SOFTWARE ENGINEERING LESSIONS LEARNED FROM DEVELOPING AND MAINTAINING WEBSITES

Tammy Kam Hung Chan and Zhen Hua Liu
TZ Software Engineering Consulting, P.O.Box 5133 San Mateo, California, United States of America

Keywords: Web Engineering, Software Engineering, Software Engineering Education.

Abstract: Developing, maintaining and enhancing software features and functions for production websites are challenging software engineering activities. There are many aspects of software engineering practices and methodologies that are different in developing software features and systems for 24x7 production website compared with developing classical standalone software systems or client-server systems. This experience paper describes software engineering lessons that we have learned from developing, enhancing and maintaining software features for production websites and summarizes the key software engineering principles and practices that are essential for delivering successful 24x7 E-commerce based production websites.

1 INTRODUCTION

Since the birth of http and web browser, there have been many software systems and tools to help software engineers to build production websites. In the early days, there were Apache and CGI. Today, there are many application server based solutions ranging from J2EE based commercial application servers that are widely adopted by commercial business to PHP, Perl based systems that are commonly used by small businesses and non-profit organizations. However, no matter what systems or tools are used, the set of software engineering principles for developing web-based systems are the same. Web-based system is different from traditional standalone or client/server software system. The most notable differences are in the way the systems are used, their life cycle, and the way they are developed and maintained.

As a result of the differences between traditional software systems and web-based software systems, many of the traditional software principles and practices for developing standalone or client-server system software, while still applies, have subtle differences when applying to software systems in the web environment. The software design principles of *design for scalability*, *design for maintenance and robustness*, and *design for reuse* are essential principles in the context of many other domains of software development. However, their applications to the domain of web-based system have some unique and interesting characteristics to explore. In this paper, we discuss how the design principles for scalability, maintenance, and reusability are applied to web-based systems base on our experiences of developing, deploying and maintaining software production web sites.

The rest of the paper is organized as follows: in section 2, we discuss design for scalability; in section 3, we discuss design for maintenances and robustness; in section 4, we discuss design for reusability. In section 5, we state what we think is important to teach software engineering principles for web system development. In section 6, we conclude the paper.

2 DESIGN FOR SCALABILITY

One of the biggest challenges in building a web-based system is the ability to make the system scale. Scalability means the flexibility to accommodate user growth as well as handling high loads or sudden demand bursts. We find that designing and developing software for web base system by leveraging the principles of multi-tier architecture, state minimization, and workflow orientation is effective in building a scalable web system.

2.1 Multi-tier Architecture

To a user, a website appears to be a single system running on a single machine that appears to be "always-up". However, when building such a

Kam Hung Chan T. and Hua Liu Z. (2007).
SOFTWARE ENGINEERING LESSIONS LEARNED FROM DEVELOPING AND MAINTAINING WEBSITES.
In *Proceedings of the Second International Conference on Software and Data Technologies - SE*, pages 401-404
DOI: 10.5220/0001325904010404

system, what works is not a single machine running a single application but a cluster of machines running a distributed application in a multi-tiered architecture. Typically, in a multi-tier architecture, the software system is divided into layers or tiers, organized into the web-tier, middle-tier and data-tier. Each tier has a cluster of machines each running a piece of code that specializes in one or more tasks pertaining to that tier. Although each software program residing on an individual tier may appear to be simple and is hosted by a cluster of relatively inexpensive hardware, the combined effort among these simple programs running in a distributed manner can fulfil complex business requests.

The power of scalability achieved through division of task and cooperation of effort is a different design philosophy from developing traditional standalone system or client/server system with fat clients and highly coupled central server system. The computing logic in a traditional system design is accomplished by one sophisticated monolithic program. Though the monolithic program is very powerful, it is hard to scale because the logic is tightly coupled and must be one on a system that is limited in capacity. In a multi-tier architecture, the computing logic is separated into multiple pieces where each piece can be hosted and computed independently by one or more machines. In fact, machines running a copy of the code can be added or removed to meet load demands.

2.2 State Minimization Design

Besides having a multi-tiered architecture, our experiences find that minimizing state tracking enhances a web-based system's ability to provide scalable service as well. Traditional server software system prefers to keep track of client states. This is the strength of server centric system. Stateful server systems often work very efficiently due to the knowledge it needed to perform a task is readily available. However, the state oriented server design principle is only efficient at small-scale level and does not work well under large-scale web-based application.

Minimizing state tracking leans to a more scalable system. When we minimize the use of state-ful session beans in EJB system or conversational service in Tuxedo system, we find it much easier to scale, load balance and provide failover capability for the system. A system that has less state to keep track is easier to redirect user requests from one web server to another and easier to share and redistribute loads amongst the servers. In a web-based system, this is important because as the web system load increases, it means easier to add more machines and run more instances of the applications to distribute the load without having to worry about replicating user states amongst the servers. Therefore, whenever possible, avoid state tracking.

Making applications stateless, on the other hand, is not always possible. For example, without state, how can we keep track of a shopping cart? The key to scalable state tracking, when state is necessary to your system, is to consolidate the state information and save into a common place, such as database server. The database server can be viewed as a sophisticated state-ful engine that is capable of handling system crash and data recovery. When storing state data, we find that it is better to store state data, which is temporary in nature, into a different database server from that of the business data, which are durable. This allows us to apply different backup strategy base on the nature of the data.

2.3 Workflow-oriented Design

Another practice that helps with scalability in a web-based system is a workflow-oriented design. In traditional software development, a task is accomplished through a series of steps that are executed in a sequential and synchronous manner. Response is not sent to users until all the steps are finished. However, in a web-based system, this synchronous task accomplishment model may not be scalable for large number of users and requests. Instead, a workflow model based design is often helpful in many cases. In a workflow model, user requests are queued and processed asynchronously. The workflow model makes task accomplished in a pipelined manner to promote parallel execution for each step and improves the overall system throughput.

3 DESIGN FOR MAINTENANCE AND ROBUSTNESS

While scalability is important to web-based systems, robustness cannot be overlooked. The robustness of a web-based system is dependent on the maintainability of the system.

3.1 Robust Error Handling

In a multi-tiered web-based system where the code is spread over a number of machines comprising of the various tiers, it becomes more difficult to track down problems and keeping the system running

error free. As a result, it becomes crucial that we write robust error handling code in order to keep this complex system maintainable.

The most basic step in writing robust error handling code is logging. In a web-based system it is important to log, in great details, every error or exception that occurred. Each logged error message should indicate when the error occur, which program this error occurred in, any information that is relevant to the error, such as the data values that were involved at the time the error occurred. The information collected in the log will allow us to debug and improve on the system. Information in the error logs is crucial to trace and investigate incidents occurred in the system.

Besides using the error log to resolve issues and bugs form customers, a systematic analysis of the log can be a great way to find potential bugs in the system and to find resource contentions issues that should be corrected by reconfiguring various parts of the system.

Another aspect of error handling is error recovery. Distributed applications in a web-based system tend to run into non-deterministic but often recoverable errors. For example, when encountering resource contention related errors, such as a failure to obtain locks in the backend, or a failure to acquire the needed resources, we should write code to re-try instead of backing out right away. This will make the application more robust and tolerant to common errors resulted from resource contention.

3.2 Maintaining Data Integrity

Maintaining data integrity is another key for delivering robust and easy to maintain software for web-based applications. All the code running in a distributed web-based system should enforce data integrity.

Our experience shows that enforcing data integrity at the database tier is especially important. This is because external data can be imported to the database system without going through the regular web tier or mid-tier. This is typical when customers may request the business to import large amount of client data directly, bypassing the web front end. This is often done by DBA via some fast data loading programs. Another scenario is when the DBA manually repairs data via ad-hoc SQL query. Under both situations, invalid data will only be caught if the database server were performing data integrity checks. Database system has been efficiently designed to enforce declarative constraints and can even leverage constraint information to optimise queries. Therefore, enforcing data integrity check at the database layer is a simple and efficient strategy that we should always use.

3.3 Code generation Tool

It is common for most of the web-based application to rely on database, in particular relational database, as the backend data server to store and access data. We have learned that manually writing database access code can cause many maintenance problems. Therefore, we find it is worth the effort to write a tool to generate database access code automatically. This tool can generate database access code for different programming languages such as C/C++, Perl, Java. This way, whenever there are database schema changes, we can use the tool to re-generate the database access code automatically instead of manually editing potentially large amount of code. The end result is that writing tool for automatic code generation improve code maintenance and robustness dramatically. Applying this software engineering lessons to database access layer for web-based applications brings in tremendous benefit system maintenance and robustness.

4 DESIGN FOR SOFTWARE REUSABILITY

Design for reuse can improve quality and reduce cost. Software reuse in traditional software development involves the reuse of functionality provided by standard libraries, reuse of component-based software such as Java Beans or DCOM/COM components, and the reuse of design like design pattern and frameworks. In this section, we will discuss what we have learned in the area of reuse with respect to web-based software system.

4.1 Code and Component Reuse

In both traditional and web-based software system, code and component reuse is facilitated by the concept of modular design. However, in a web-based system, re-use can be applied at a much large scope. While in traditional software system, re-use is often at the level of reusing existing library or class packages, in web based system, the entire backend system can be re-used for hosting different web front ends. The idea is to reuse the solutions of past projects with minimal extension to meet requirements of new projects. In fact, many website share common components that can be developed once and reused many times. For example, web user

login module, http session state management module, data caching module, product catalogue management module, email notification module etc. The use of multi-tier approach to developing website promotes such reuse.

4.2 Business Problem Solution Reuse

It is important to be able to reuse solutions to solved business problems in the past. In a typical web-based system, the business requirements and feature requests are complex and yet the product cycle of delivering the features are getting shorter and shorter. The short-sighted solution is to develop specific solutions for one project or one business problem at a time. However, it is much more beneficial to design solutions with future reusability in mind. This means the solution should be designed as generic as possible so as to improve the chance of reusing the solution for future business problems.

4.3 Design Reuse

Design reuse is one of the most applied strategies in web-based system development. Of all the design patterns out there, the one that we find most useful is the model view controller (MVC) architecture design. MVC is a common design principle that separates the business logic, data content, and presentation. We find this is very useful to web-based system. In web-based applications, web interface software developers often interact closely with graphic designers who design the user interface (UI) appearance. These developers also interfaces with the product manager who collects feedback from website users and customers. Our experience shows that there are always late requirements and last minute changes that impact the appearance and layout of the page. By following the model view controller base design, it allows us to cope with late requirements on UI with no impact to the rest of the system.

5 EDUCATION OF WEB DEVELOPMENT

Web Programming and development are taught as undergraduate courses in computer science and information systems. Some courses focus on learning actual programming languages and tools that are widely used for website development and some courses go one step further to learn software engineering principles that are essential for website development. From our experience and lessons, we feel that teaching web software development principles, methodologies and contrast them with traditional software engineering is essential. This will let students have a profound understanding of the principles underneath the development and deployment production website. This shall be more valuable than merely grasping the software tools and products that are used for website development.

6 CONCLUSIONS

Developing, deploying and maintaining web-based system and production website is a challenging software engineering experience. It demands multi-disciplined software skills from various software domains and different thinking paradigm from developing traditional software systems. However, the underlying software engineering principles of ***designing for scalability, maintainability, and reusability*** remain the same. We, software developers, need to be aware of this and learn to apply these principles in the domains of web-based applications consistently.

REFERENCES

S. Hadjerrouit. *Web-based Application Development: A Software Engineering Approach*, SIGCSE Bulletin.

A. Ginige. Web Engineering: *Managing the Complexity of Web Systems Development.* SEKE 2002

S. Murugesan, Y Deshpande. *Meeting the Challenges of Web Application Development: The Web Engineering Approach*. ICSE 2002

Y.Desphande, A. Chandrarathna, A. Ginige. *Web Site Auditing – First Step Towards Re-engineering.* SEKE 2002

S. Murugesan, A. Ginige. *Web Engineering: Introduction and Perspsectives.*

S. Mondal, K. Gupta. *Choosing a Middleware for Web-Integration of a legacy Application*. ACM SIGSOFT Software Engineering Notes Vol 25 no 3

T. Lau, J. Lu, E. Hedges, E. Xing. Migrating E-Commerce Database Applications to an Enterprise Java Environment. CASCON 2003: 223-237