Project Evaluation & Self-Reflection

Yang LI
Zhongjin LUO
Guohui YANG
Yiqun LIN
Yirui WANG
Xinying WU

School of Software Engineering
Tongji University
Group 4

June 26, 2017

Contents

1 Project Achievements

Reviewing the project from the end of semester, as we produce a large amount of document about our system, which contains a brief introduction and following key points:

Use Case Model

In the use case modeling, we have a global use case and several specification of use cases, which represent missions and stakeholder goals in the whole systems and the detailed scenario. Thus, it can provide the shortest summary of what our system will offer, and this project planning skeleton, to be used to build initial priorities, estimates, team allocation and timing. These use case constitute a powerful, user-centric tool for the software requirements specification process. It ensure that what has the real business value and the user really want is developed, not those trivial functions speculated from a developer or system inside perspective.

Supplementary Specification

The Supplementary Specification provides an overview of the entire document. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this Supplementary Specification. It also captures the system requirements that are not readily captured in the use cases of the use-case model.
We discuss this part in many ways, like security, performance, data storage and computing, maintain and more. As the complex system have many additional functions, we have dived into details about the dilemma and come up with some feasible solutions.

Architectural Analysis

Architectural Analysis reflects early decisions and working assumptions on implementing the Vision, as well as decisions concerning the physical and logical architecture and non-functional requirements of the system. It is produced by the designer, often in collaboration with the stakeholders. It takes the form of an informal, rich picture, story-board, or iconic graph. Conceptually, it illustrates the essential nature of the proposed solution, conveying the governing ideas and including the major building blocks. The level of formality of the architectural overview is project dependent. Take our system for example, in this large, high-ceremony project, it may be necessary to capture the architecture overview in the appropriate sections of the Software Architecture Document, so that it may be formally reviewed.
We have designed the package diagram and divide our system into six layers, Presentation

Layer, Service Layer, Controller Layer, Business Logical Layer, Domain Layer and Data Layer. Then, we analysis every layer's thing, and what it play in our whole system. It makes our project more clear and easy to understand.

Analysis Model

In this part, we have a global class diagram, every scenario's sequence diagram and communication diagram. The class diagram support the specification of the concept of classes known from object-oriented programming. The objective of class diagrams is to specify classes and their operations as well as the dependencies between them.
As for the sequence diagram, it can help specifying the link between a use case and the methods and classes defined in class diagrams. It is therefore important to ensure consistency between these use case and class diagrams.
Communication diagrams illustrate object interactions in a graph or network format, in which objects can be placed anywhere on the diagram.
However, it has been some difference between this similar diagrams. Sequence diagrams have some advantages over communication diagrams. Perhaps first and foremost, the UML specification is more sequence diagram centricmore thought and effort has been put into the notation and semantics. Thus, tool support is better and more notation options are available. Also, it is easier to see the call-flow sequence with sequence diagramssimply read top to bottom. With communication diagrams we must read the sequence numbers, such as "1:" and "2:". Hence, sequence diagrams are excellent for documentation or to easily read a reverse-engineered call-flow sequence, generated from source code with a UML tool.
But on the other hand, communication diagrams have advantages when applying "UML as sketch" to draw on walls because they are much more space-efficient. This is because the boxes can be easily placed or erased anywherehorizontal or vertical. Consequently as well, modifying wall sketches is easier with communication diagramsit is simple to erase a box at one location, draw a new one elsewhere, and sketch a line to it. In contrast, new objects in a sequence diagrams must always be added to the right edge, which is limiting as it quickly consumes and exhausts right-edge space on a page; free space in the vertical dimension is not efficiently used. Developers doing sequence diagrams on walls rapidly feel the drawing pain when contrasted with communication diagrams.
Likewise, when drawing diagrams that are to be published on narrow pages, communication diagrams have the advantage over sequence diagrams of allowing vertical expansion for new objectsmuch more can be packed into a small visual space.

Subsystems

Informally speaking, a subsystem is a cross between a package and a class. It behaves like a package in that it groups other model elements, but it also behaves like a class because it has a specified behavior.
To say that a subsystem behaves like a package implies that it has an inside view and an outside view. Inside the subsystem are a number of other model elements(e.g., classes) that

collaborate to fulfill the behavior for which the subsystem is responsible. From the outside of the subsystem these classes are not necessarily visible; in fact, it is a very good idea to make them invisible. From the outside, inner details of the subsystem are not interesting, and it is best to treat the subsystem as a single unit.
As we design following subsystems: order management, user management subsystem, package sign subsystem, payment subsystem, tracking subsystem and return products subsystem. It help architects greatly in realizing many of a software system's architecture qualities such as insulation from change, replaceable implementation, dynamic replacement, abstraction, reuse and more.

Mechanisms

A building is made simpler and more harmonious by the conformance to a pattern of common features. It is made simpler by the presence of four common mechanisms that apply consistently throughout the language.
In our system design, we take communication mechanism and security mechanism in detail. In the first one, we design the API between the front-end and back-end, and use JSON to communicate. While in the security mechanism, we use SHA and MD5 encryption to make sure the security. Those mechanisms have their own job in our systems.

User Interface

As a direct way to show to the interface, User Interface plays a important way, We have designed several views including login, main, search, order and account page. Since mobile UI is quite different with web UI, so we have two solutions, a desktop UI for PC to improve the user experience.

Prototype

The Prototyping Model is a systems development method in which a prototype is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed. This model works best in scenarios where not all of the project requirements are known in detail ahead of time. It is an iterative, trial-and-error process that takes place between the developers and the users.
As our prototype, it can improved and increased user involvement, and reduced time and costs. We take our connection with Oracle Database and Encryption for example.

2 Limitations of Design

There still are several limitations in our design. The security mechanism is not perfect. The final design should prevent cross-domain access and SQL injection. For the former, we will take some detailed action for consideration, such as look for the header, use and .htaccess file, and use tokens. As to the latter, we prepare statements with parameterized queries, stored procedures white list input validation, escape all user supplied input and so on. There may be a lot program vulnerability in security, we will work on it all the time.
Besides, reality is also a big part, that is the design should be closer to the reality, and we need to do some detailed research to figure out the exact user scenario. In addition, maybe we can simplify the overall architecture. Some subsystems are considered not comprehensive enough. For example, in the payment subsystem, customer can pay by Apple Pay, check and so on. With wave of smart home, we can combine it with some IoT things to provide a better user experience.
Thus, some potential limitations can be fixed soon is under our calendar, we will discuss details in the several chapters.

3 Potential Improvements

3.1 Use Cases

Requirements change all the time. In fact, in addition to the original given scenarios, from which we extracted use cases, there are more potential use cases still. For example, considering efficiency, in some densely populated areas like school, residential areas, people may hope that they can take their packages at any time they want. Therefore, self-help package signing system is in need. It allows customers to get their packages at a nearby centralized package storage, without worrying about time constraints. In this case, we have to modify our use cases model and related designs to accommodate new requirements.
That also raises another issue:

3.2 Coupling and Cohesion

Generally, low coupling and high cohesion is a critical principle we should follow in the process of object-oriented analysis and design. Otherwise, the more and more complex systems will lead to a maintenance nightmare. In our analysis and design, however, we probably did not meet this rule strictly. Take our subsystem design as an example, interfaces design is not high cohesion.

3.3 Security Mechanism

As to an e-commerce platform, security is always the primary consideration at most of the time. So far we have created a security mechanism, but there are still something that we should think carefully, that is, such following capabilities:

1. **Specification of physical exposure**: Can someone access and impact the component? For example, can someone replace the memory?
2. **Specification of trust**: Is a component verified so that it can be trusted (no propagation of fault or vulnerability)?
3. **Security levels and domains**: What are the criticality levels and domains of components? Does a component handle several domains or levels?
4. **Specification of encryption mechanism**: Is data encrypted when it is transported between two components? Is the encryption mechanism strong enough?

5. **Specification of authentication mechanism**: Does the system use an authentication mechanism (user/password, password, etc.)? Can we trust it?

We only considered the encryption and authentication mechanism here. We can purpose improvements plans from other aspects.

4 Open Issues

4.1 Integrality

Although we have design all pages UI of our system, but we just achieve part of system function. We haven't fitted all screen resolutions and all browsers. In deed, we have to design Web UI to meet customer need. And we also hope that develop native app for iOS and Android, because Web View is a bad user experience in mobile.
We havent designed our database, just store our data in a simple table without thinking about redundancy and performance. Database performance would affect the whole system performance and Database redundancy would lead to lead to data inconsistency. And in the class, we found that other group use distributed database to guarantee performance, we all think that it is a superb choice to add this concept to our system to improve performance.
As for payment, we just use third-party payment service, which require customer use third-party payment to pay for their order. And for enterprise customer, they pay for their order manually every time. We haven't add our own paying wallet. If have, we can provide a more convenient method for customer, and we can also allow enterprise customer choose to pay automatically, which will reduce their workload.

4.2 Separability

Now, in our system, all customers(including personal customers and enterprise customer) use the same application. We found that the functional requirement of personal customers and enterprise customers is different. For example, enterprise customers has many packages to send, while personal customers have little packages. So, we need to improve our system to make enterprise customers manage their packages more easily.

4.3 Security

Now, we can users can log in via WeChat or Facebook, it is unsafe when a user's social account is stolen. We haven't record users' usual IP address and usual equipment. We found that if we bring face recognition and fingerprint recognition to our application cannot only improve security, but also improve user experience. We haven't add real name authentication to our system, which is unsafe for user's fund.
We haven't work out to prevent SQL injection, XSS, CSRF. And evildoer will use these

faultiness to invade our database. We also haven't use IP limitation to prevent malicious visitors.

4.4 Concurrency

We haven't considered concurrence access problems. In our system, if a large number of customers use our system in the same time, our system may crash. And system crash will lead to data insecurity and data missing.

5 Development Plan

As now we have the documents about our Onion Express system, the development plan can be pretty clear. And we use agile software development as follows:

- Customer satisfaction by early and continuous delivery of valuable software
- Welcome changing requirements, even in late development
- Working software is delivered frequently(weeks rather than months)
- Close, daily cooperation between business people and developers
- Projects are built around motivated individuals, who should be trusted
- Face-to-face conversation is the best form of communication(co-location)
- Working software is the principal measure of progress
- Sustainable development, able to maintain a constant pace
- Continuous attention to technical excellence and good design
- Simplicity the art of maximizing the amount of work not done is essential
- Best architectures, requirements, and designs emerge from self-organizing teams
- Regularly, the team reflects on how to become more effective, and adjusts accordingly

Development Plan

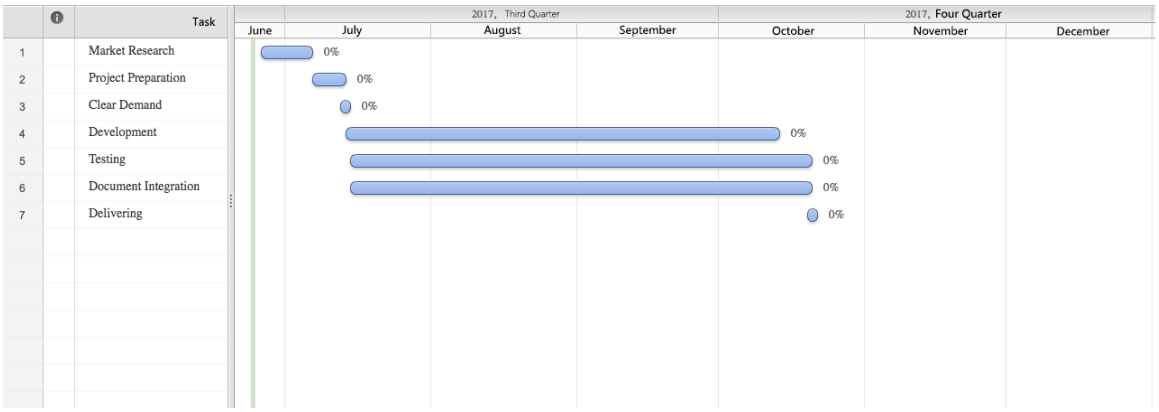

Figure 5.1: Gantt Chart of Development Plan

Test Plan

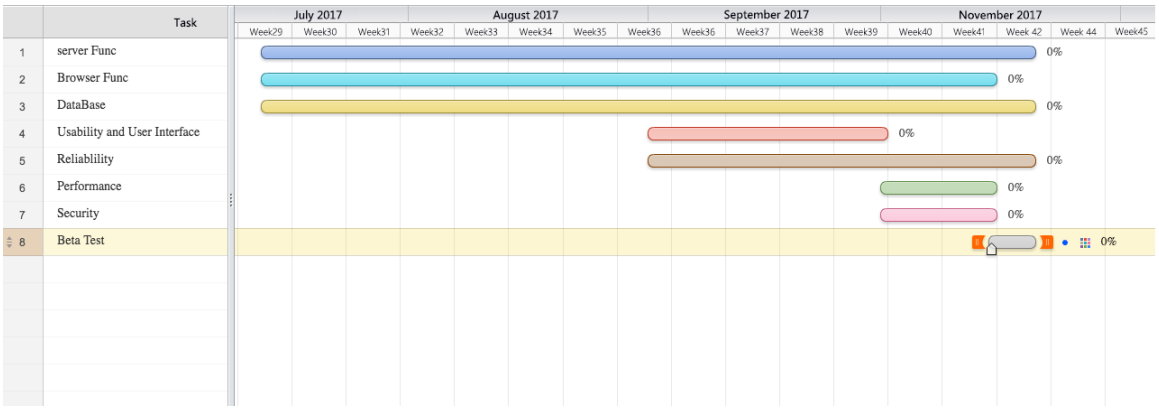

Figure 5.2: Gantt Chart of Test Plan

Detailed Development

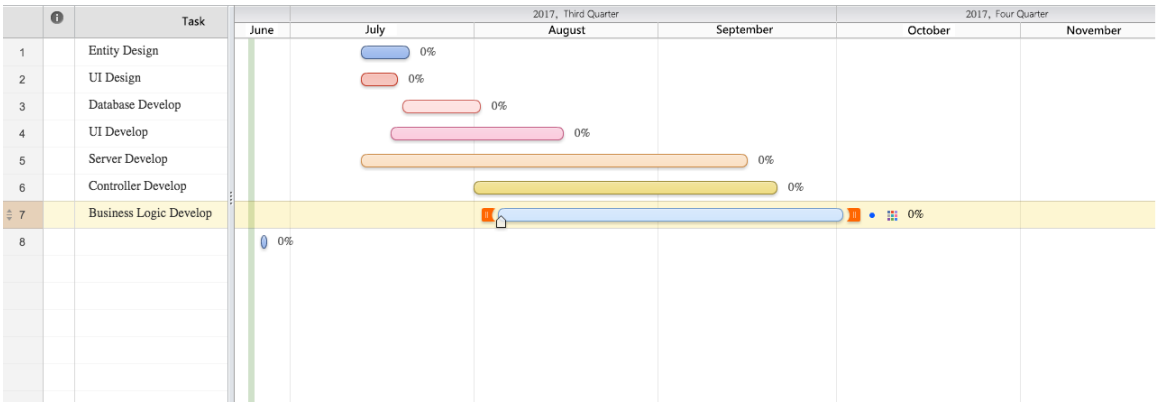

Figure 5.3: Gantt Chart of Detailed Development Plan

6 Course Reflection

Yang LI

As the LaTeX file open sourced in GitHub and everything comes to an end, it is time to review all this semester that we have learned and how it works in my daily project.
The first thing is that as a team leader, I have to organize our teammates to assign different part to everyone according to their strengths. Thus, we have meeting weekly to discuss in Tongji Apple Club and record the contents in README file. Having this well-organized documents, I am full of satisfaction. What's more, I need to set the deadline to every points in each assignment to ensure we can finish on time. To be honest, due to procrastination, it is pretty difficult.
Next come to the course content, as a organizer, I should understand all the knowledge, to combine them together and check whether there has a mistake. And I also make some contributions in the specific scenario, such as class diagram, iOS UI, design an analysis mechanism and so on, which make me have a deep understanding of a project.
As I filled in the questionnaire, Prof. LIU and Prof. CAO both have a fluent polish and taught clearly, so we can always catch the key points accurately and English teaching did not affect us at all, while it lay a solid foundation for our future study abroad. Thanks to two professors and TAs.

Zhongjin LUO

In this class, I experienced a whole process of System Analysis and Design. I not only learn to use unified modeling language, but also learn to analyze and solve problems using analytical skills. In the first stage, I learn that the first step to design is to analyze use case. And after practicing, I have learned to draw use case diagram and describe the whole use procedure normatively. In the second stage, I have learn the analysis model including class diagram, sequence diagram, communication diagram. I found that the sequence diagram and communication sequence diagram are little different. And converting text to graphs are more clear. In the third step, I have learned to encapsulate different classes to a subsystem and provide interfaces to other subsystem. I found that dividing the whole system to different subsystem make the whole architecture more clear, and of course, it is more easy to maintain.
In class, I found that UML have a lot of advantages. It provides standard for software development, reduces costs to develop diagrams of UML using supporting tools, reduces development time and has large visual elements to construct and easy to follow. Finally, I have improved my English listening and writing ability.

Guohui YANG

The course OOAD with UML was a critical component for my overall understanding of software development process. And it also provided us with direction, insight, and examples then to let us learn by doing, even though it was sometimes challenging.
Through this semester, I have comprehended the fundamentals of the object-oriented paradigm, understood the use of UML notation during analysis and design, and mastered the use of UML tools as well. Furthermore, I had a chance to apply what I learned to our team project which ran through a whole semester. This course allow me to have my first glimpse of the practical large software development process, about which I was curious before. How to abstract the requirements in reality as a understandable model? How to clarify the process of development? What should we do in process of analysis, design and even implementation? This course gave me some kinds of answers.
It is worth mentioning that team work also played an important part in this course, since the course project can not be done on someone's own. Division of work is necessary but we also need to have some standards unified. That requires us to have fully discussion, especially in some hard-to-decide cases, as to us OOAD was totally new. I think we all had gained much.
In conclusion, I am very grateful for the opportunity to have taken this course. I found it enlightening, informative and beneficial towards my goals of being the a good software developer. Also, my success in this course has provided me with the knowledge needed to my further learning. Thank you professors and TAs for your help and patience.

Yirui WANG

Through class and assignment, I learned to perform a requirement analysis of a project and through use case, class diagram, sequence diagram, communication diagram to perform specific analysis and describe the solution. I learned to consider the problem systematically and comprehensively as well as analysis a particular aspect meticulously.
Compared to UML, source code focuses only on the software itself and ignores the rest of the system. Even though the code is a complete and generally unambiguous definition of what the software will do, the source code alone simply cannot tell how the software is to be used and by whom, nor how it is to be deployed; the bigger picture is missing entirely if all you have is the source code. When it comes to informal languages, the informal languages do not have a formally defined notation; there are no hard and fast rules as to what a particular notation can mean, although sometimes there are guidelines. The best thing about having modeled the system using UML is that the notation has a specific and defined meaning. If others know UML, the design would be clearly understood.

Yiqun LIN

I have learnt the process of system analysis and design, how to analyze and how to design our logistic system. We complete the analysis and design of this project step by step, and

I have learnt how to design a use case diagram, communication diagram, activity diagram and sequence diagram. Besides, I have learnt how to work in a team. In this teamwork, I was responsible for design diagram for scenario five and scenario six, and the architecture analysis diagram.
I think our team leader Yang LI is the core of our team. We managed the files on GitHub and He organized the discussion and assigned tasks after each assignment arranged.
Because the course was taught in English and I haven't learnt UML before, I can't follow the teacher without any preview. Each diagram design, I would review the slides and books(Chinese books) to make sure I really understand the point and didn't make any mistakes.
All in all, this course benefits me a lot.

Xinying WU

In system analyze and design course we did the project called onion express as a group. The whole project is divided into three parts, which allows us to complete the project step by step and better understand the knowledge.
Teamwork is a very important part of the whole project. After analyzing the requirements we had a clear division of labor. The first part I was responsible for UI design, the second part I did communication diagram and sequence diagram about return of goods and finally part I designed the subsystem with partners.
It was difficult for me to follow with the teacher because the course is taught in English and I was unfamiliar with UML. Thanks to my team partners explaining the points I do not understand and I also reviewed the textbook to make sure I didn't make mistakes.
It is interesting to have two teachers to give us classes, which gives us a different class experience. And what I learn in the course benefits me a lot.

7 Contributions

More information, please visit us on GitHub

1452559 Yang LI	Project Achievements, Course Reflection	17%
1453645 Zhongjin LUO	Open Issues, Course Reflection	17%
1451229 Guohui YANG	Potential Improvements, Course Reflection	17%
1552705 Yiqun LIN	Limitations of Design, Course Reflection	17%
1552651 Yirui WANG	Development Plan, Course Reflection	17%
1552677 Xinying WU	Development Plan, Course Reflection	17%