction more or less, and little of use in the framework. So we establish a context model according to framework. Context information is collected systematically and is used with QoS in service selection. A powerful service engine implements the service selection and is responsible for service scheduling and service call. In particular, the contributions of the framework are:

- A formal context model is established and a context information system is implemented. The context model is established through analysis of the characteristics of context information. The context information system can collect context information according to the context model from various clients. The system could be used as an infrastructure for collection of context information.
- An innovative service selection approach, which takes both QoS and context information into account, is proposed. The approach gives SOA greater flexibility, which provides different solutions for different users, or provides different solutions under different environment for the same user.
- A powerful service engine is implemented. The service engine is in charge of service selection, service scheduling and service call. It is the service engine which makes service selection approach practical in SOA.
- The framework increases the flexibility of the web application, but doesn't demand extra effort. Web services are grouped by functionality and the Web services in a group are encapsulated by a uniform interface. During the development of web application only service groups will be used and the developer does not need to know the details of service groups, the developer could view a service group as a component with certain functionalities. The flexibility brought by service selection does not need developer intervention and is achieved by the powerful service engine in the framework.

The rest of the paper is organized as follows. Section II presents an overview of the framework and Section III explains the framework in detail. Tools for the framework is introduced in Section IV. Section V discusses the related work. And finally, Section VI gives out the conclusion.

978-0-7695-4518-9/11 $26.00 © 2011 IEEE
DOI 10.1109/ICEBE.2011.54

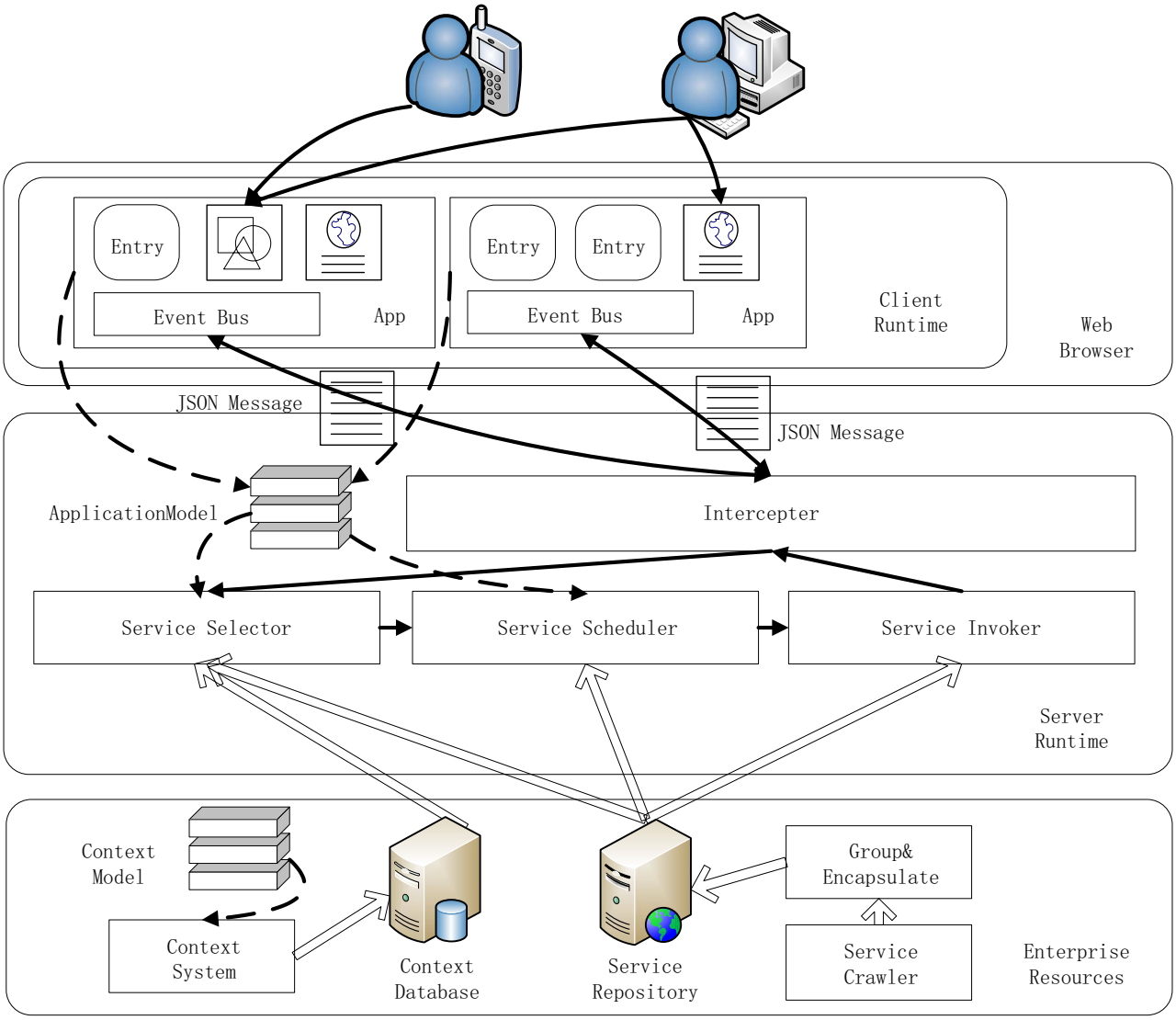

Figure 1. Framework Overview

II. Framework Overview

As shown in Figure 1, the framework can be divided into 3 layers.

The upper layer is responsible for interaction with users; user interface is an important part in applications in this layer. However, service call is not the upper layer's job. The upper layer transfers the task to the middle layer and presents the result returned by the middle layer to user, thus applications in the upper layer only need service call entries. The upper layer communicate with the middle layer through JSON format messages. Event bus is used to organize service call entries and user interfaces into an unified application.

The lower layer consists of enterprise resources in the framework, mainly a context database and a service repository. The context database stores context information collected by the context system, which is implemented on the foundation of context model. A service crawler is used to collect available web services on the Internet. The collected web services are grouped by functionality. Services in the same group have the same functionality and are encapsulated by a uniform interface. The service repository stores the results of service grouping and encapsulation.

The middle layer is the core part of the framework. The service engine is running in this layer, including interceptor, service selector, service scheduler and service invoker. The interceptor is responsible for capturing transferred service call from the upper layer. This service call request can be a composite service, which is composed from individual

services through workflow. After service call capturing, the service selector will decide the best services according to QoS and context information. The service scheduler will schedule the service call if the service is a composite one. And finally the service invoker will invoke the actual services to get the results and send the results to the interceptor. The interceptor then will dispatch the results back to the upper layer in JSON format.

Every application in the upper layer has a corresponding application model in the middle layer. The corresponding application model holds necessary information about the application, which will be used by the middle layer for service call. Using the framework to build a web application, developer only needs to develop the upper layer and the corresponding application model. And the framework offers tools for both the upper layer and the corresponding application model development to further reduce the burden of developers.

III. Framework Elaboration

A. Web Application

Web applications developed by the framework are consist of HTML code, CSS code and JavaScript code. And web applications are presented to users through web browser thus to achieve platform independence. A web application usually contains user interface, service call entries and the logic between them.

User interface is in charge of interaction between users and the application. In the framework, several commonly used layouts are predefined and dozens of useful user interface controls are provided to facilitate the user interface development.

The actual service calls are completed by service engine, so there are only service call entries in the web application. We should point out that a service entry can not only correspond to the call of an individual service, but also the call of a composite service which is composed by individual services through workflow. And the framework provides a powerful service call editor, which aims to help developers with development of service calls that correspond to the service call entries.

The logic between user interface and service call entries is accomplished through an event bus in the web application. The event bus monitors all the events in the web applications, through event publishing and event subscribing the links between user interface and service call entries can be established. And the framework provides convenient event publishing and subscribing mechanism. Except event publishing and subscribing mechanism, to further reduce the burden of developing a rich application, the framework provides injector mechanism, timer mechanism etc.

Each web application capable of running in the web browser has a corresponding application model as shown in Figure 2, which describes the web application in a formal language and aids service engine in service selection and service scheduling. The application model contains basic information for the web application, weight plan which will be used in service selection, flows which represent service calls, context information which might be used in the web application and intermediate variables used among flows.

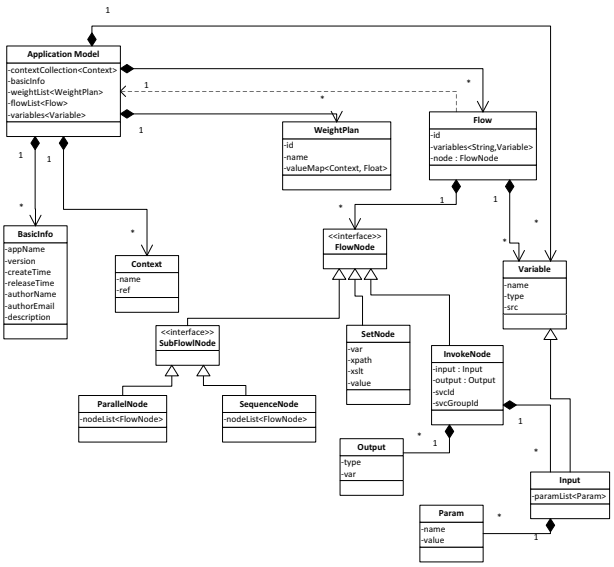

Figure 2. Application Model

B. Enterprise Resources

1) Context Database: One important feature of the framework lies in context information, web applications can use context information conveniently and the service selection is based on QoS and context information. To achieve this, context information collection is a necessity. As mentioned before, anything that capable of telling something about the environment is context information, context information is a great variety. We need a context model to classify the context information. It is almost an impossible task to establish a perfect context model which describes everything, as things are added or removed from the environment from time to time, the context model needs extensibility.

Upon on the characteristics of context information, we propose a context model using OWL [23] for its extensibility and formality. Context information varies in different domains, for instance, the context information collected from a mobile phone can be very different from the context information collected from a desktop. So the context model is divided into two parts in semantic as shown in Figure 3.

One is upper ontology, which provides a common vocabulary for representing general concept. The class context acts as an entry point of reference for declaring the ontology

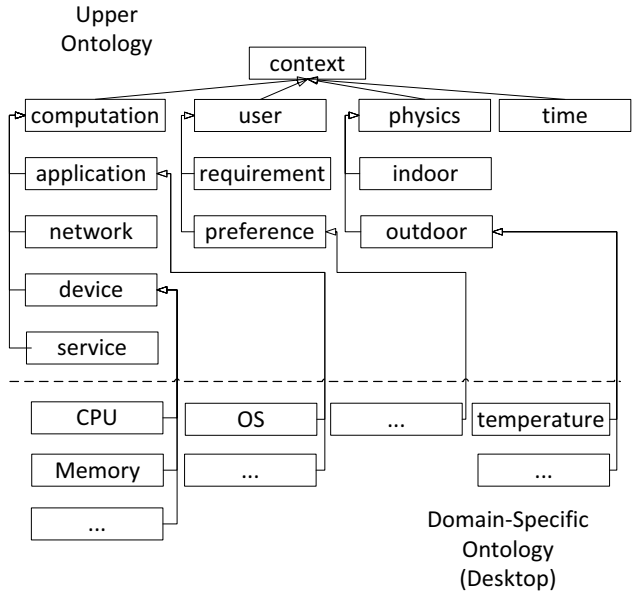

Figure 3. Context Model

context model. It is inherited by a set of derived classes, computation, user, physics and time. This means we divide context into four categories. These concepts may also consist of a set of derived classes too. The other is domain-specific ontology, which is a low level ontology defining detail of the general concepts in form of attribute and value.

On the basis of context model, we implement a context system which collects context information from various users and network terminals, such as mobile phones or desktops. And in addition to the collection of context information, the context system manages the context information from various sources in a uniform way and provides multiple query interfaces with different granularity to facilitate the use of context information.

2) Service Repository: The functionality of web application is completed by service call. Services are fundamental elements in the framework, so we use a crawler to collect thousands of available services from Internet. With rapid development and wide adoption of Web service, there has merged services that have the same functionality, so we group services by functionality to offer web applications greater flexibility by using service selection as leverage.

After the crawler has collected WSDL [24] documents from Internet, an information extractor will extract functionality information from the documents, such as types, messages, port types, etc. Then a term generator will be used to generate terms using the extracted information. The term generation has four steps, lexical analysis, tag removal, stop word removal and vector generator. In the end, the cluster will use a K-MEANS [32] algorithm to group the services by functionality. We will confirm the services in the same group having the same functionality manually.

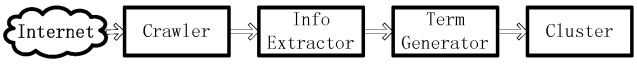

Figure 4. Service Grouping

After service grouping is done, although the services in the same group have the same functionality, they may have different input/output parameters. We encapsulate services in the same group using a uniform interface, to make the use of these services. We need three elements to encapsulate a service, a JAVA [25] file, a XML [26] configuration file and a XLST [27] file.

The JAVA file should implement a JAVA interface file, which have six interfaces as shown in Listing 1.

```
public interface ServiceAdapter{
    //The protocol to be used for calling the
        service, could be get, post, SOAP1.1,
        SOAP1.2, etc.
    public abstract Protocol getProtocol();
    //The URL of the service.
    public abstract String getServiceURL();
    //The URL of the SOAPAction, return null if
        the protocol is not SOAP1.1 or SOAP1.2.
    public abstract String getSOAPActionURL();
    //The map structure params represents the
        uniform input parameter for the group,
        this interface transforms the uniform
        input parameter to the parameter the
        service needs.
    public abstract String getStructuredParams(Map
        <String, String> params);
    //Get the XLST file.
    public abstract InputStream getXSLT();
    //Return the result in string using unified
        format.
    public abstract String postResult();
}
```

Listing 1. JAVA Interface

The XLST file is used to transforms result returned by the service to a uniform JSON format string, which will be delivered through postResult method in the JAVA interface file. If we write JAVA file and XLST file for every service in the group, the service invoker will be able to call all the services in the group using the same input parameters (params in method getStructuredParams in the JAVA interface file) and get the same output parameters as result (string returned by method postResult in the JAVA interface file).

```
<service>
    <serviceID>serviceID</serviceID>
    <groupID>groupID</groupID>
    <implementClass>serviceAdapterImpl</
        implementClass>
</service>
```

Listing 2. XML Configuration

The XML configuration file records encapsulation information for the service as shown is Listing 2. With these three files, the services in the same group could be invoked by the service invoker with no difference, thus we can use service groups for web application development in the framework.

And we have collected thousands of available services and have encapsulated hundreds of groups, provide them as enterprise resources in the framework.

C. Service Selection

The increasing availability of Web services that offer the same functionality but are different in non-functional properties has made the service selection a practical problem. Service selection is to select the best service from a group of services offer the same functionality. And in the past few years, QoS service selection has been extensively studied. QoS service selection select the QoS-optimal service from a group of services, Table I gives out some common QoS.

Table I
COMMON QOS

Cost	represents money that a consumer of a Web service must pay in order to use the Web service
Reliability	Represents the degree that a Web service is able to serve a request
Reputation	Represents the reputation of a Web services based on user feedback
Throughput	Represents the number of request that a Web service can serve at the same time
Response Time	Represents the time elapsed from the submission of a request to the time the response is received

In QoS service selection a utility function (or in other term) is used to evaluate the service. The most prevalent definition of utility function is by the weighted sum approach, which can be shown mathematically in the following formula.

$$F_i = \sum_{j=1}^{n} w_j \cdot f_{ij}, \text{ where } \sum_{j=1}^{n} w_j = 1 \text{ and } 0 \leq w_j \leq 1$$

Object function f_{ij} denotes evaluation score of some QoS, and as QoS is a variety, weight coefficient w_i which summed up to 1 is used to reflect tradeoffs among QoS. Although QoS service selection approaches can improve SOA flexibility to some extent, they have limitations. They can'n provide different solutions under various environments as they are totally ignorant of the complex and constantly changing environment.

We propose an improved service selection, which take both QoS and context information into account, to further enhance the flexibility of SOA. Service selection is done on non-functional properties, which in our regard include QoS and context information. Unlike QoS, which is usually numerical, context information is hard to use directly. So in our approach we have two categories of non-functional properties, QoS properties and context properties which should be telling how services will response to the environment.

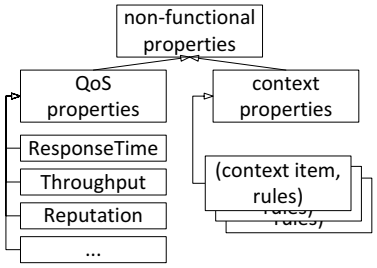

Figure 5. Non-functional Properties

Using context ontology model we proposed, the environment will be described in form of a set of context items, which have context item attribute. And as mentioned, context properties should be telling how services will response to the environment. To achieve this, we model context properties as key-value sets. The key is context item and the value is rules associated with the context item. The key tells which context item this context property has relation with, and the value tells how the service will response to related context item.

Similar to the first order logic, rules in the value are composed of constants, predict symbols which take effect on context item attributes, variables and logical operators. The details of these elements are given out in Table II.

Table II
RULES ELEMENTS

Constants	Individual values, sets and ranges
Predicate Symbols	LessThan, NotEqual, Equal, LessThanOrEqual GreaterThan, etc which compare individual value with another; ContainedIn, NotContainedIn which judge relation for individual value and set; Within, NotWithin which judge relation for individual value and range
Variables	Represent attributes for context items which can be individual values, sets and range
Logical Operators	$\neg$,$\vee$,$\wedge$,$\rightarrow$

We can use these elements to define AssertClause recursively as following:

1) Suppose A is predicate symbol takes effect on context item attributes, then A is AssertClause.
2) If A is AssertClause, then $\neg A$ is AssertClause.
3) If A and B are AssertClause, then $A \wedge B$ is AssertClause.
4) If A and B are AssertClause, then $A \vee B$ is AssertClause.

The rules are in the form of following formula, which means if AssertClause is true, the service will get a score of n, otherwise will get a score 0.5 (which means the context item has little effect on the service). And n is in the range of [0, 1].

$$RULE : AssertClause \rightarrow Score(n)$$

And in our approach, the object function f_{ij} could be used to denote score of some QoS property, and could also be used to represent score of some context property. Suppose G is a service group with services S_1,...,S_n, and for a service S_i, suppose it has QoS properties q_{i1},...,q_{im} and context properties $q_{i(m+1)}$,...,q_{io}. The object function f_{ij} for QoS property q_{ij} is calculated as following if the QoS property is to be maximized (the bigger the better):

$$f_{ij} = \frac{q_{ij}}{max(q_{kj})(k=1,...,n)}$$

The object function f_{ij} for QoS property q_{ij} is calculated as following if the QoS property is to be minimized (the smaller the better):

$$f_{ij} = \frac{min(q_{kj})(k=1,...,n)}{q_{ij}}$$

And the object function f_{ij} for context property q_{ij} is calculated as following where r_{ij} is the rule in the context property:

$$f_{ij} = r_{ij}()$$

So the utility function for the service S_i can be calculated as following (we assume the first x QoS properties are to be maximized):

$$F_i = \sum_{j=1}^{x} w_j \cdot \frac{q_{ij}}{max(q_{kj})(k=1,...,n)} + \sum_{j=x+1}^{m} w_j \cdot \frac{min(q_{kj})(k=1,...,n)}{q_{ij}} + \sum_{j=m+1}^{o} w_j \cdot r_{ij}() \text{ where } \sum_{j=1}^{o} w_j = 1 \text{ and } 0 \leq w_j \leq 1$$

As showed in the formula, utility function calculation for the service contains three parts. Two of them concern QoS properties and the last one concerns context properties. The improved utility function takes into account both QoS properties and context properties, making service selection able to response to context. By using this service selection approach, service selector in the framework can select best services according to QoS and context information, making the SOA much more flexible.

D. Service Engine

Service engine is responsible for actual service calls, and is core part of the framework. Service engine is run in server runtime and is consists of interceptor, service selector, service scheduler and service invoker.

The interceptor is in charge of the communication with web applications. And it provides callback functions, before the begin of service calls and after the end of services calls, so as to offer more operability.

By using the service selection approach we proposed, the service selector selects best services from groups according to QoS and context information. The object functions are calculated through formulas in subsection Service Selection, and the weight coefficients can be got from weight plan in web application model. The service selector will select the services with highest utility from groups.

The main function of service scheduler can be divided into five parts, interceptor information resolution, serial call of services, parallel call of services, single service call and parameter assignment. They correspond to interpret IntercepterInfo, SequenceNode, ParallelNode, InvokeNode and SetNode in the web application model. IntercepterInfo connects the service scheduler and the intercepter, and is the starting node and the ending node of a flow. InvokeNode represents calling a single service, which SequenceNode and ParallelNode mean call services in sequence or in parallel. SetNode helps data transformation among other nodes. The UML of service selector is shown in Figure 6.

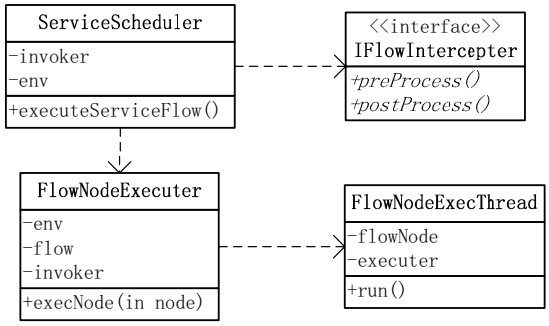

Figure 6. Service Scheduler

The service invoker will call services using service selection results and pass responses to the interceptor. It calls services in a uniform way by using implemented service adapters, and transforms the responses using defined XLST.

The service engine completes functionality of web applications through actual service calls, and makes fulfillment of functionality flexible by using enterprise resources, service selection approach we proposed, etc. It is the service engine that makes web application more flexible.

IV. TOOLS FOR FRAMEWORK WEB APPLICATION DEVELOPMENT

As mentioned in Section III, developers only need to develop the upper layer and the corresponding application model, while the functionality of web application is accomplished through service calls. And the framework provides two useful tools, web application editor and application model editor, to facilitate the development.

Web application editor is an Eclipse plug-in, which provides developers a general web project manager and organizes files of a web project in publishing format. It accelerates development by offering common user interface controls and useful mechanisms, such as event publishing and subscribing mechanism, injector mechanism, etc.

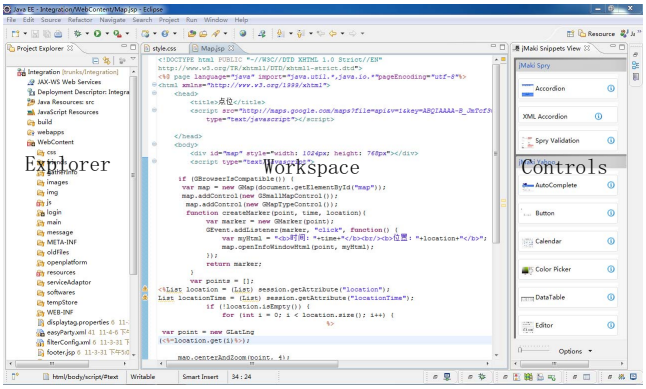

Figure 7. Web Application Editor

Application model editor, which is a visual editor and as one of the core components of Web application editor, is provided to facilitate the edition of application model. The editor consists of flow editor and overview editor. The flow editor help developers with service flows of application model in WYSWYG (What You See What You Get) mode, and the overview editor is responsible to all the contents remaining. These two tools improve the usability of iWeb framework, decrease development difficulty and shorten the development time cycle.

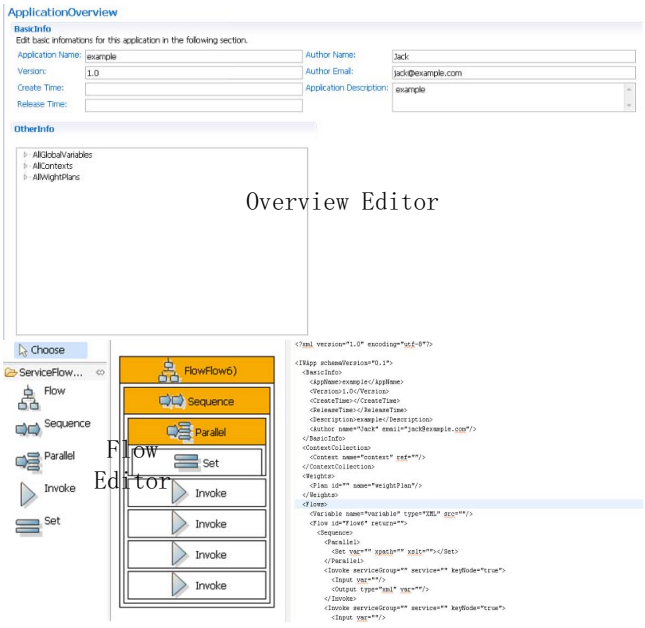

Figure 8. Application Model Editor

V. Related Work

Web mashups are web applications generated by combining content, presentation, or application functionality from disparate web sources [11]. Researchers have created different mashup tools and platforms, such as Yahoo Pipes [28], Google Mashup Editor [29], Microsoft's Popfly [30] and IBMs QEDWiki [31]. However, Web APIs and RESTful (Representational State Transfer) reign supreme in web mashups. Building web mashups using Web services is a burdensome task, which is especially true when the web mashups are designed to be adaptive.

Service selection can be used to enhance user experience of SOA. There exist several service selection approaches. Zeng [2] [3] proposed a global planning approach for QoS-driven service selection in composed service. Tao [4] proposed a broker-based architecture and several efficient heuristic algorithms to find the optimal services in composed service under QoS constrains. Alrifai [5] proposed an approach combining global optimization and local optimization to efficiently find optimal services. However, these approaches focus on QoS and are ignorant of context; they can only increase the flexibility of SOA to some extent.

Context is commonly used in service discovery and recommendation. Chen [6] designed an event-driven rule based system to recommend services according to people's context changes. Yang [7] proposed and implemented a system which can deliver contextually matched Web services to meet service requesters' needs. Hua [8] proposed an approach to capture the potential services which the user might need, based on the relations among context values. In these approaches, context is used for user intent deduction, in other words, context is used for functionality deduction. However, in these approaches the functionality of SOA is indifferent to context. Current context researches in Web service and SOA can't be used directly in our behalf.

VI. Conclusion

In this paper we present the iWeb framework, which is a service-oriented web application framework with service selection over QoS and context information. Developing tools including web application editor and application model editor are provided in the framework to facilitate the web application development. Web applications developed and run in the framework provide better user experience as the service selection approach will choose the best services according to the QoS and context. The framework contains context system and service repository. The context system, which is implemented based on a context model established through analysis of the characteristics of context information, could be used as an infrastructure for collection of context information. The service repository stores the grouped services from the Internet, which makes the service selection approach practical. In the end, the service engine in the runtime implements the innovative service selection approach and is responsible for service scheduling and service call. By all these aspects, the framework offers the web application great flexibility and adaption.

ACKNOWLEDGMENT

This work is supported by China National High-Tech Project (863) under grant No. 2007AA010306.

REFERENCES

[1] Jin Yu, Boualern Benatallah, Fabio Casati, Florian Daniel. Understanding Mashup Development. IEEE Internet Computing, vol. 12, no. 5, pp. 44-52, Sep. 2008.

[2] L. Zeng, B. Benatallah. QoS-Aware Middleware for Web service Composition. IEEE Transactions on Software Engineering, Vol. 30, No. 5, May 2004.

[3] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. Quality Driven Web services Composition. In Proceedings of international conference on World Wide Web (WWW), Budapest, Hungary. ACM Press, May 2003.

[4] Yu T, Zhang Y, and Lin KJ. Efficient Algorithms for Web services Selection with End-to-End QoS Constraints. In ACM Transactions on the Web, Vol. 1, No. 1, Article 6, May 2007.

[5] Alrifai M., Risse T.. Combining Global Optimization with Local Selection for Efficient QoS-aware Service Composition. In Proceedings of international conference on World Wide Web(WWW), April, 2009.

[6] I. Y. L. Chen, S. J. H. Yang, J. Jiang, Ubiquitous provision of context aware web services, IEEE International Conference on Services Computing (SCC), Chicago, USA, September 18-22, 2006, pages: 60-68.

[7] S. J. H. Yang, J. Zhang, I. Y. L. Chen, A JESS-enabled context elicitation system for providing context-aware Web services. Export Systems with Applications, Volume 34, Issue 4 (May 2008), pages: 2254-2266.

[8] Hua Xiao, Ying Zou, Joanna Ng, and Leho Nigul. An Approach for Context-Aware Service Discovery and Recommendation. In Proceedings of international conference on Web service (ICWS), August, 2010.

[9] Khalid Elgazzar, Ahmed E. Hassan, Patrick Martin. Clustering WSDL Documents to Bootstrap the Discovery of Web Services. In Proceedings of international conference on Web service (ICWS), August, 2010.

[10] Qi Zhao, Xuanzhe Liu, Dawei Sun, Tiancheng Liu, Ying Li, Gang Huang. Mashing-up Rich User Interfaces for Human-Interaction in WS-BPEL. In Proceedings of international conference on Web service (ICWS), August, 2010.

[11] Braga. D, Ceri. S, Daniel. F, Martinenghi. D. Mashing Up Search Services. IEEE Internet Computing, vol. 12, no. 5, pp. 16-23, Sep. 2008.

[12] Kyriakos Kritikos, Dimitris Plexousakis. Mixed-Integer Programming for QoS-based web service matchmaking. IEEE Transactions on Services Computing, Vol. 2, No. 6, pp. 122-139, 2009.

[13] E. Al-Masri and Q. H. Mahmoud. Discovering the best web service. In WWW, pages 1257C1258, 2007.

[14] Xia Wang, Tomas Vitvar, Mick Kerrigan, and Ioan Toma. A QoS-aware Selection Model for Semantic Web Services. International conference on service oriented computing, pp. 390-401, 2006.

[15] V. Agarwal and P. Jalote. Enabling End-to-End Support for Non-Functional Properties in Web Services. In Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA'09), Taipei, Taiwan, 2009.

[16] Sen Luo, Bin Xu, Kewu Sun. Compose Real Web Service with Context. 2010 IEEE International Conference on Web Services (ICWS 2010). July 5-10, 2010, Miami, Florida, USA.

[17] Xiangpeng Zhao, Zongyan Qiu, Chao Cai, Hongli Yang. A Formal Model of HumanWorkflow. Proceedings of 2008 IEEE International Conference on Web Services (ICWS 2008).pp 195-202.

[18] Kwei-Jay Lin, Jing Zhang, Yanlong Zhai, Bin Xu. The design and implementation of service process reconfiguration with end-to-end QoS constraints in SOA. Journal of Service Oriented Computing and Applications (SOCA). 2010(4):157-168.

[19] He Q, Yan J, Jin H, Yang Y. Adaptation of web service composition based on workflow patterns. In: Proceedings of international conference on service-oriented computing (ICSOC), 2008.

[20] R. Jurca, B. Faltings, and W. Binder. Reliable QoSMonitoring Based on Client Feedback. 16th international conference on World Wide Web (WWW2007), ACM, 2007, pp. 1003-1012.

[21] Daniel A. Menasc , Emiliano Casalicchio , Vinod Dubey. On Optimal Service Selection in Service Oriented Architectures. Performance Evaluation Journal, North-Holland, Elsevier Science, in press, July 2009.

[22] Xuanzhe Liu, Wei Sun, Yi Hui, Haiqi Liang. Investigating Service Composition based on Mashup. Service Congress 2007. pp 332-339.

[23] Web Ontology Language. http://www.w3.org/2004/OWL/

[24] Web Service Definition Language. http://www.w3.org/TR/wsdl/

[25] JAVA. http://http://www.oracle.com/technetwork/java/index.html/

[26] Extensible Makeup Language. http://www.w3.org/XML/

[27] Extensible Stylesheet Language Transformations. http://www.w3.org/TR/xslt/

[28] Yahoo Pipes. http://pipes.yahoo.com/

[29] Google Mashup Editor. http://code.google.com/gme/

[30] Microsoft's Popfly. http://www.popfly.com/

[31] IBMs QEDWiki. http://services.alphaworks.ibm.com/qedwiki/

[32] Jain AK, Dubes RC, Algorithms for clustering data. Prentice-Hall, Englewood Cliffs, 1988.