SLOTH EXPRESS

PACKAGE DELIVERY SYSTEM ANALYSIS DOCUMENT

1452764 何冬怡 1452798 李 想 1452697 彭嘉琦 1593369 Luc Berro

CONTENT

1 Introduction

The project aims at building a system for logistic companies (assuming that logistic companies doesn't receive individual orders) which helps them in logistics tracking, goods packing, goods distribution, after-sales management, data storage, information processing, etc.

For user interface, we would provide users with Web/Android/IOS apps (which will be displayed in architectural analysis) at last.

- **System Analysis**

We have designed the whole structure of our system which is built on MVP pattern. Compared with MVC pattern, MVP makes it possible to separate Model and View totally. Then it will be easier to modify out View layer in the following process. Moreover, testing the logic can be done without user interface – the logic would be put in the Presenter layer.

In the high-level architecture, different layers and packages have been specified and subsystems have been classified. Besides, we've considered APIs such as IOS and Android API.

Class diagrams are done, which means we have considered the static structure of the model, detailed interfaces of classes and types of variables are decided. While how these objects interact with each other are described in the sequence and communication diagrams.

We've thought about how the system works from bottom layer such as how network interact with server, these are displayed in our sequence and communication diagrams.

The UI (will be displayed later) have been updated, we add more details and some UI descriptions to explain the key functions.

- **Updates**

Added database: Even if we've considered data storage before, we add it to our architecture this time really and begin to take it as a part of our system seriously. Because database is really an important part – it's directly related to the efficiency of our system.

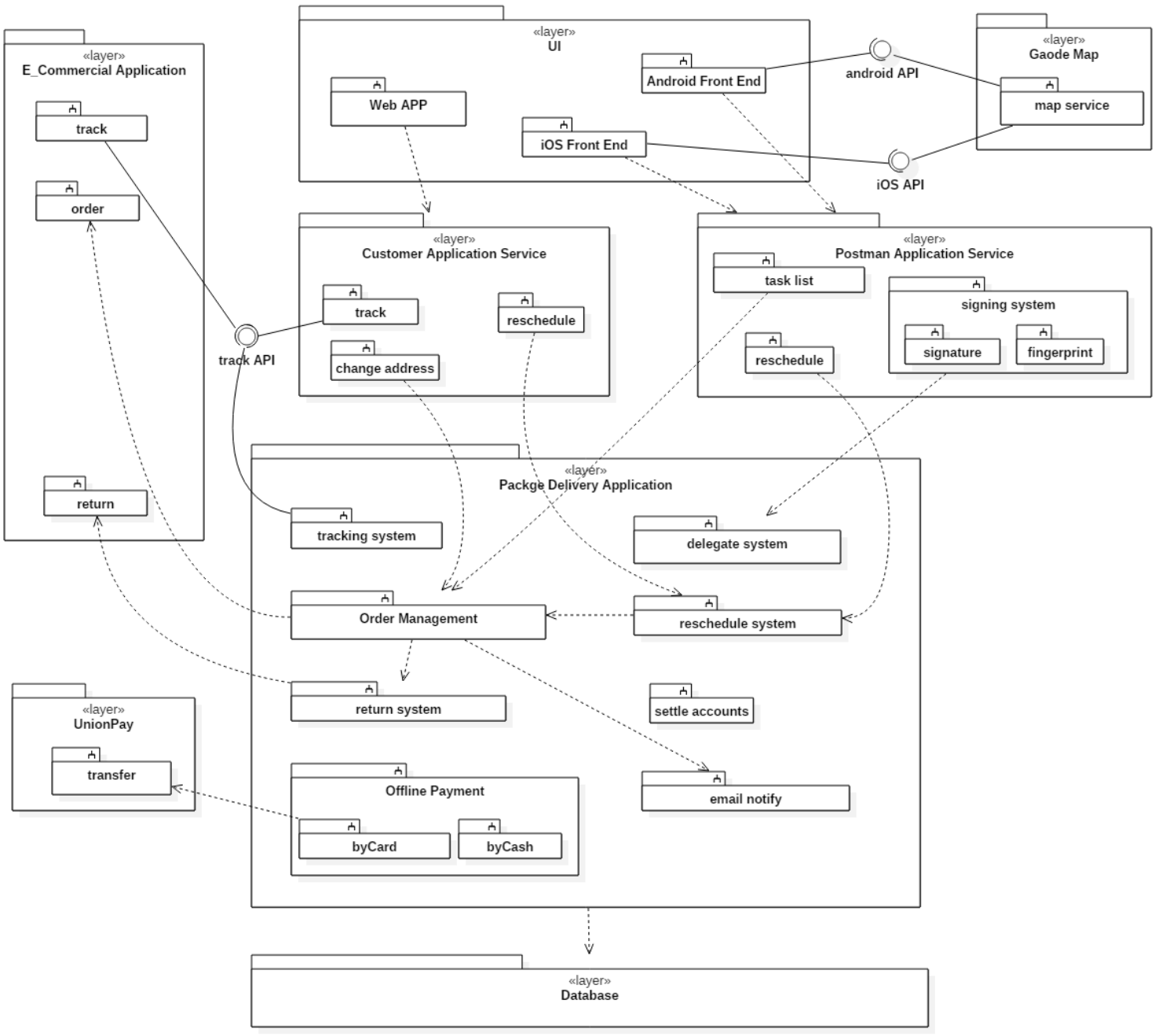
Specified the detail of settling accounts with e-commercial company: we would finish this task through bank. It's mainly out of security and convenience.

Specified the detail of map: We choose the APIs of Gaode Map to supply map service for postman. Between the two mainstream maps – Baidu and Gaode, the APIs of Gaode is friendlier for Android.

Considered more about APIs: In the class diagram, we added detailed request information on how to use the tracking API.

Updated UIs: We updated the UI for customer – adding reschedule and changing address, which helps customer manage his/her order more conveniently.

2 Architectural Analysis



2.1 Outline

The entire system is designed following MVP architectural pattern, composed of 5 layers and depends on 3 external packages. From top to bottom, the layers are considered to be **User Interface**, **Customer/Postman Application Service**, **Package Delivery Application** and **Database**. External packages are **E-Commercial Application**, **UnionPay** and **Gaode Map**.

2.2 **UI layer (View)**

In the UI layer of the system, there are **Web APP** (for customer), **Android Front-End** and **iOS Front-end** (for postman) working as view in MVP pattern. The Android/iOS front-end invoke corresponding API provided by Gaode Map company for map services, including GPS positioning and navigation.
In particular, E-Commercial Application is partly the user interface of the system but not included in UI layer, for it is external.

2.3 **Customer/Postman Application Service (Presenter)**

The two packages are considered to be on the same level, which work as the presenter in MVP pattern. We encapsulate related services into these two packages to supply services for applications in different front-ended uniformly. Track, reschedule and change address are accessible in **Customer Application Service**; and **Postman Application Service** includes subsystems of task list, reschedule and signing.

2.4 **Package Delivery Application (Model)**

The Package Delivery Application acts as the model in MVP pattern, which deals with all business logic. Inside the layer there are **tracking**, **delegate**, **order management**, **reschedule**, **return**, **settle accounts**, **payment** and **email notify** system.

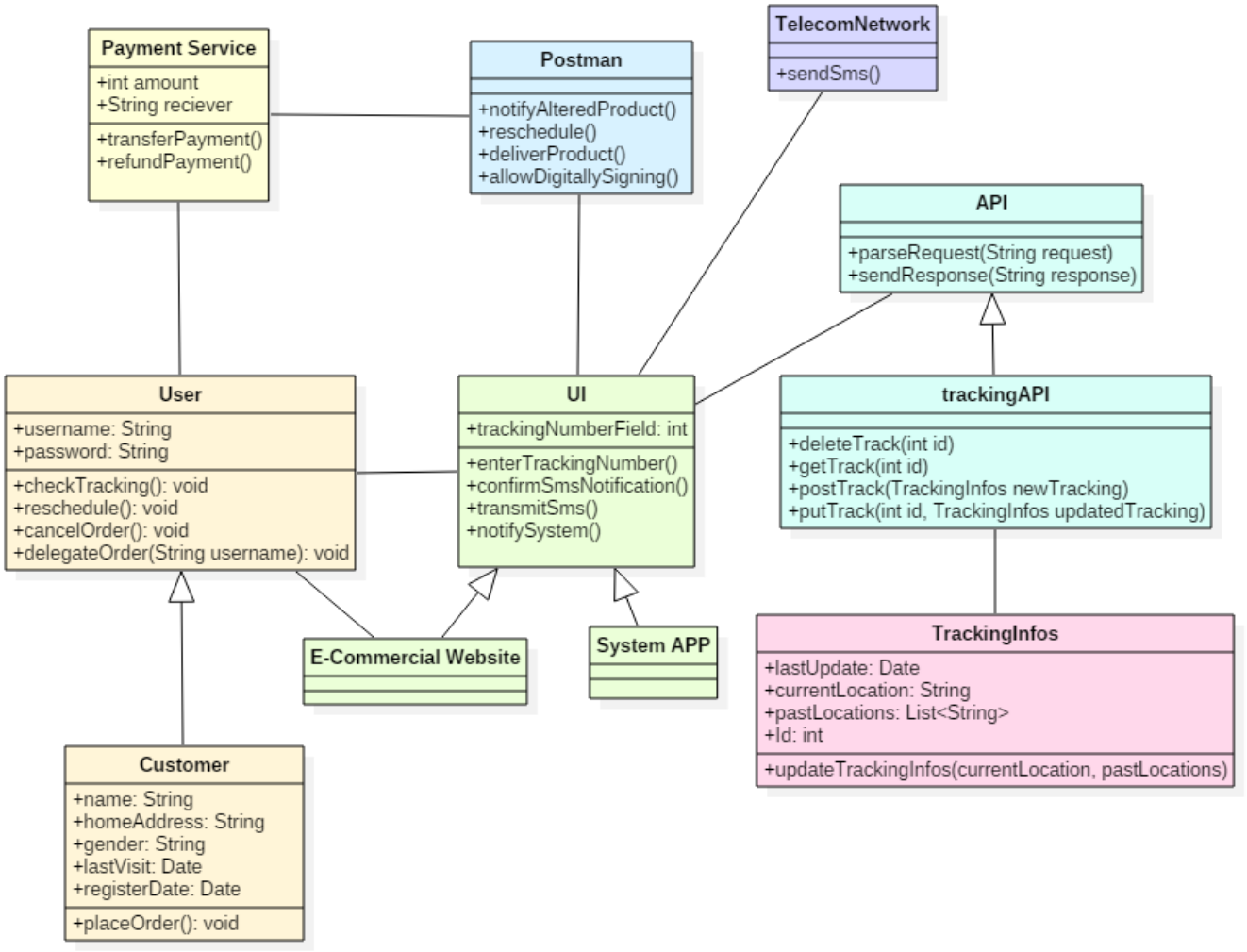
As for external dependencies, the tracking system provides a track interface for customer and e-commercial application; and offline payment system invokes API from UnionPay for transfer services.

2.5 **Database layer**

This layer is at the bottom of system architecture. It stores all the business data of the company including staff information, order information, account statement, nodes of dispatch center, etc.

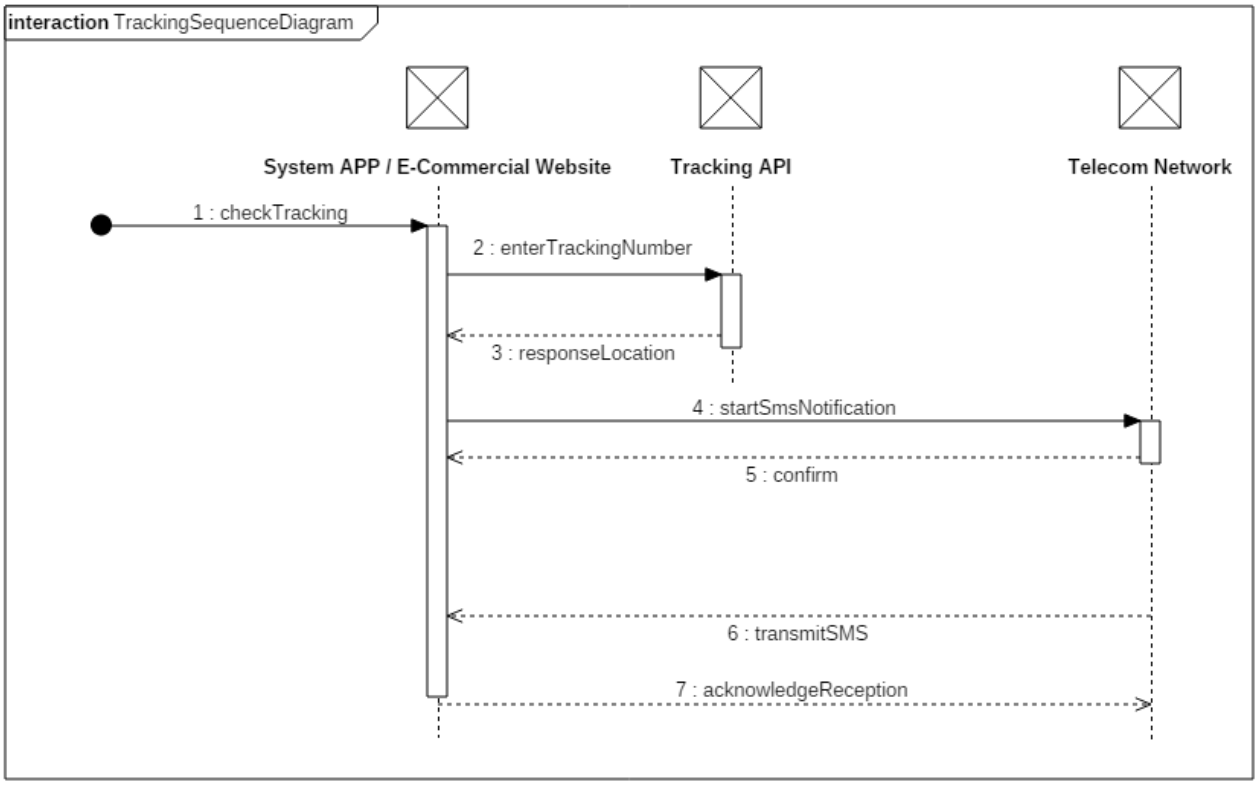
3 Analysis Model

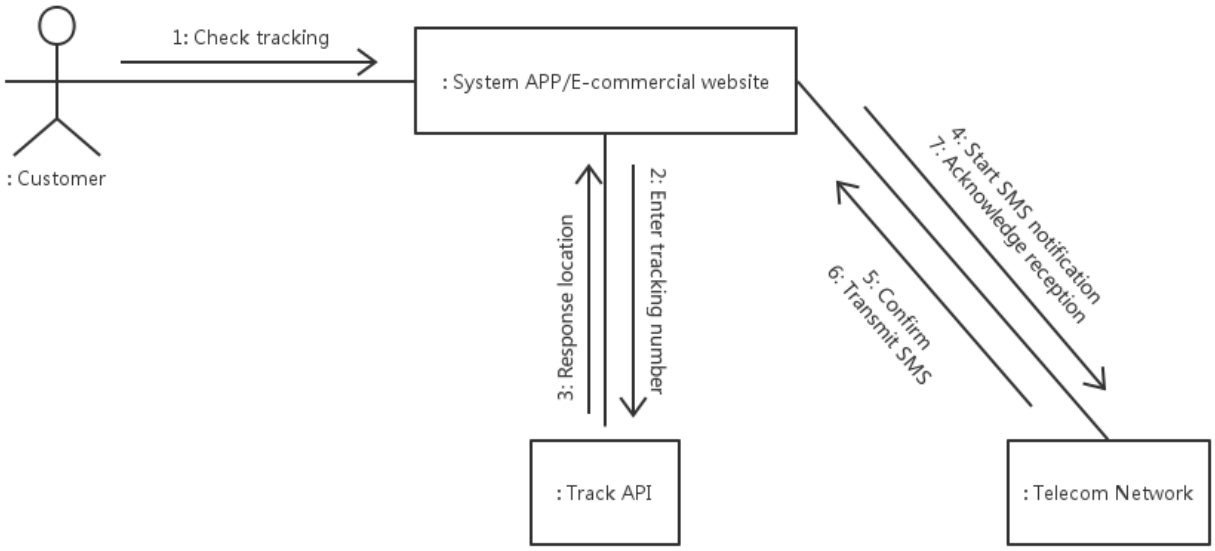
3.1 Class Diagram



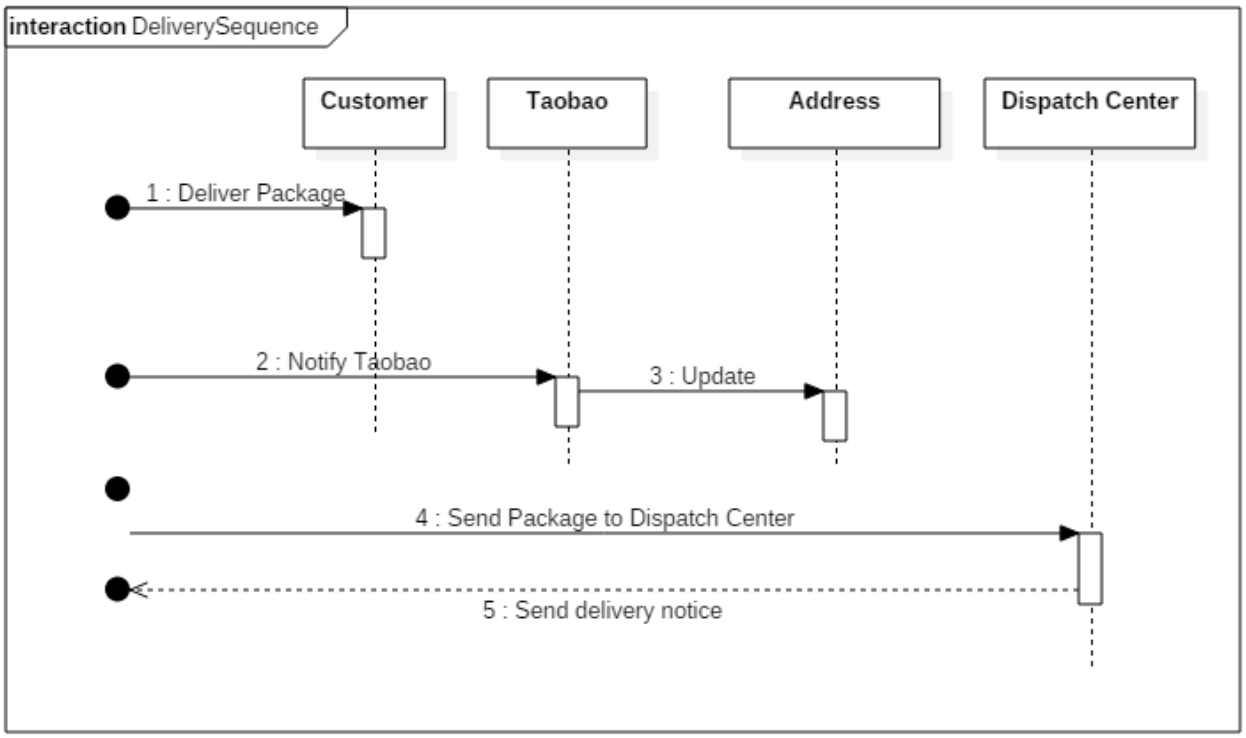
Class Diagram 1

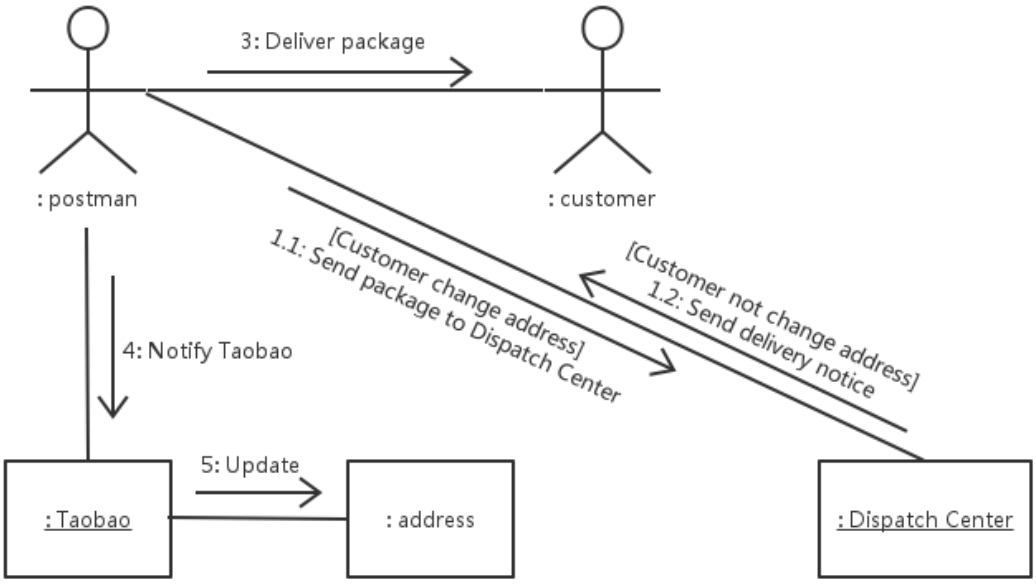
3.2 **Tracking System**



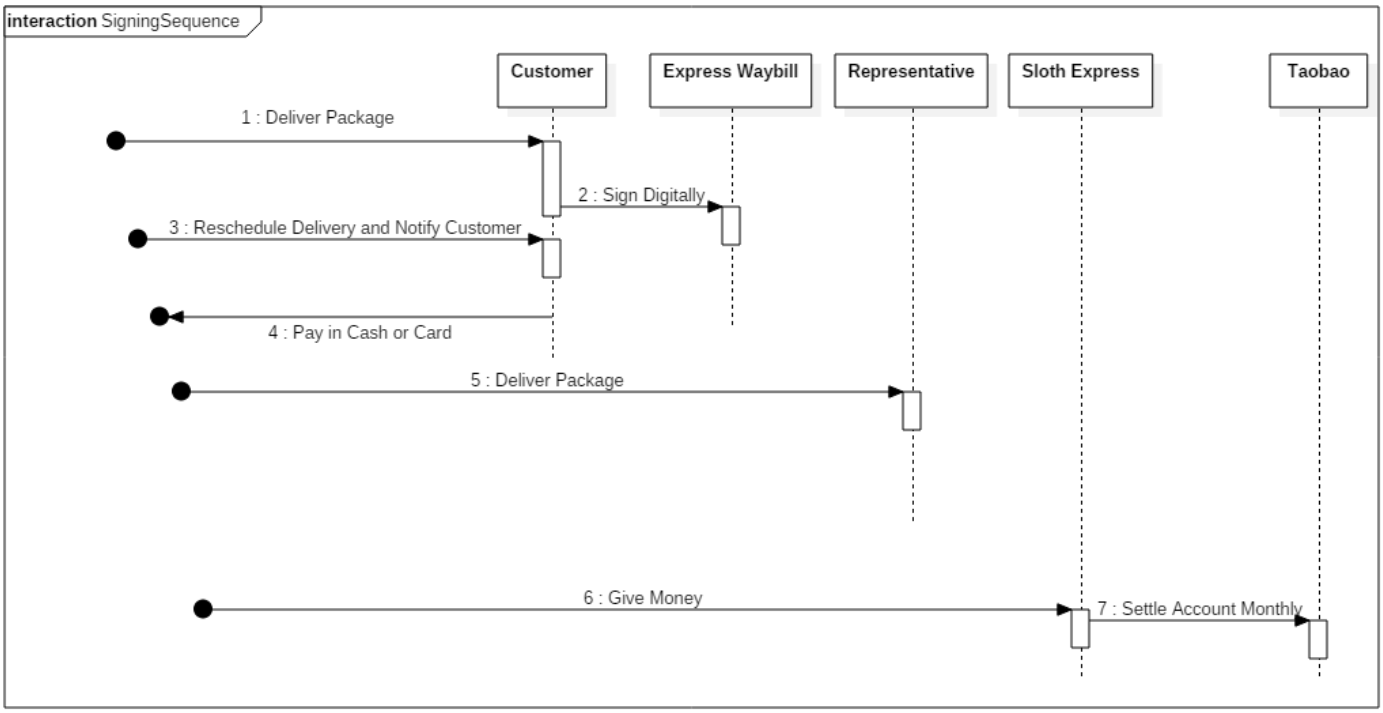


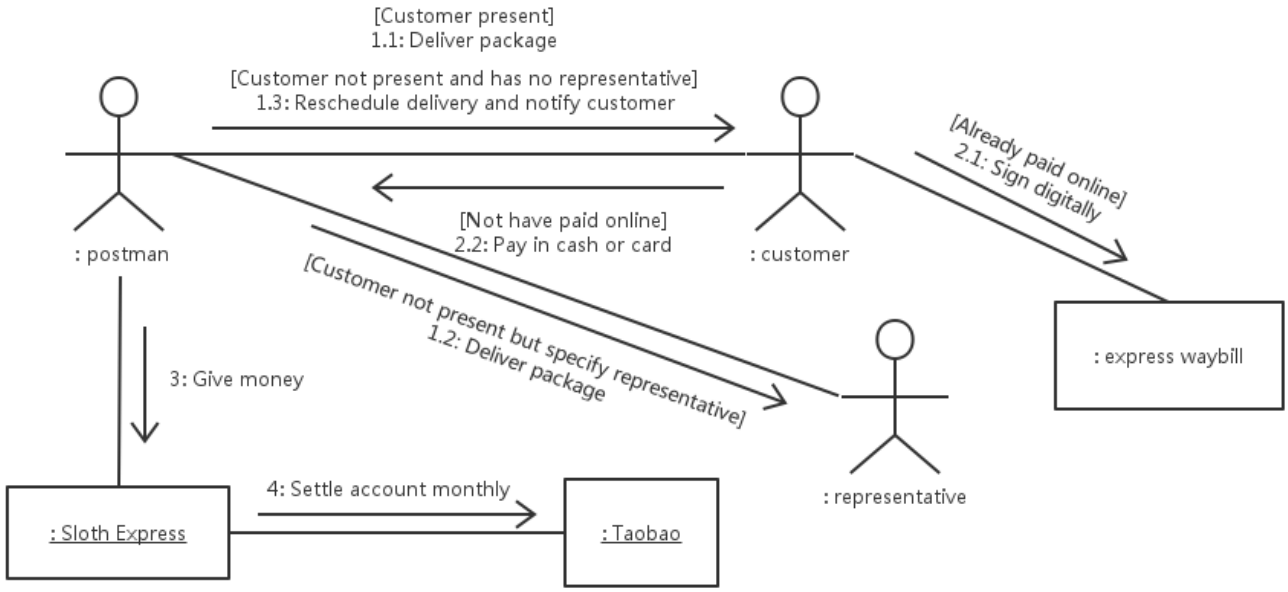
3.3 **Delivery System**



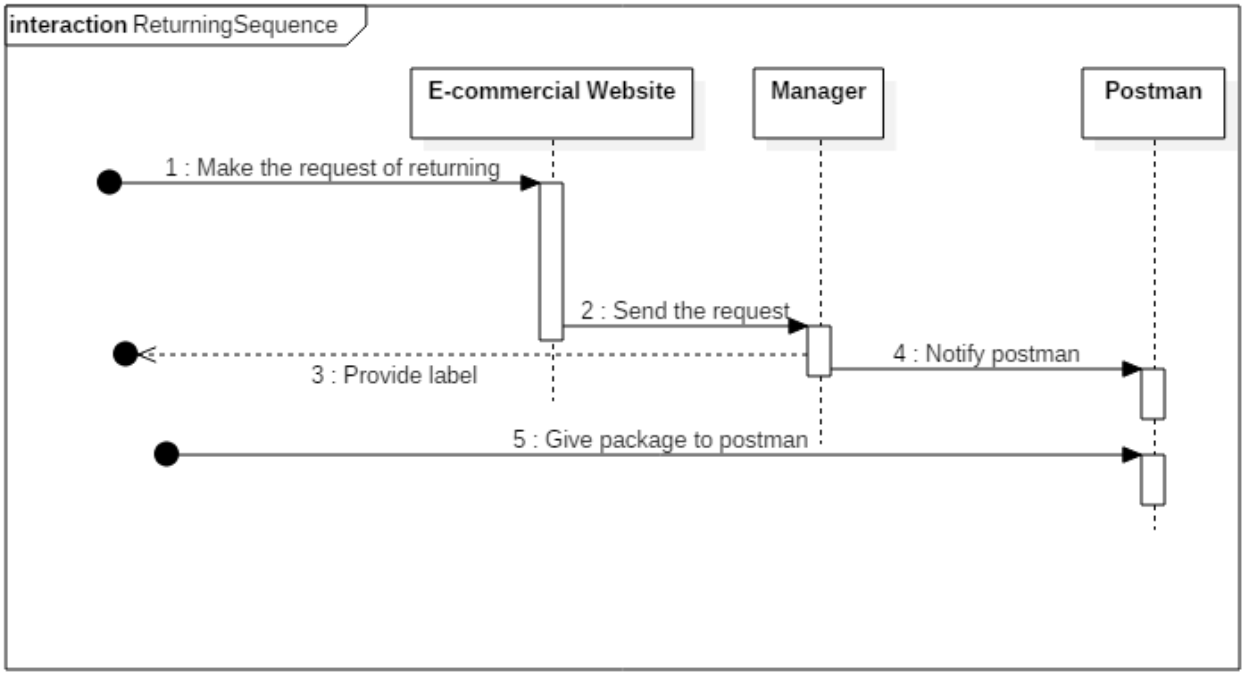


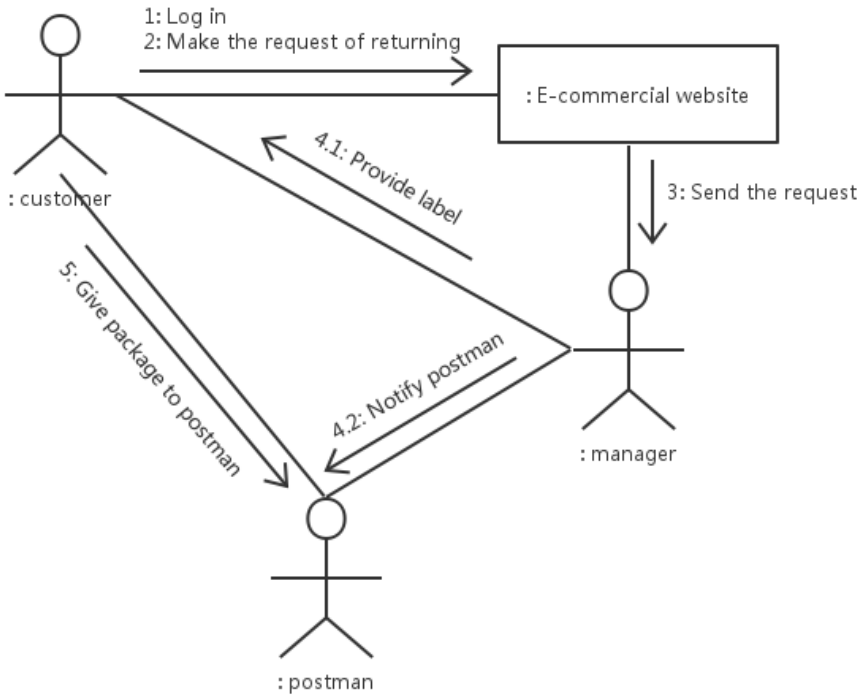
3.4 Signing System



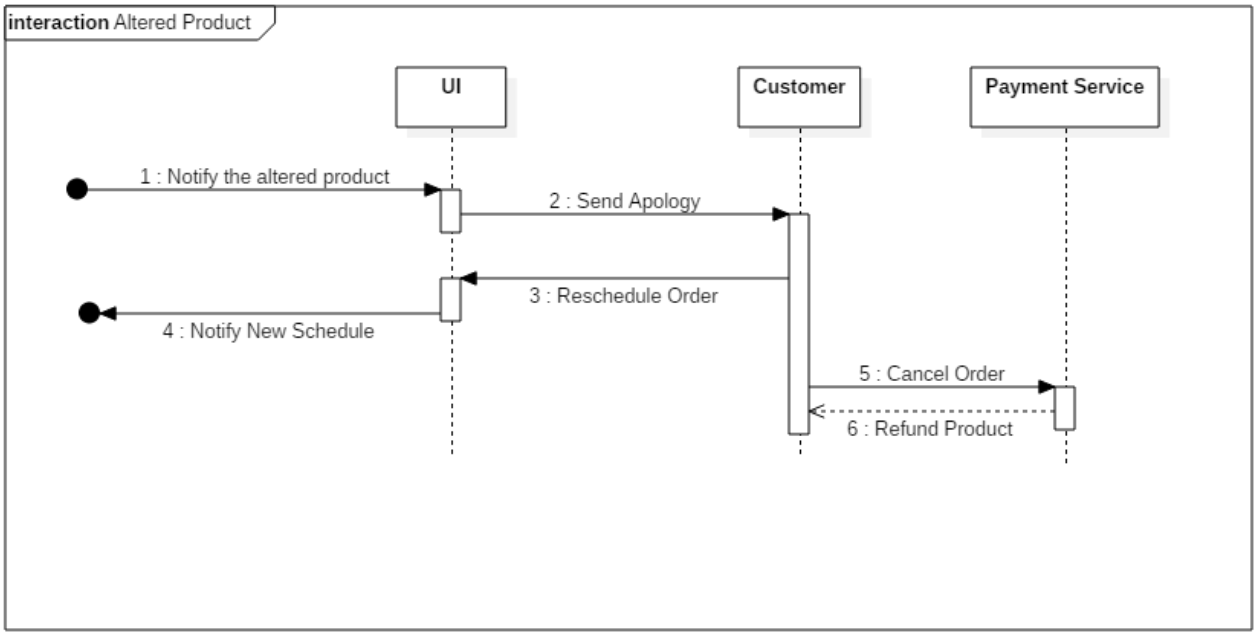


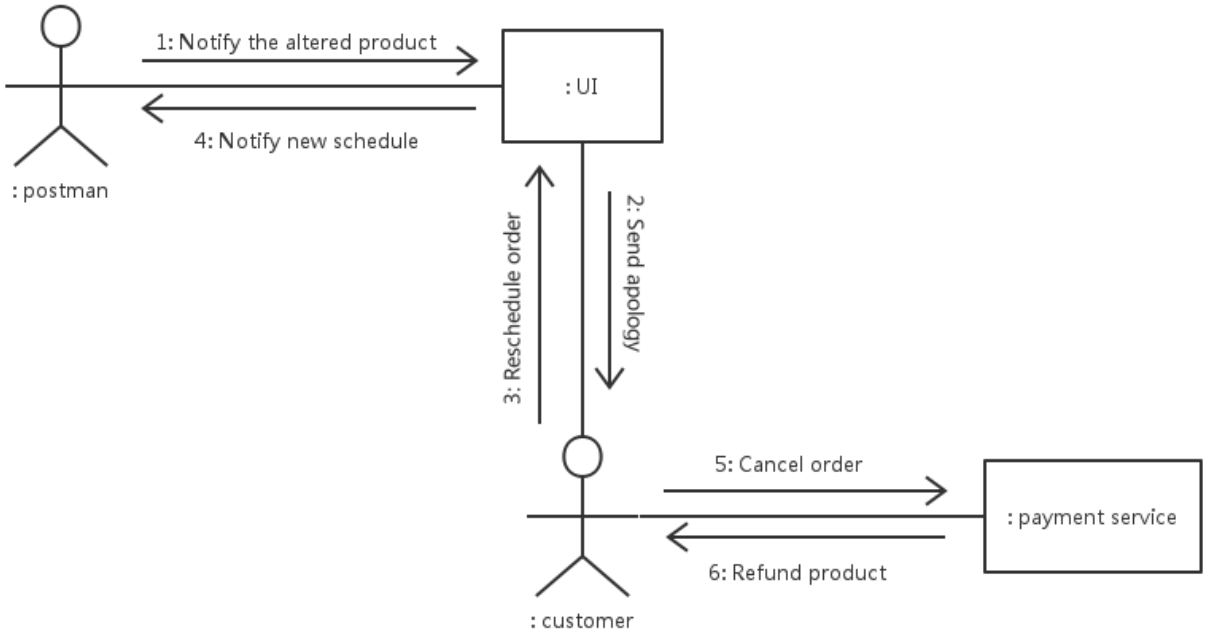
3.5 **Returning System**





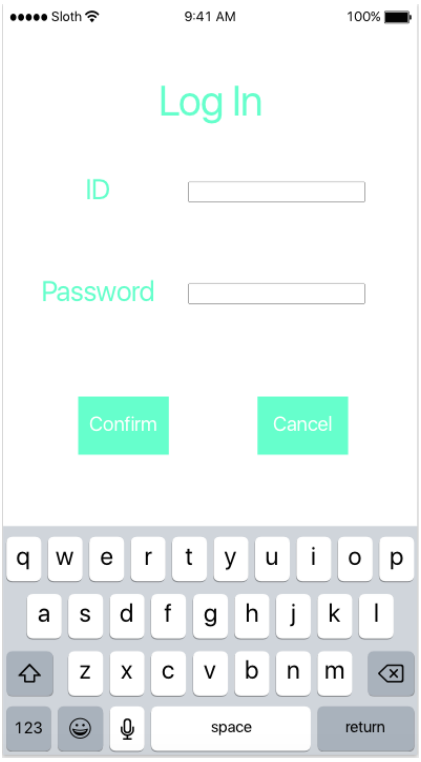
3.6 **Altered Product**



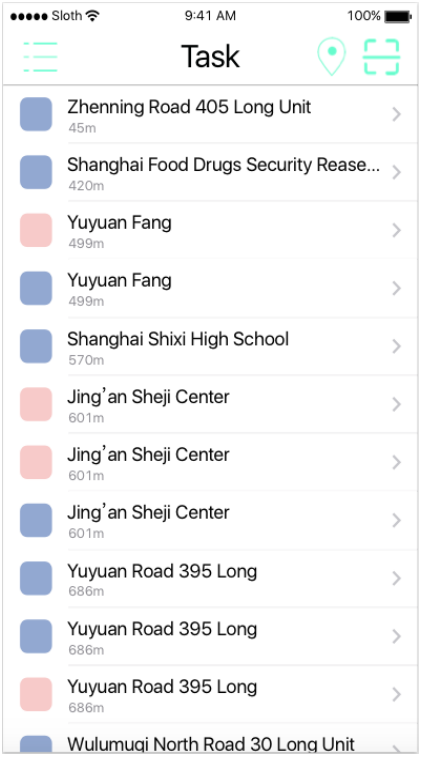


4 Updated User interface

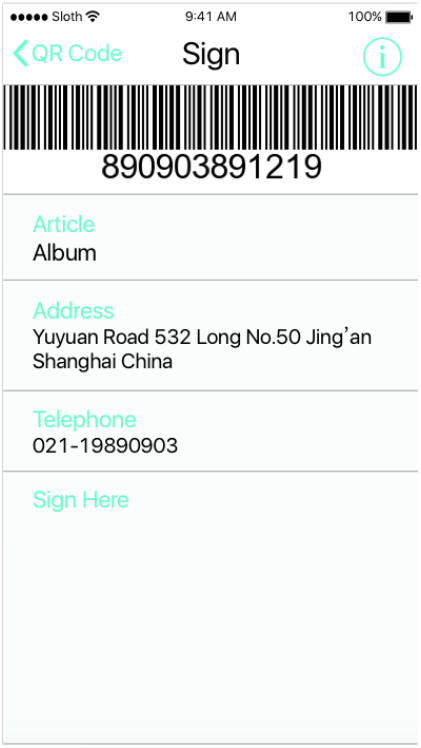
4.1 iOS APP



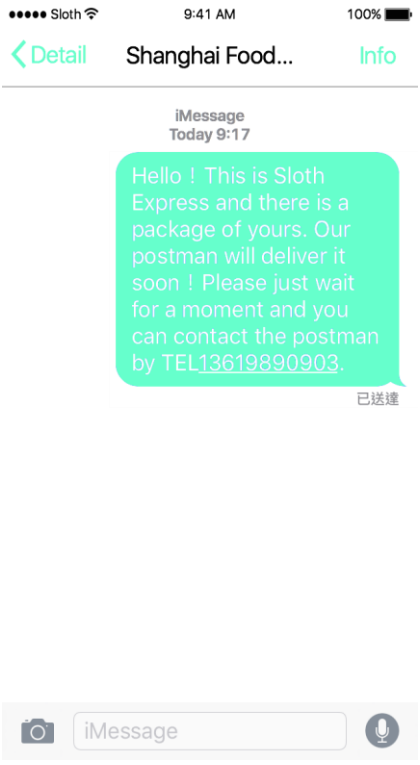
Log in: It will be shown when the postman uses our App on the first time. Just input his ID and password offered by his company, and he can use all functions in this App.



Task: This page is just like a list. It shows all the tasks the postman should finish today. The delivery task is marked in Rose Quartz, as well as the sending task is marked in Serenity. The bigger word is a simple address and the smaller word is the distance between the postman and the destination. On the upper-right corner, there is a map icon and a QR code icon. The user can touch them to use the appropriate function.

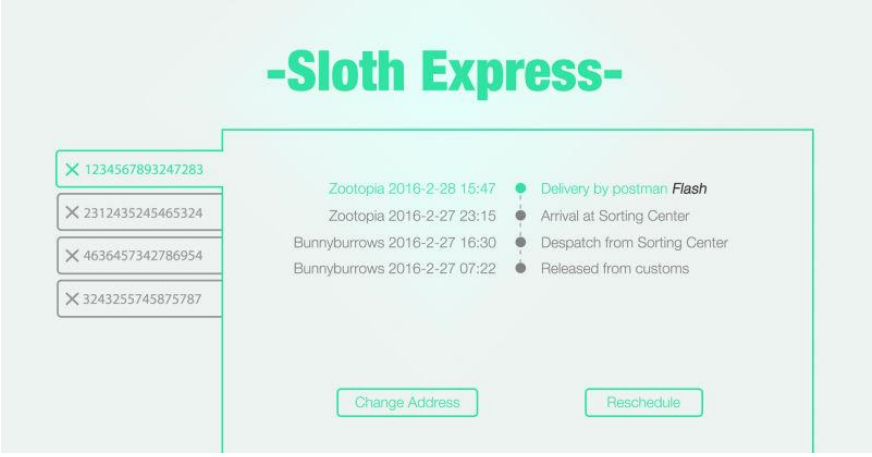


Sign: The package's barcode is on the top of the page. Under it is the product name, the detailed address and the customer's telephone number. The sign region is on the bottom of the page. The customer can use a magnetic pen to sign his name on the Pad.



Message: This page shows a message, which is sent by the postman. It tells the customer that he has a package, and offers the postman's phone number for customer to contact with.

4.2 **Web APP**

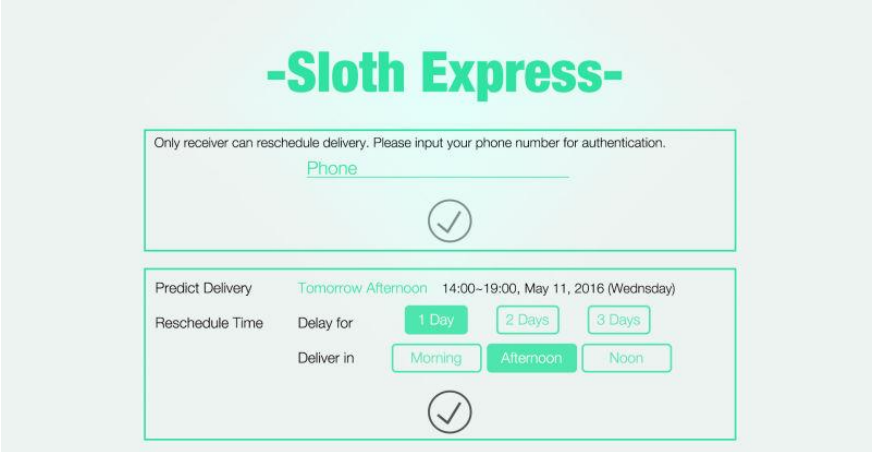


Detailed information

This page shows the detailed information of packages, including where the package has arrived, when the package arrived there, and who has taken the package. The two buttons link the next two pages.

Change address

First, the user should input his phone number to check if he has the jurisdiction to change address. When the authentication is passed, the user can input the new address in the bottom box.



Reschedule

Only receiver can reschedule delivery, so this function also needs an authentication step by inputting the phone number. When the user passes the authentication, he can select the delay days and when to deliver.

5 Annotated references

5.1 Book

Addy Osmani. Learning JavaScript Design Patterns [M]. Sebastopol, California: O'Reilly Media, 2012

With *Learning JavaScript Design Patterns*, you'll learn how to write beautiful, structured, and maintainable JavaScript by applying classical and modern design patterns to the language. If you want to keep your code efficient, more manageable, and up-to-date with the latest best practices, this book is for you.

Explore many popular design patterns, including Modules, Observers, Facades, and Mediators. Learn how modern architectural patterns—such as MVC, MVP, and MVVM—are useful from the perspective of a modern web application developer. This book also walks experienced JavaScript developers through modern module formats, how to namespace code effectively, and other essential topics.

- Learn the structure of design patterns and how they are written
- Understand different pattern categories, including creational, structural, and behavioral
- Walk through more than 20 classical and modern design patterns in JavaScript
- Use several options for writing modular code—including the Module pattern, Asyncronous Module Definition (AMD), and Common JS
- Discover design patterns implemented in the jQuery library
- Learn popular design patterns for writing maintainable jQuery plug-ins

5.2 Articles

[1] Li Xuandong, Cui Meng, Pei Yu, Zhao Jianhua, Zheng Guoliang. Timing Analysis of UML Activity Diagrams[A].In: Martin Gogolla, Cris Kobryn. <UML>2001 — The Unified Modeling Language, Modeling Languages, Concepts, and Tools[C]. Toronto: 4th International Conference,2001:62-75.

UML activity diagrams can be used for modeling the dynamic aspects of systems and for constructing executable systems through forward and reverse engineering. They are very suitable for describing the model of program behavior. In this paper, we extend UML activity diagrams by introducing timing constraints so that they can be used to model real-time software systems, and give the solution for timing analysis of UML activity diagrams. We give the solution for timing analysis of simple UML activity diagrams (containing no loop) by linear programming, and present an algorithm for checking UML activity diagrams using integer time verification techniques. This work forms a base for verification of real-time software systems.

[2] Marlon Dumas, Arthur H. M. ter Hofstede. UML Activity Diagrams as a

Workflow Specification Language[A]. In: Martin Gogolla, Cris Kobryn. <UML>2001 — The Unified Modeling Language, Modeling Languages, Concepts, and Tools[C]. Toronto: 4th International Conference,2001:76-90

If UML activity diagrams are to succeed as a standard in the area of organizational process modeling, they need to compare well to alternative languages such as those provided by commercial Workflow Management Systems. This paper examines the expressiveness and the adequacy of activity diagrams for workflow specification, by systematically evaluating their ability to capture a collection of workflow patterns. This analysis provides insights into the relative strengths and weaknesses of activity diagrams. In particular, it is shown that, given an appropriate clarification of their semantics, activity diagrams are able to capture situations arising in practice, which cannot be captured by most commercial Workflow Management Systems. On the other hand, the study shows that activity diagrams fail to capture some useful situations, thereby suggesting directions for improvement.

6 Contributions of team members

1452764 何冬怡(Leader)
Document summary and arrangement;
Web UI updates;
High-level Architecture (Layer diagram);
Textual description of system;

1452798 李 想
Description of UI updates;
iOS UI updates;
Annotated references;
Communication diagrams;

1452697 彭嘉琦
Main introduction of document;
High-level Architecture (Layer diagram);

1593369 Luc Berro
Class diagrams;
Sequence diagrams;
Presentation;

7 Enclosure

7.1 Modelling file of diagrams

With URLs below to view our **communication diagram**s:

Altered Product	http://www.processon.com/view/link/5734b3cae4b0d4e09767820b
Delivering Product	http://www.processon.com/view/link/5734b45fe4b0a43bbdd89e6b
Returning Product	http://www.processon.com/view/link/5734b495e4b0a43bbdd89f11
Signing Product	http://www.processon.com/view/link/5734b4b6e4b0d4e097678505
Tracking Product	http://www.processon.com/view/link/5734b4d9e4b0c102ad2c515c

High-level Architecture (Layer Diagram) /architecture.mdj
Anaysis Models (Class diagrams & Sequence diagrams) /model.mdj

7.2 UI mockups images

Web UI /WebUI **iOS UI** /iOSUI