SOFTWARE INSTITUTE, NANJING UNIVERSITY, CHINA

SCORE Contest Project

QR Marks the Spot

Available at: http://218.94.159.100:6080/my_qrms/

More documents can be accessed at: http://nju.limewebs.com/docs/

Instructor:

Dong Shao	dongshao@software.nju.edu.cn

Team Members:

WANG Xing	wx08@software.nju.edu.cn
YIN Shunming	ysm08@software.nju.edu.cn
ZHAO Lu	zl08@software.nju.edu.cn
ZHANG Zu	zhang08@software.nju.edu.cn

Table of Contents

1. Executive Summary

QR Marks the Spot (QRMS) is a platform for hosting QR-code-based games. With the growing popularity of socialization and location-based games, there are more and more games that blend real and virtual worlds into gaming, single world. QRMS provides a web-based platform for creating and hosting some kinds of location-based games. The biggest difference between QR Marks the Spot and other location-based games is that in QRMS all games are amended with *QR codes* [1]. QR codes provide a way to combine location with related information. Like many social games, games in QRMS are all created by users. A user creates a game based on a game scenario provided by us, others participate in it. In order to make sure that users can host and participate in games with minimal effort, we provide easy ways to host games and search for games, and some tools to manage games. Besides, we provide some ways for users to communicate with each other.

As mentioned above, QRMS does not (and should not) provide specific games; it only provides game scenarios and the environment to support them. Users can create games based on these scenarios. We have designed three scenarios to satisfy different types of users. We call them *Inspiring Mark*, *Puzzle Solving*, and *Conquest & Defence*. They differ in game concept and difficulty. Inspiring Mark is the simplest and easiest of the three, both to the launchers and participants of games. The other two game scenarios are funnier, but they may seem more difficult for beginners. The details of the three game scenarios will be showed in *Problem Statement*. We have implemented all of the three and they are online now.

We decide that an open and convenient communication platform for users is essential. So we provide functions such as making comments on games, sending and receiving internal messages. We also provide a forum and an official blog for users to exchange their ideas and experiences.

We use the *SCRUM* [2] as our software methodology. In developing practice, we draw on some of the practices of *XP* [3], such as pair programming, face-to-face communication, TDD, continuous integration, etc. We use *SVN* [4] as our source version control system. In addition, we make use of *Zimbra Collaboration Suite* [5] as our cooperative office system (http://swi.eicp.net/).

In order to guarantee the quality of our product, we use unit test, continuous integration test, sprint integration test and acceptance test. These tests ensure both internal quality and external quality of our product from different aspects.

The main part of our project is a website platform, which is based on *Java EE* [6] and *Struts2* [7] + *Spring* [8] + *Hibernate* [9] (S2SH) framework. Besides, Conquest & Defence requires the user to install our android mobile client application. In addition, under the licenses' permit, we take advantage of a number of external services (*Google Map* [10]), libraries (*jQuery* [11], *Google Chart API* [12], etc.), components (*tinyMCE* [13]), and systems (*Wordpress* [14], *jForum* [15]).

In the process of participating in SCORE2011, we have learned some knowledge of SCRUM and XP practices. Besides, we have learned many useful technologies.

2. Developments process

2.1 What methodology have we used?

We have used agile methodology. In particular, we have used a combined SCRUM and XP methodology for the development of our project.

2.2 Why have we chosen it?

A notable feature of SCRUM is that it can provide a rapid response to changes. Considering that our project is exploratory, SCRUM is a good choice. Besides, the requirements should be iterated since we may create new ideas and requirements during the development process. What's more, our instructor is expert in SCRUM and he can instruct us on the use of SCRUM. Last, our team members are glad to use agile for a high efficiency. For the reasons above, we decided to use SCRUM in the development of our project. Meanwhile, in order to improve the quality of the product and the efficiency of the development, we also use some common practices in XP, such as TDD, pair programming, and continuous integration. Figure 1 shows our work flow in one sprint.

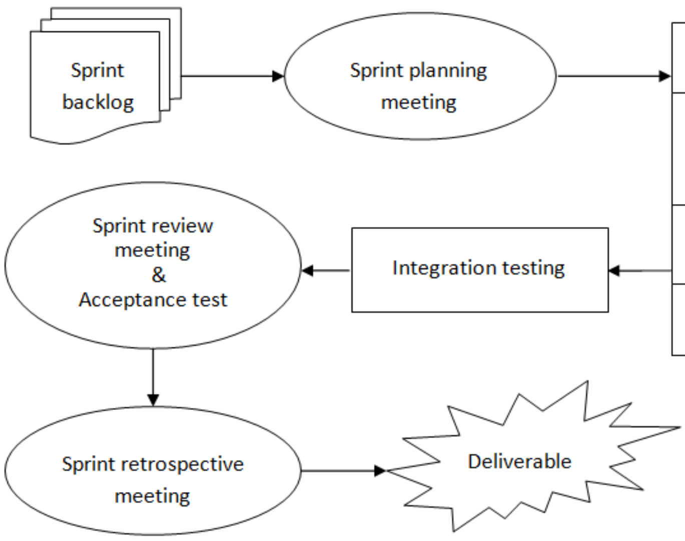

Figure 1 A Sprint Cycle

Here I will briefly introduce **some terms** in SCRUM. For detailed information of how we practiced these, please refer to *7. Management Plan*

- **Sprint**: An iteration cycle. At the end of each sprint we will get a deliverable product.
- **Sprint Backlog**: It specifies the goals and tasks of one sprint.
- **Story**: Each requirement in a sprint backlog is called a story, short for user story.
- **Task board**: It is a white board that we used to assign tasks and monitor our progress.

- **Burndown chart**: We use it to monitor our development process

The meetings we have had are listed below:

- **Daily standup meeting**: Before the beginning of new work, we have a quick meeting which requires all of us keeping standing up.
- **Sprint planning meeting**: Plan the work of next sprint.
- **Sprint review meeting**: Review this sprint after the sprint ends and we show what we have done to audience.
- **Sprint retrospective meeting**: We will retrospect all we have done in this sprint and learn some lessons and experience from it.

2.3 How did we practice XP?

Pair programming: We used pair programming. According to the features and specialization of team members, we divided team members into two groups. Each group completed their own tasks. In practice, one person is in charge of writing code, another person is in charge of overall designing, reviewing, and observing errors in the code. And we would change our combination after a period of time to spread each member's knowledge over the whole team.

TDD: Test Driven Development was adopted. We wrote unit tests and then the code. We refactored our code to do better design during the development.

Continuous Integration: We used open source software *Cruise Control* [16] as our continuous integration server. We made the server check the SVN server every 30 minutes. If something is changed, the server would update the change and run unit tests by *Ant* [17] and *JUnit* [18] according to the configuration files. If there were any errors, it would send an email to our cooperative office system.

3. Requirements and Problem Statement

QRMS provides a platform for playing QR-based games. This platform mainly solves three problems:

- Launch a game conveniently.
- Participate in a game conveniently.
- Share experiences with each other conveniently.

Users of QRMS can be divided into two roles. The first role is the launcher of a game. The other is the participant of a game. Note that when we say the participant of a game we mean those who do not launch the game but only join the game. A specific user may belong to both of roles.

3.1 Game Scenarios

We have designed three game scenarios. A user chooses one of them and launches or participates in a game of the scenario. We will introduce the three game scenarios as well as their user requirements. At last we will introduce other user requirements that do not belong to any of the three game scenarios. The reason to do so is to provide a clear view of all our user requirements. Actually, we didn't separate user requirements this way in our development process.

3.1.1 Inspiring Mark

The first game scenario is *Inspiring Mark*. The reason for designing this game scenario is to provide users with an easy way to join a game. This can prevent users from giving up at the beginning. We think this is very important to attract potential users and increase user stickiness. In this game scenario, a user puts some information into a QR code and attaches it to the location related to it. Then he or she registers some information on the website. Such information can be the name of the game, the location of the QR code, the hint to help find the QR code and so on. Other users, known as the participants, can search the website for nearby games, select one of them, and look for the QR code in the real world. When he or she finds it, he or she can get the information contained in it through a barcode reader in his or her mobile phone. What makes this simple game scenario interesting, meaningful and dynamic is that we encourage users to (play to their greatest imagination to) put inspiring things (and that is why we call this scenario Inspiring Mark) into QR code with their greatest imagination. Such inspiring things can be anything, for example, happy things, reflection on life, beautiful poetries, and introduction to the buildings nearby. There is no limitation of what types of things the user should note on QR code, so that we can provide unlimited possibilities. In fact, it is possible for users to make their own game scenarios based on Inspiring Mark. In order to contain more information on a QR code, we do not put all the information in the QR code itself; all that is stored in a QR code is a URL, which points to a web page containing the actual information. This information can be any type of rich text, which is edited by the launcher of the game. Our website is responsible for generating the web page, the URL of the web page and the QR code containing the URL. This game scenario is the simplest of the three and it provides the easiest way for users to join us.

Specifically, the user requirements belonging to Inspiring Mark are:

- Launcher of a game can easily register the information of the game and get the QR code.
- Participants of the game can easily search for games and view the details of it (except for the content of the QR code)
- Participants of a game can easily make comments on the game to share his or her experiences.

3.1.2 Puzzle Solving

The second game scenario is *Puzzle Solving*. In this scenario, users make chain puzzles with QR code; others try to solve the puzzle. Let's take an example to show how a game of Puzzle Solving is launched and participated. Tom, who lives in Paris, was going to launch a game. First, he selected three sites in advance, denoted by S1, S2 and S3. Then he registered (on the website) the first location of S1 and the contents of three QR codes (denoted by QR1, QR2 and QR3) on the website. The content of QR1 indicates the way to find S2, where you can get QR2. The content of QR2 help you find S3 and QR3, where the big surprise lies. The website generated these three QR codes for Tom; what he should do last is to put them to corresponding sites. By now a Puzzle Solving game has been successfully launched. Helen is a hardcore user of QRMS and she is travelling around Paris. She searches for some Puzzle Solving games in Pairs and gets great interest in Tom's game. She views the location of S1 and manages to get there. She refers to the hint leaved by Tom and finally she succeeds in finding the QR1 hidden there. She takes out her mobile phone, scans the QR code and finds the location of next site. She finds QR2 the same way as QR1, and finally, she finds QR3 and the big surprise in it. She is so cheerful and cannot wait to share her experiences with others on the website!

We can see that though Puzzle Solving differs from Inspiring Mark in game concepts, the way to launch and participate in these two game scenarios is very similar. So the user requirements of Puzzle Solving are very similar to those of Inspiring Mark. For the sake of saving space, we do not plan to show them here.

3.1.3 Conquest & Defence

The third game scenario is *Conquest & Defence*. This game mode simulates war and the battlefield in the whole world. The weapon our players use is an android mobile phone. The users can use our client application to conquer and defend a stronghold which is a virtual territory that maps the real geographical location. To join this game you should install our game application in your android phone, and you will find many strongholds around the world which are set by other players. If you want to conquer one and be an honorable lord, you just need to travel to the city or maybe it is in the city you stay in and find the exact location. If you are lucky enough, you will find a QR code there which is a key to break the stronghold. Now you press the conquer button and then shoot the QR code, here you complete a presentation of an attack. A stronghold can initially withstand 10 times of attacks, the last attacker will become the lord of the stronghold. Each player can only attack the same strongholds once per day. So there is a high probability that you attack a stronghold but you do not own the place. You should cheer up! Because the journey you take to find the stronghold weighs more than everything! The other way you can become a lord is that you set up a stronghold yourself. Firstly you should prepare a QR code as the mark of the stronghold. Secondly you should find a suitable location and place the QR code there, then you press the set up button, now you are required to photograph the QR code and the application will upload it to our server. Congratulations! You set up a new stronghold in the game world. Of course, your stronghold will be attacked. But you can fix the strongholds, you can only do the work of repair once per day, and each one repair can add additional 3 withstanding of attacks; of course the limit is 10. You can see the worldwide players setting up, attacking, and occupying the strongholds and the game experience they share. You will find more interesting information in your Conquest & Defence journey.

Specifically, the user requirements belonging to Conquest & Defence are:

- Participants of the game should go to the right geographical location by guidance of the client application and are able to upload the QR code there easily to finish an attack to the stronghold.
- Participants of the game should upload a QR code easily to set up a stronghold.
- Participants of the game should get access to all the news around the city he stays in.
- Participants of the game should be able to find all the strongholds other players set up around the world.
- Participants of the game should have a platform to communicate with each other.

- Participants of the game should be able to report the incorrect stronghold in case that it lacks of a QR code or for other reasons.

3.2 Other Requirements

In addition, there are some requirements that do not belong only to any of the three game scenarios. These user requirements are listed below:

- User authentication should be provided. One can sign up to be a user. A user can sign in and sign out.
- Launcher of a game can view and modify the information of the game. He or she can get the QR code again so that he or she can replace the old QR code with a new one if the old one is missing or damaged.
- A user can star interesting games so that he or she can view them quickly.
- A user can view and modify his or her profile.
- A user can modify details of the launched games.
- A user can post in our forum to exchange ideas and experiences during games.
- A user can contribute to our official blog and good articles will be posted on the blog.
- A user can make interactions with our website via our homepage in Facebook and Twitter.
- The administrator can publish news or events of the website.

4. Requirement Specification

In this section we will describe the solution to the problem described in the previous section. Requirement specification can be described in various forms, and it can be formal or semi-formal. We decide to use user manuals to describe a semi-formal requirement specification, because we think user manuals are the clearest way to describe it. Besides, we don’t have an existing formal requirement specification to display here; the output documents related to user requirements in SCRUM in our project are only sprint backlogs, and it is semi-formal. Corresponding to problem statement, we will first introduce the user manuals of the three game scenarios and then we will introduce the user manuals related to other requirements.

4.1 Inspiring Mark

4.1.1 How to launch a game?

To launch a game of Inspiring Mark you should first have an account. You can register one for free. Sign in to your account and open the home page of Inspiring Mark. Click “Launch a game” and fill in the form. In front of the form you can see a Google Map and a marker on it. Drag the marker to the location of the planned destination of the QR code and some fields of the form will be filled automatically; these fields are the longitude and latitude of the position, the country, the province/state, and the city. You can also use our “quick locate” function to locate to your destination by address quickly. Besides, if you have completed your user profile the marker will be automatically located to the city that you specified in the user profile. The next thing to do is to fill other fields. Make a meaningful name of the game. Write some hints to potential explorers. You can also provide a picture as a hint. Here comes the most important thing: you use a rich text editor to edit the content which will be viewed by the one who successfully find your QR code. You are encouraged to put “inspiring” things into it. After all these steps, click the “Launch” button and you will see a QR code in a new page. Choose a size, print it, and stick it to the destination you specified in the previous form. Till now, you have successfully launched a game of Inspiring Mark.

4.1.2 How to seek a game?

To seek a game you first open the main page of Inspiring Mark and click “Seek a game” and a form will be shown. In this form you can search games by various conditions, such as address, province, country and geographical area. A list of eligible games will be shown and you can choose one of them to view the details. Make use of this information to find the QR code. Some key information includes the longitude and latitude of the QR code, the hint, and the related picture.

4.1.3 How to make a comment to a game?

After an exciting exploration, you can make some comments to the game you have just participated in. Open the game’s page and click “make a comment”. You can choose a comment type, such as “Found it”, “Didn’t find it”, “Write Note”, and “Needs Maintenance”. You can also upload a picture. Your comment will be shown under the game’s page.

4.1.4 How to modify a game?

You can modify the game launched by yourself to add some new information or fix some errors. Open the user profile page and a list of games you have launched will be shown. Click “modify” and you will see a form with the details of the game. Do some modifications and click “modify” button to finish.

4.2 Puzzle Solving

As is mentioned in the *Problem Statement*, although Puzzle Solving differs from Inspiring Mark in game concepts, the way to launch and participate in these two game scenarios is very similar. So we do not plan to show the entire user manual of Puzzle Solving, which is very similar to Inspiring Mark; we will just tell the key differences between them. To explore the world of Puzzle Solving you should first open the home page of Puzzle Solving. Remember that in Puzzle Solving, there is more than one QR code in most cases. So when launching a game you should first select the number of QR codes in your game; then you should design the contents of the QR codes. These contents will make up the chain puzzle to guide the participants how to find the QR codes one by one. After this, you will get several QR codes instead of only one (but we really allow users to create a Puzzle Solving game with only one QR code) and you should print these QR codes and put them to the corresponding locations. When you view the details of a game, only the location of the first QR code will be shown and this is the start point of the game. Seeking a game and modifying a game are similar to Inspiring Mark.

4.3 Conquest & Defence

4.3.1 How to view the information of a stronghold?

There is a Google Map in the main interface to show all the strongholds around you. Each stronghold is noted by a marker. You can view the detailed information of a stronghold by clicking the marker on it. A new interface will take you to the stronghold with everything you need inside it.

4.3.2 How to search for strongholds?

In the bottom of the main interface, we provide a button that introduces the searching interface. In the searching interface, the players can input keywords of the stronghold in an input box. After the search button is pressed, a list of related strongholds will appear.

4.3.3 How to set up a stronghold?

First, the player should prepare a tangible QR code that contains any content the player like to add. Then the player should go to the location and place the QR code and remember to ensure that the QR code can be found and can stay there as long as it can. After this, the player should press the set up button in the menu of the main interface, then a camera interface will appear, if the player shoot the QR code correctly, there will be a hint indicates that the setting up is successful.

4.3.4 How to occupy a stronghold?

First the player should enter the stronghold-introduction interface to get the geographical information, and then the player goes to the exact location to find the QR code there. Then the player should press the button in the stronghold-introduction interface, and then a camera interface will appear, if the player shoots the QR code correctly, there will be a hint indicates that the attacking is successful. But whether you will occupy the stronghold depends on the health points of the stronghold, for a stronghold have ten health points since set up, and an attack consumes one health point, and the attacker who consumes the last health point will occupy the stronghold. A player can only attack one stronghold once within 24 hours.

4.3.5 How to reinforce your stronghold?

In the “My Strongholds” interface we provide a button that can reinforce the strongholds. One stronghold can be reinforced once within 24 hours. Each reinforcement can add three health points to the stronghold.

4.3.6 Other functions

In order to facilitate the communication among players we provide some functions like broadcast list, message list, ranking list, and news list.

4.4 Other Requirement Specification

4.4.1 How to sign up to QRMS?

Click the “Sign Up!” button on top of the website, fill in the form, and you will receive an email with an activating link. Click the activating link to activate your account.

4.4.2 How to sign in and sign out?

Click “Sign In” button on top of the website. In the pop-up window, type in your email and password and click the “sign in” button. After signing in, the previous “Sign In” button on the top of the website will be replaced by your user name and two links: “profile” and “sign out”. Click the “sign out” link to sign out.

4.4.3 How to view and edit your profile?

After signing in, click the “profile” link on top of the website to view your profile. You can also do some modification to it and click “modify” button to take effect.

4.4.4 How to recover the QR codes of a game?

Sometimes you will find it helpful to recover the QR codes of a game launched by yourself, especially when the printed QR codes are broken or missing. Open your user profile page and click “recover” link in the game list to recover them.

4.4.5 How to star interesting games and view them?

When finding an interesting game you can star it and view it later. Click “star this game” besides the title of a game in the detailed information page of the game. In the user profile page you can see all your stared games by clicking “My stared games”.

4.4.6 How to change to another game scenario?

If you are on the homepage of QRMS, just click the corresponding link. If you are on other pages, click the “Change game scenario” link on the middle of the right column. After clicking it, you will see a pop-up window. Choose a game scenario on it and you can change to that game scenario.

4.4.7 How to view a user’s basic profile?

In the game details page, you can click the link of the launcher of the game to view his or her basic profile, including his or her name, location (country, province, and city), and the games launched by him or her.

4.4.8 How to send internal messages?

In a user’s profile page, you can see a link: “send him/her a message”. Click this link, fill in the title and the content of the message, and click the “Send” button. One can view all his or her internal messages on his or her user profile page.

4.4.9 How to exchange your ideas and experiences?

There are at least four ways to exchange your ideas and experiences. You can make a comment to a game to share your experiences. You can also send internal messages to other users. Besides, you can post in our forum to exchange your ideas. What’s more, you can contribute to our official blog to share your exciting and interesting experiences.

5. Architecture Design

The system is divided into three main parts: the server, the web client and the mobile client. The server provides data storage, business logic and interfaces for the clients. The clients pass user inputs to the server and show to users the results returned from the server to users. Figure 2 shows the overview of the system architecture.

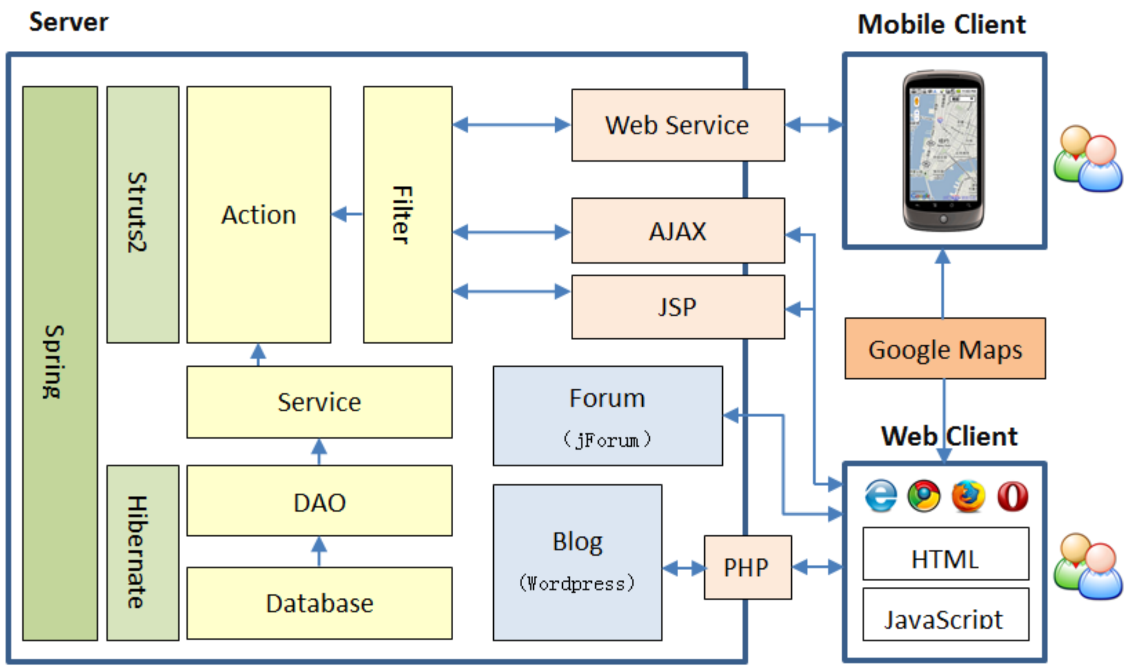

Figure 2 Architecture design overview

At the server side, we used Java EE and Struts2 + Spring + Hibernate (S2SH) framework to implement system architecture, except for our forum system and blog system. We used these techniques because the team members are familiar with Java programming language. Moreover, S2SH framework provided us with maximum flexibility. Hibernate shields the underlying mechanism for database access. Struts2 provides a clear way to manipulate business logic. Spring, which is a well-known open-source IOC framework, makes sure that the components of the system are loosely-coupled. Compared with widely-used PHP technique, using S2SH framework can result to relatively low development efficiency; this is a shortcoming of using S2SH framework.

We did not implement our own forum and blog system; instead, we used open-source jForum and Wordpress. There were two reasons for doing so. First, we did not have enough time to implement our own forum and blog from scratch. Second, jForum and Wordpress are mature products; we did not need to reinvent the wheel.

5.1 Server

As is mentioned above, we used S2SH framework on server side. The database access is encapsulated into DAO (Data Access Object). Tables in the database are mapped to beans, which are POJOs (Plain Old Java Objects), via hibernate mapping files. Services are based on DAOs and encapsulate basic services that will be used by actions. Actions encapsulate business logic. The data stream will be pre-processed by filters to provide some convenient features of AOP (Aspect-Oriented Programming). The server interacts with the clients in three ways. The first is JSP (Java Server Pages), which is the main interface. On the server side, JSP generates in the server side the contents to be passed to web clients. The second is AJAX. This widely-used technique in Web 2.0 websites can be used to make interactions from clients to server asynchronously. The third is web services. The web services we used are based on our S2SH framework, which is lightweight and *REST* [19]-like. These web services are used to make interactions form server to mobile clients. Compared with *SOAP* [20], our REST-like web services are easier to use and implement.

5.2 Web Client

Web Client provides a way for users to interact with server through a browser. JavaScript is widely-used here. Popular jQuery library and some jQuery plugins are used to provide a good user experience. Web pages are well designed and organized, also aim to provide a good user experience.

5.3 Mobile Client

Mobile Client is used to support Conquest & Defence, the third game scenario. It is a program running on the Android platform. It interacts with server via web services provided by the server.

5.4 Architecture Pattern

Obviously, the system uses MVC pattern. This is the direct result of using S2SH framework. The model tier is implemented by Hibernate. Hibernate relates beans to database tables with mapping files. Such mapping files are represented in the form of XML. The view tier is implemented by JSP. The control tier is implemented by Struts2; it controls the model tier and the view tier. In addition, we used *sitemesh* [21] framework in view tier to provide a unified style of user interface.

6. Project Plan

To achieve an excellent product as much as possible, making a scientific and effective plan is very important. We will introduce our overall plan at first and describe the plan of each phase. Then we will show the results of the plans. At last, we will explain how we evaluate the execution of the plans.

6.1 A Brief Plan

Considering that QRMS is an exploratory project and we didn't have full and clear requirements at the beginning, we needed to explore requirements. Thus, at the beginning of the project, we could just provide a rough plan.

We made our project plan on the basis of the definition of our product, our schedules, and potential risks. We list these things in the following table:

Goal of the Product	A site for creating and hosting QR-based games with supporting environment, such as convenient tools and platform for communications. The site provides three game scenarios.
Start / End time:	July, 2010 ~ Dec, 2010
Periods Available:	2010.7.15 ~ 2010.8.20; 2010.11.1 ~ 2010.12.31
Possible Risks	1. We can devote all our daytime to the development of our product in the summer while the time we can devote to the project will decrease at the fall semester. 2. Because that we don't have a lab and we are busy with lessons at the beginning two months of the fall semester, we do not have enough time to develop our project. 3. In SCRUM, a sprint is a development stage and it shouldn't be broken. So we must choose periods during which all of us are free for at least two weeks. 4. We are not familiar with some tools, such as SVN and Cruise Control. 5. Because we adopt SCRUM, monitoring our work is difficult. We may not catch up with the schedule.
Policy to these risks:	1. We can work in the evening in the fall semester. And we can work at weekends. 2. We arrange our work after the first two months of the fall semester. 3. We choose a period that we can ensure we won't be disturbed. 4. We should spend a lot of time reviewing our work before Sprint 3. 5. Before our project starts, we should consult our instructor. 6. We decide to use burndown chart to monitor our process.

We made a rough schedule as follows:

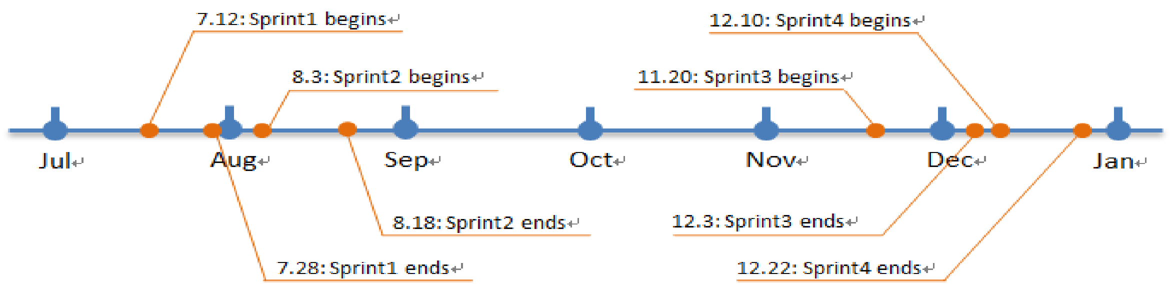

Figure 3 A rough time table

Rough sprint goals:

- Sprint 1: The development of the framework and basic functions of one game scenario
- Sprint 2: Enhance the functions and beautify the web pages
- Sprint 3: Develop the other two game scenarios
- Sprint 4: Improve user experience and the three game scenarios

6.2 Detailed Plans

Because we used SCRUM, we organized our development in units of sprint. At the sprint planning meeting of each sprint, we set goals and tasks of this sprint. Based on each sprint's meeting reports, we list the plan of each sprint in the following forms. You can view all our meeting reports at:

http://nju.limewebs.com/docs/index.php?option=com_content&view=category&id=37&Itemid=58

Sprint 1:

Sprint Goal	Accomplish all the tasks defined in sprint backlog, mainly the base establishment of our website and functions of the Puzzle Solving game scenario
Start / End time:	July 7, 2010 to July 30th,2010
Sprint Review Meeting:	July 30th, 2010
Possible Risks:	1. Team members are not familiar with S2SH. Our process may not catch up with the schedule. 2. We may encounter technical problems about web programming 3. We are not familiar with writing test cases for web. Practicing TDD may encounter problems.
Policy to these risks:	1. We can consult our instructor and learn in our leisure time. 2. We can consult our instructor and classmates who are familiar with related technologies. 3. We can learn from books and Internet or consult our teachers.

Sprint 2:

Sprint Goal	Accomplish all the tasks defined in the sprint backlog, such as beautifying the website and enhancing the details
Start / End time:	August 4, 2010 to August 12, 2010
Sprint Review Meeting:	August 12, 2010
Possible Risks:	1. Time is limited. We only have no more than 8 days. We may complete only part of the stories. 2. Our instructor will go out for a meeting for some days. We cannot consult him timely if we encounter problems.
Policy to these risks:	1. We don't arrange a lot of work in this Sprint and we should ensure efficiency 2. We can communicate with our instructor with E-mail. And we can consult other teacher, too.

Sprint 3:

Sprint Goal	Accomplish all the tasks defined in the sprint backlog, such as the functions of Inspiring Mark and Conquest & Defence.
Start / End time:	Nov 22nd, 2010 to Nov 3rd, 2010
Sprint Review Meeting:	9.00 am, Dec 3rd
Possible Risks:	1. During the daytime, we have classes and the time we can provide may be less. We may not catch up with the schedule. 2. In Android system development, we may encounter technical problems. 3. Cruise Control needs to transfer from one computer to another other because our lab is changed. 4. We blow over what we do in the summer and some knowledge about Web Programming. We may spend a lot time if our new work needs to change previous code.
Policy to these risks:	1. We should ensure our efficiency and we guarantee that we devote our weekends to it. 2. We can consult our classmates who have experience in Android development. And we can learn from books and Internet. 3. The team member who is responsible for monitoring it needs to spend time doing it before our work starts. 4. We should review our work first before we do new work.

Sprint 4:

Sprint Goal	Accomplish all the tasks defined in the sprint backlog, such as enhancing the functions of the web client and finishing Conquest & Defence game scenario.
Start / End time:	Dec 6th, 2010 to Dec 17th ,2010
Sprint Review Meeting:	Dec 18th, 2010
Possible Risks:	Nobody of us is familiar with Photoshop. We may encounter difficulties when we need to beautify web pages and the Android client.
Policy to these risks:	We consult our classmates and learn some Photoshop knowledge from books.

6.3 Evaluation of Plan

During the development process, we used burndown chart to provide process visibility so that we can monitor the execution of the plan. For detailed information of how we did this, please refer to *7.6 How did we monitor our project process?*

After each sprint, we needed to evaluate the execution of the plan. We did this in the sprint review meeting. For detailed information of how we did this, please refer to *7.4 How did we conduct meetings in SCRUM?*

7. Management Plan

In this section, we will introduce our management plan from the following aspects: project management, product, process, and staff.

7.1 How did we communicate with each other and manage our documents and source code?

Due to the situation that we worked together in the lab, we can communicate face-to-face, so it was simple and convenient to share our ideas. Besides, Zimbra enabled us to share our documents. The usage of SVN simplified the source code management and version control. SVN also logged what each member had done and when there were bugs or errors, we could easily find responsible person.

7.2 How did we define our roles?

According to the characteristic of our project, face-to-face communication with customers defined in SCRUM was not suitable for us. We redefined roles and responsibility based on common SCRUM practices. We selected one team member as the product owner. The total goals of our project were discussed by our team and the product owner was responsible for the final confirmation. We selected our team leader as the SCRUM master. He held daily standup meetings, sprint planning meetings, sprint review meetings and sprint retrospective meetings.

7.3 How did we organize our work?

We divided our work into four sprints and used sprint backlogs to direct our work in a sprint.

Sprint: According to available time that can be devoted and the scale of the project, we decided to split the project into 4 sprints; two at the summer vacation, and two at the fall semester. Each sprint lasted for about two weeks.

Sprint Backlog: Considering the actual situation of our project, we did some adjustment to the common practices used in SCRUM. Common practices suggest that there should be an overall product backlog at the beginning and we just choose some stories from it every time a new sprint starts. Because the requirements were explored by ourselves, an overall product backlog was not possible for us. We just made one sprint backlog at the beginning of the sprint. Since we were inexperienced in estimating, we used card estimation method. In practice, everyone first held up a card that showed his estimation in finishing the story, and then gave his reason. Finally, we used the average time as the time this story would cost. We also sorted these stories according to their importance.

7.4 How did we conduct meetings in SCRUM?

We held some common meetings used in SCRUM:

Sprint planning meeting: Firstly, our product owner introduced the sprint goal. Then, we decided the length of the sprint and when and where we had the review meeting. Each team member estimated the time that he could devoted in the next sprint and we made clear what each story meant, split the story into smaller tasks. At last, we confirmed what we should do in the next sprint based on our time and the importance of each story.

Sprint review meeting: At the end of each sprint, we invited our teachers and some classmates who are potential customers of our product to watch our demo and listen to our introduction about original sprint plan and accomplished things. We would also collect feedback to improve our product.

Sprint retrospective meeting: At the end of the review meeting, our team members and instructor would hold a retrospective meeting. We reviewed the last sprint, and summed up lessons and experience.

Daily standup meeting: In sprint 1 and 2, we held our standup meeting at 10.00 am in the lab. In sprint 3 and 4, considering that we had classes in the morning and afternoon, we held the standup meeting at 9:00 pm in the evening. It usually took 10 to 15 minutes. All team members described what he did yesterday and updated the task board. And each one should tell what he planned to do today and anything he wanted to tell others, for example, what changes he had made to source code, some suggestions or difficulties he had come about. There were three questions each team member should answer:

1. What have you done yesterday?
2. What are you planning to do today?
3. What else do you want to tell others?

7.5 How did we assign tasks?

In our team, two have experience in Android development, and the rest two have experience in web development. When we did pair programming, we took this into consideration.

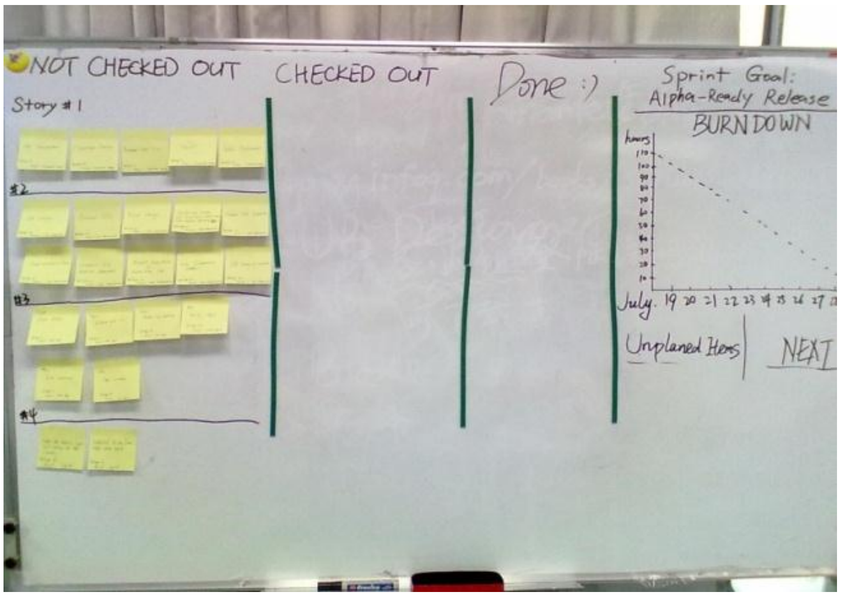

Figure 4 Task Board

We used a task board to help assign our work. We put unfinished stories onto the “NOT CHECKED OUT” column. Stories that we were planning to do were put onto the “CHECKED OUT” column. And completed stories were put onto the “DONE” column. Considering the small area of whiteboard, we didn’t put all stories on it. In addition, in the upper right corner you can see the burndown chart, whose detail function will be described soon. We assigned our daily tasks during the stand-up meeting. We updated our task board. Each team member moved the stories that he had completed to the DONE column and moved what he would do today from the NOT CHECKED OUT column to the CHECKED OUT column. In order to achieve a high efficiency, tasks were not totally assigned by the team leader, but were chosen by each member’s interest.

7.6 How did we monitor our project process?

We mainly used burndown chart to monitor project's process. Every group estimated its yesterday's workload after standup meeting, and then updated burndown chart. Through checking burndown chart we can uncover some problems about the projects. Too fast dropping of burndown chart means wrong estimated workload or a too fast developing, which may lead to poor quality of the project; too slow dropping means inefficient developing or problems encountered, which may lead to project delay. All these problems should be handled timely with some measures, such as paying more attention to quality and slowing down develop speed.

As the figure showed below, when we updated the burndown chart at August 8th, we found that our workload was almost zero. After we listened to each other's explanation, we knew that one pair group came up with technical problems. For the other pair, one member didn't come, and the other couldn't work alone. Thus, the assigned task wasn't completed. Figuring out the problem, we consulted our instructor for technical problems and solved it. Then curve became normal.

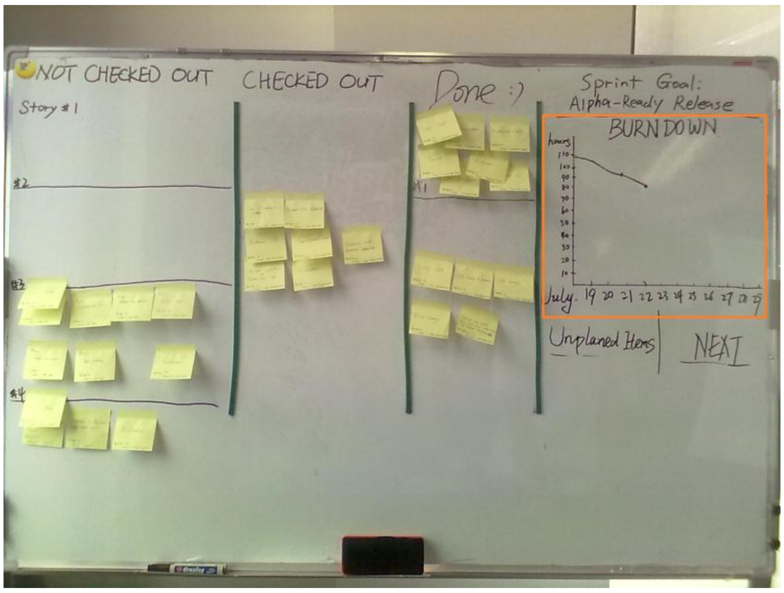

Figure 5 Burndown Chart

7.7 How did we manage bugs?

Bug tracking system

To spare enough time for fixing bugs, when we calculated the time we could devote, we didn't use all our available time but left a buffer time. When we found bugs, we noted them in an excel chart. We sorted bugs according to their priority. To those with high priority, we would fix them as soon as possible. To those with low priority, we would fix them later.

7.8 How did we manage personnel?

Our agile team was self-adaptive. We chose self-adjust method in personnel management. We didn't set down strict rules and regulations. When one team member felt tired, he could rest for a while or did some other work. Those ensured team harmony and efficiency.

8. Implementation

8.1 Fuzzy search

The users can search for games based on various conditions, one of which is address. Different users have different language styles. Take Washington as an example, some users tend to call it Washington DC. Another example is Nanjing (a city in China); some users prefer to call it Nanking. We hope that our system can give all correct results when users are searching for games by address. At first, we planned to create such a database that contained the cities and their aliases. We searched the Internet for those data, but we soon realized that relative information was hard to get, so we dropped this idea. After we studied the Google Map API, we found that Google Map provides the service of *Geocoding* [22] which can convert an address into geographic coordinates. So, we came up with an idea that we could firstly call the service of the Google Geocoding when users posted request. And then we compared the geographic coordinates with the data in our database and displayed the information of the games within a certain range (the default setting is 100km around, and a user can specify this number). Figure 6 shows the process of fuzzy search.

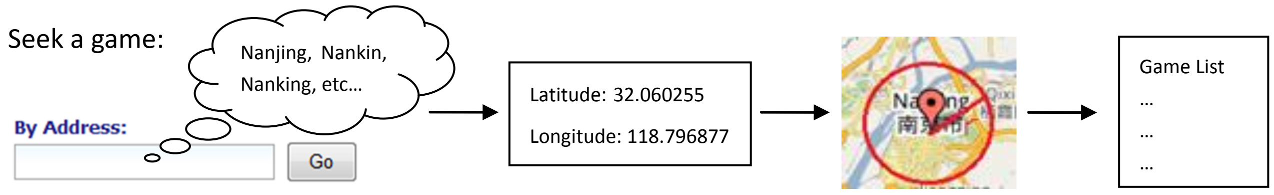

Figure 6 The process of fuzzy search

8.2 Interaction with Android System

An important part of the product is the mobile client for supporting Conquest & Defence. Because all data are stored in the server, it is a key design decision to determine how the mobile client communicates with the server. The socket was first excluded by us because the server is based on tomcat and it is an http server, not supporting the classic socket programming. The common practice to communicate with a website is to use web services. Based on what we now know, there are two types of web services; one is lightweight, for example, RESTful web service; another is heavyweight, like SOAP. We chose a lightweight solution, which is a REST-like web service, because it is easier to implement and it saves time. Besides, we make use of our current S2SH framework and do some modifications to the common use of it by using our own result type. By doing so we implemented a simple and flexible way to make communications between the server and the mobile client.

9. Validation and Verification

In order to verify that our product was bug-free, we used unit tests as well as some kinds of global tests. Besides, we used a continuous integration tool to support some automatic test. Below we will describe how we did these tests.

9.1 Unit Test

Because we have used TDD, the first task of most stories is unit test. Before we wrote any running code, we would write their test cases. These test cases were evolutionary; if there were important changes in source code, related test cases should be changed. Technically, we used JUnit and mock to help write the test cases. Every time we modified the old code and added new code we would run all the test cases again to test if new bugs were taken to the system.

9.2 Continuous Integration Test

Continuous integration was used to support automatic test. By doing some configurations on the continuous integration server, all test cases would be run when a change was committed to the SVN server. Easy access to the results of the tests is provided via web pages, emails and RSS.

9.3 Sprint Integration Test

In order to make sure that our product was runnable, at the end of the development of every sprint, team members would do integration testing on all new functions of this sprint and some functions of the previous sprint to ensure that our product was runnable.

9.4 Acceptance Test

In order that the product was satisfied with the requirements, we demoed all we had done in the current sprint and compared them with our primary sprint backlog in the sprint review meeting. At the meeting we decided whether these user stories were implemented correctly.

10. Outcomes and Lessons Learned

10.1 Implemented Modules

We have finished all the parts of the system mentioned in this document, including the server, web client, and mobile client. We have implemented all the three game scenarios.

Here are some snapshots of our website and mobile client.

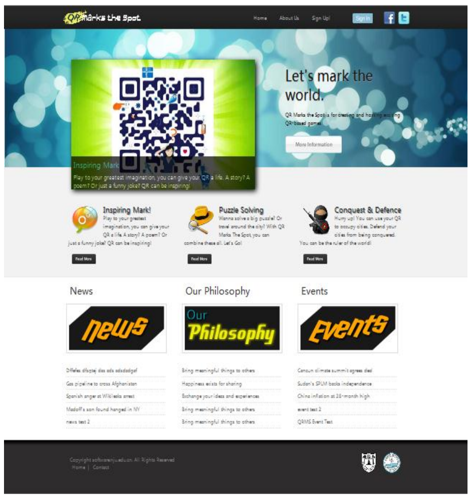

Figure 7 Homepage of our website

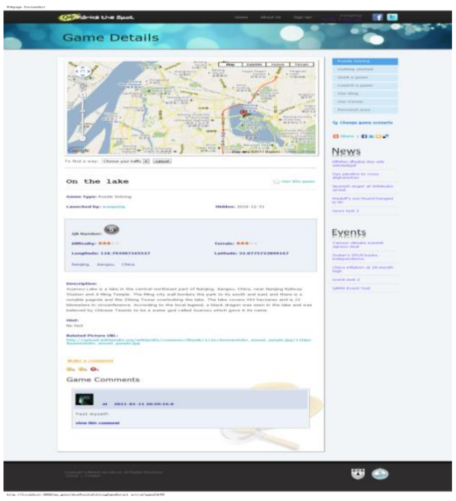

Figure 8 Detailed information of a game

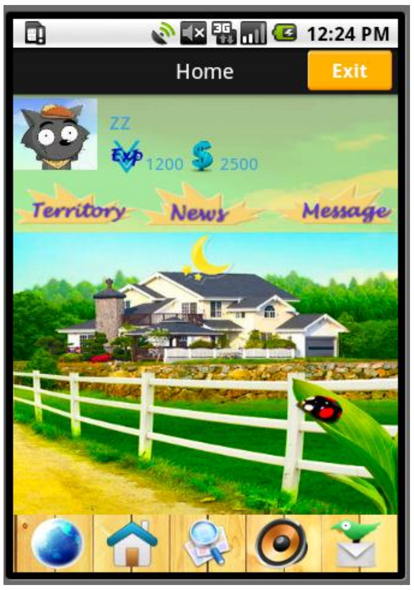

Figure 9 An interface of the mobile client

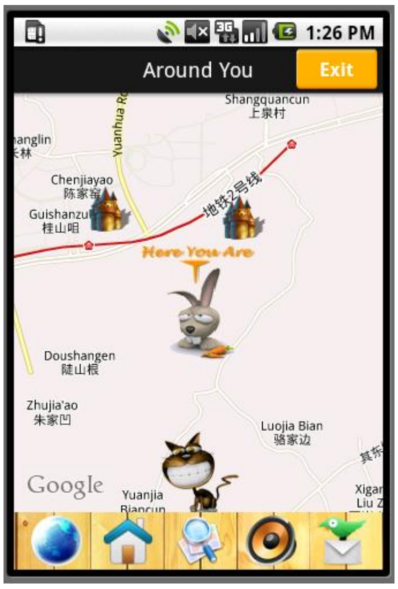

Figure 10 Another interface of the mobile client

10.2 Source Code Statistics

We divided our project into two sub projects here. One is the server and the web client, another is the Android client. We survey our two sub projects:

	JSP	Configuration files	JavaScript	Java	CSS	**Total**
Web Client& Server	10800	2500	600	9000	1500	*24400*
Android Client	0	1400	0	2700	0	*4100*
Total	*10800*	*3900*	*600*	*11700*	*1500*	*28500*

10.3 Documents and Photos of the Project

We have uploaded our documents and photos of the project to the Internet. They showed our process of development and practices in SCRUM. You can access them at: http://nju.limewebs.com/docs/

10.4 The Outlook of Our Project

Our product has been on-line and is now under β-test.

You can access it at: http://218.94.159.100:6080/my_qrms/

There is a trend that location-based games are becoming popular and now QR codes are used in various fields. So we think location-based games based on QR codes will become popular. We will possibly add new game scenarios according to users' feedback in the future. Besides, we will develop an iPhone client for our project to attract more users if there is enough time.

10.5 Lessons Learned

We have faced some challenges in our development. At the beginning we were not very familiar with some technologies so we could not make an accurate evaluation of how much time was needed by each user story, and this caused some chaos, for example, near the end of the first sprint we were surprised to find that a lot of functions had not been implemented and so we had to increase our working load. Fortunately, we gained enough experience after the first sprint and this problem went away in the following sprints. This tells us that an accurate evaluation of the working load is very important. Another challenge we encountered was that we once noticed that we had different understanding of some user stories. This was caused by lack of sufficient discussion over the details of each story. So we increased the amount of discussion in the following sprints to make sure we have the same understanding of all user stories. There are many such challenges and we have gained valuable experience in the process of solving them.

11. References

[1] QR Code -- http://en.wikipedia.org/wiki/QR_Code
[2] SCRUM -- http://en.wikipedia.org/wiki/Scrum_(development)
[3] XP -- http://en.wikipedia.org/wiki/Extreme_Programming
[4] SVN -- http://subversion.apache.org/
[5] Zimbra Collaboration Suite -- http://www.zimbra.com/products/zimbra-collaboration-suite.html
[6] Java EE -- http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition
[7] Struts2 -- http://struts.apache.org/2.2.1/index.html
[8] Spring -- http://www.springsource.org/
[9] Hibernate -- http://www.hibernate.org/
[10] Google Maps -- http://maps.google.com/
[11] jQuery -- http://jquery.com/

[12] Google Chart API -- http://code.google.com/apis/chart/
[13] tinyMCE -- http://tinymce.moxiecode.com/
[14] Wordpress -- http://wordpress.org/
[15] jForum -- http://jforum.net/
[16] Cruise Control -- http://cruisecontrol.sourceforge.net/
[17] Ant -- http://ant.apache.org/
[18] JUnit -- http://www.junit.org/
[19] REST -- http://en.wikipedia.org/wiki/Representational_State_Transfer
[20] SOAP -- http://en.wikipedia.org/wiki/SOAP
[21] Sitemesh -- http://www.opensymphony.com/sitemesh/
[22] Geocoding -- http://code.google.com/apis/maps/documentation/geocoding/