A New Paradigm for Web App Development, Deployment, Distribution, and Collaboration

Chao Wu and Yike Guo
Department of Computing, Imperial College London, London, U.K.

Keywords: Web Application, Web Development, Cloud-enabled Web Application.

Abstract: Web application is getting great prosperous while web browser is becoming one of the most important platforms not only on PCs, but also on mobile devices. And web application producing and consuming are going through a process of transformation sharped by the trends including Cloud computing, social networking, online application store, etc. It's necessary to look at the whole web application paradigm and get vision for its future. In this paper, we gave our insight on the web application paradigm, discussing its aspects of development, deployment, distribution, economic model, cloud platform, social diffusion and so on, and represent both the architecture and implementation based on our understanding.

1 INTRODUCTION

Web browser is becoming a key platform for applications. And Web applications have a superior position in gathering users, and have advantages including platform independency, no client side deploying, better accessibility, etc. So they are important for both producers and consumers. Producers would like to deliver utility through Web application to user, and gain considerable profit as return. And users would like to gain utility to fulfil their requirements within an acceptable budget. We believe current paradigm of Web application cannot fully fulfil demands from both sides.

For producers, their requirements include:

- Development: Web development could be easier and take less time, if reusability could reach a new level, and if much manual work could be done automatically.
- Deployment: Deployment costs a lot of time and money (for setting up machine, OS, Web application server, database, libraries, domain name, building environment, etc.).
- Scaling up: When user amount booms, deployment server needs be to scale up easily.
- Distribution and promotion: it's a challenge for producers to sell and promote their applications. They might spend a lot of time on tasks like SEO, trying to get attention from users.
- Understanding users' requirement: developers normally cannot meet users, and sometimes can hardly understand users, which hinder them from developing desired application.
- Cost and risk: Development, deployment, scaling up, distribution, and all these activities cost money. And the risk is that we do not know whether the application would success while spending money.
- Credit and reputation: Producers need to build reputation among users, and their credit could be an important indicator for collaboration in developer community.

For consumers, their requirements include:

- Utilities and performance: Web application is a mean of delivering utilities. User pays for the utility they want, not those useless functions. Also the performance should be promising: a large amount of users would not affect individual access.
- Accessibility: Web application should be attached to a user, not a browser. Users access their applications like their own instruments whenever needed.
- Low cost: User just pays for the utility they get. And there should have multiple pricing schemas for an application to fit different usage pattern.
- Participation: Users should be able to participate application's evolution, through mechanism like social rating and feedback.

Wu C. and Guo Y..
A New Paradigm for Web App Development, Deployment, Distribution, and Collaboration.
DOI: 10.5220/0004072904330438
In *Proceedings of the 7th International Conference on Software Paradigm Trends* (ICSOFT-2012), pages 433-438
ISBN: 978-989-8565-19-8

User can claim their requirements to encourage the producing of new application. Also users can share or recommend applications to their social network.

To meet these requirements, many technologies have emerged in recent years:

- Cloud computing: Cloud computing (Hayes, 2008; Armbrust et al. 2009) provides a new way of delivering computing as a service. It enables the elasticity for resource consumption, which is necessary in Web application development in different levels: in infrastructure level, services like Amazon EC2 allow developers to rent virtual computers to host their Web applications; in platform level, services like Google App Engine enable the development and deployment in centrally managed data centre. There are some other services on Cloud designed for Web application development, such as database (Hacigumus, 2002) and testing. Notably, a few platforms labled "Cloud Enabled Application Platform" (Natis et al., 2010) emerged.
- Tools for automation: The dream of application development without programmers was proposed many years ago (Martin, 1982). Since then, many automatic methodologies and tools are proposed and realized, such as MDA (Sendall and Kozaczynski , 2003), and tools like maven and jenkins. Now, the whole producing cycle of Web application is supported by these automatic methodologies and tools.
- Web application marketplace: Both consumers and producers benefit from the idea of application store as a distribution platform. Majority of application stores are distribution platforms for mobile application, with the model of distributing digital copies of mobile applications and then installing them on users' devices. But Web application means a different story: the marketplace sells the utility of the application, rather than copies.
- Social application: Many works have discussed the impact of social network on product diffusion. Web application could also be promoted and diffused through social networking approach. Facebook made this approach real and popular. Currently, there are many social games and other Web applications relying on Facebook to gather users.

These approaches facilitate the development of Web application. However, concentrating on a single aspect, rather than considering them as a whole, limits their effects. For example, if there is no economic consideration of distributing during development phase, the application would be not flexible for different pricing schemas. As a result, we must consider these trends as a whole paradigm for Web application. For such paradigm, we believe four principles should be emphasized according to our vision:

- **Utility.** Utility refers to the satisfaction received from consuming a Web application. Computation is about converting resources to utilities, and delivering utilities to consumers. The application producer consumes resources from resource provider (such as cloud provider), convert the resource (computation, storage, etc.) with business logic to build a service to deliver utility. Web application consumer uses a Web application to gain the utility without caring about the underlying resource or way's of constructing application. So in designing Web application paradigm, it's crucial to make sure that it facilitates the conversion and delivery of utility.
- **Economics.** Economic mechanism should be adopted to drive the whole paradigm: 1) producer would like to maximize its profit with the budget of resource consuming for converting a certain amount of utilities; 2) consumer would like to maximize received utilities within the budget of application consuming. Consuming of both resources and applications should have the properties of elasticity, which means paying by their usage. This 'pay by usage' principle should be considered not just in phase of distribution and promotion, but also in development and deployment.
- **Automation.** Many activities of Web application producing could be standardized and automatized, which increases the reusability, lowers the cost, and saves the time. For standardization and automation, it's important to design a platform rather than a set of tools.
- **Social Network.** Applications are built for users, and users make applications alive. Users should be able to share applications and recommend them to the others. They help the evolution of an application by giving feedback, requesting new feature, and purchasing it. Users and developers should be able to communicate with each other efficiently. So it's necessary to adopt a social

network as a communication and collaboration infrastructure to support the whole paradigm.

According to these principles, in this paper we design an architecture and example system to demonstrate our vision of new Web application paradigm. Our architecture of new Web application is presented in Section 2. And an implementation of example system for demonstration and experiment is described in Section 3. Finally, Section 4 gives the conclusions and future work.

2 ARCHITECTURE

Based on the analysis before, in this section we give reference architecture for Web application paradigm, and the implementation would be presented in the next section. The whole architecture is shown in Figure 1, which could be viewed as four service sets for development, deployment, marketplace, and social network.

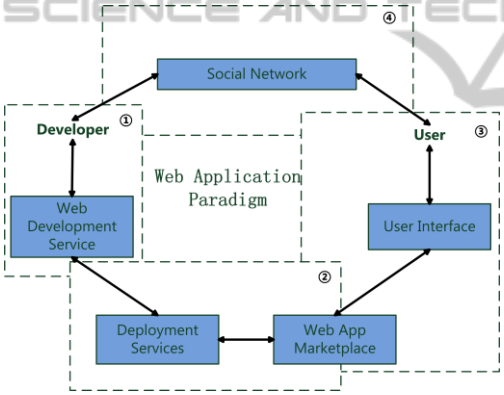

Figure 1: Architecture for Web application paradigm.

2.1 Development

Development should be done in an automatic services aided environment to improve reusability and productivity. These services include:

1. Initialization: Currently, to create a new Web application, we need to setup the environment containing OS, libraries, framework, Web server / database server, etc. These preparation works would be automatically done in a configured development environment with necessary components, and then delivered to developers with the form of 'Web application template', which is the skeleton code for a certain type Web application.

2. Management and teamwork: With this service, developers could easily do the management job including editing, testing, etc. These activities are done with automatic tools support such as a synchronized IDE workspace. And it's not necessary for a project team to create and maintain its own version control environment. Furthermore, communication and collaboration features should also be supported. Although there are lots of communication and collaboration tools, it would be better if they can be integrated with the development environment.

3. Programming SDK: To fit the deployment, marketplace, and social networking environment, SDK should be provided to support: 1) the access to the cloud service like data/storage, messaging, etc.; 2) the interfaces for marketplace like signal sign on, usage monitoring, and restricted zone; 3) the interfaces for social network like recommendation and sharing; 4) the interoperability to support open access.

4. Code marketplace: To improve the reusability, a marketplace for source code could be provided, where developers share or sell their codes and components.

With these services, Web application development would be done with the way shown in Figure 2.

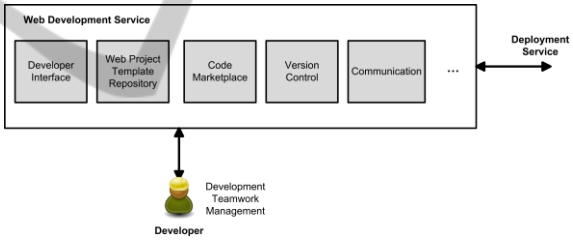

Figure 2: Web application development service.

2.2 Deployment

Automatic and continuous deployment is necessary for Web application, making it easy to migrate from development environment to deployment environment, and from deployment environment to application marketplace. It's desirable to utilize Cloud infrastructure for deployment environment, which overcomes the common problem of over-provision or under-provision for Web application. On Cloud, application server would be extended with cloud-supporting functionality, such as enabling multi-tenancy at the container level. Beside application server, data/storage services are also provided in cloud-based deployment environment.

We gave a design of extended application container (EAC) in (Sijin et al., 2012).

Both the application server and data/storage services should be implemented on the cluster, and

support the function of resource monitoring and dynamic provision. The cost of resource consuming should be able accountable, and the deployment ability can be scaled up or down when the users request amount doesn't match the workload of current deployment environment. With these services, deployment would be done with the way shown in Figure 3.

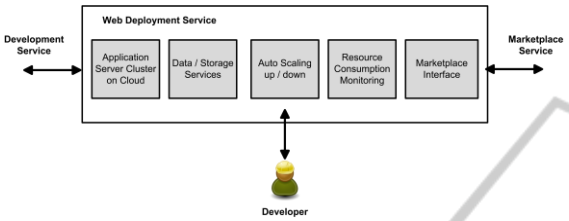

Figure 3: Deployment service.

2.3 Marketplace

Web application marketplace allows users to browse and use Web applications either for free or at a cost. Marketplace mechanism changes the ecology of Web application producing and consuming as shown in Figure 4. In traditional process of Web application development and accessing, producers have an idea, turn it to development activity, and deploy/publish the application to the Web; Users have their demand for some function, so they search or browse for target application.

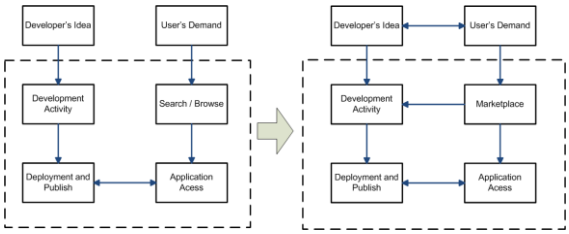

Figure 4: Change of ecology for Web application producing and consuming.

It's not easy for the suppliers to meet the requester. Marketplace provides a convenient place for both sides to find each other. Users are easier to find suitable application (through mechanisms in marketplace like social rating, relevant app recommendation, etc.). Producers could also quickly find the demand from users.

Marketplace should be able to provide different pricing schemas. Producers can choose one or multiple pricing schemas suitable for their application and then provide them to users. The purchased applications can be accessed by users through different forms of personal portal including website, Web browser (with plug-in), and mobile gateway application.

There should also be a crowdsourcing marketplace for users to have a place to specify their demand, and request developers to realize it. Some successful examples of crowdsourcing platform showed that using marketplace and economic mechanism, producers would be well motivated and organized.

2.4 Social Network

Social Network is a crucial infrastructure to support Web application's distribution and collaboration, providing following services:

- Application sharing.
- Application recommendation.
- Feedback and communication.
- Reputation and credit.

Summarization. With social networking infrastructure, marketplace, and cloud-enabled application development and deployment. The pattern of Web application producing and consuming would change, as shown in Figure 5. Marketplace and social network are positioned in the centre, and facilitate the flow of utilities, demands, resources between consumer, producer, and application.

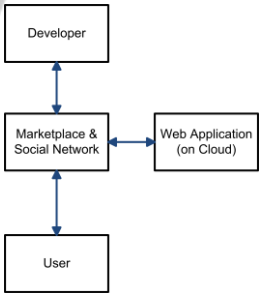

Figure 5: New pattern of Web application producing and consuming.

3 IMPLEMENTATION

We implemented a platform for demonstration and experiment, based on the principles and architecture described before. The platform was built on IC Cloud (Gu et al., 2010) as its cloud infrastructure, and implemented with open protocols and tools (including maven, jenkins, git, tomcat, etc.). The platform has 6 subsystems, which would be described below:

3.1 Development Service

This subsystem provides a set of automatic services, which make Web application development easier and cost less. Here are the main services in this subsystem:

1. **Web Application Template and SDK.** In this platform, developers start development by creating a Web project through a few clicks on the website, or through commend line, or through the eclipse plug-in.

Programming templates are provided, including "Spring Java framework", "GWT App", etc. In templates, a programming SDK is included with APIs for:

- Single sign-on: provides user management and single sign-on function. With it, users on the platform access all the applications without creating additional account.
- Usage monitoring: helps the application to notify and record when a user is gaining the utility from the application. It's similar to the process of declaring a database transaction.
- Restricted zone: lets the application to declare some part of its function is restricted and could only be accessed after users pay for it.
- Data and storage service: enables the application to access data and storage services provided by the platform.
- Other APIs include OAuth support, URL fetch, etc.

These APIs are mainly Web services provided by deployment environment, so a test server is provided for local development.

2. **Code Repository and Automatic Build.** A git repository is created automatically for a project. Once codes are pushed to git repository, the platform would build the project automatically. Jenkins with plugins like POM is adopted, so reports for building could be accessed. After build, the project would be automatically deployed to deployment environment.

3. **IDE Support and Test Server.** We integrated development services and a test server in Eclipse, so user can 1) create a new project; 2) access a new data/storage services; 3) version control with git; 4) and push code for deployment in Eclipse.

3.2 Deployment Environment

This subsystem provides a convenient way to deploy and scale up Web application, with Cloud infrastructure. Specifically, several services are provided:

1. Application server cluster and load balancer: Httpd load balancer and tomcat cluster are used to build application server cluster. Tomcat application server is packaged in a VM image, and created and clustered automatically when needed.

2. Data and storage service: Three types of data and storage services are provided: RDBMS services based on MySQL cluster, NoSQL data services based on MongoDB, and S3 style storage services.

3. Resource monitoring and scaling-up/scaling-down: JavaMelody is adopted to monitor Web application in application server node. Once the request amount exceeds the capacity of current cluster, deployment environment would scale up, which is mainly the processing of allocating new application server nodes, adding them to the cluster, and deploying the application to them.

Developer can view the current scale of deployment, or set it.

4. Interface to application marketplace: After deployment, producers can publish the application to Web application marketplace. Different pricing schemas are supported, and producers can set single or multiple pricing schemas for their applications.

3.3 User Service

This subsystem provides consumer interface to search, browse, purchase, and access Web applications. It contains following services:

1. Application marketplace: Users search, browse, and purchase the (usage of) Web application they needed. Different pricing schemas could be chosen. Users can view their report of usage, as well as related applications, user ratings, and other information. With single sign-on function, all the users in the marketplace become the potential user for the application. With usage API, the usage of user on an application would be logged, and the usage report and billing would be automatically generated.

2. Personal portal: Purchased Web applications would be accessible from users' personal portal. Meta-data of the purchase would be checked by a filter to assure the users' permission. User can access his/her portal through multiple approaches including website, Web browser (through plug-ins), and mobile device application.

3.4 Social Networking Interface

Services for social networking are provided:

1. Application sharing and social promotion: Users can share and recommend their applications for their friends in Facebook.

2. Social rating and developer credit: Users can directly rate the application, or indirectly rate it through purchasing, sharing, and recommendation.

These social rating would be used as the basis to determine the developers' credit.

3. Communication system: Users and developers can communicate with each other. Developers could ask for advice while the application is still in prototyping.

3.5 Crowd Sourcing

This subsystem provides a place for end users to post their requirement, and motivate developers to fulfil it. Users can submit a post in requirement marketplace, specifying the application they want, as well as the reward.

3.6 Development Marketplace

This subsystem is provided to encourage the sharing and reuse of code. Developers could share their source code, and other developers can search the code, and reuse it.

4 CONCLUSIONS AND FUTURE WORK

In this paper, we gave our vision of the new paradigm for Web application, which is becoming one of the most important types of application for both users and developers. Our understanding of this new paradigm emphasized the considerations of utility, elasticity, economics, and social networking. Based on these considerations, we designed reference architecture and implemented an experiment platform for demonstration. During the presentation, we did not discuss much about the technological detail, but tried to convey our overall vision of this paradigm, because in practical platform these details would very much.

For example, in our platform implementation, all the services are included in a single system and hosted in a signal data centre. However, in practice, the whole paradigm could be formed with multiple providers for different component, and hosted across multiple data centres.

Many works are planned to be done in future, such as how to federate the different service providers to an integrated environment, how to design a new type of copyright to fit the crowdsourcing development, how to estimate the application resources on the cloud so that application producers can choose suitable pricing strategy, and how to optimize the architecture of deployment component during scaling-up. We would work on these problems based this paper, and revise the platform implementation continuously.

REFERENCES

Natis, Y. V., Pezzini, M., \& Knipp, E. (2010). Gartner Reference Architecture for Cloud-Enabled Application Platforms. Analysis, (July).

Guo, L., Guo, Y., \& Tian, X. (2010). IC Cloud: A Design Space for Composable Cloud Computing. 2010 IEEE 3rd International Conference on Cloud Computing, 394-401. Ieee.

Hacigumus H. Providing Database as a Service. Data Engineering. 2002.

Sendall S, Kozaczynski W. Model transformation: the heart and soul of model-driven software development. IEEE Software. 2003;20(5):42-45.

Hayes B. Cloud computing. Communications of the ACM. 2008;51(7):9.

Armbrust M, Fox A, Griffith R, et al. Above the Clouds : A Berkeley View of Cloud Computing Cloud Computing : An Old Idea Whose Time Has (Finally) Come. Computing. 2009:07-013.

James Martin. Application Development without Programmers. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1982.

Sijin H., Li G., Yike G., Chao W., Moustafa G. (2011). Elastic Application Container: A Lightweight Approach for Cloud Resource Provisioning. AINA-2012