Conference Management System

University of Illinois at Urbana Champaign

CS428 : Software Engineering II

Prof. Darko Marinov

Spring 2014

Team **CMS**

Alexander Hadiwijaya (hadiwij2)

Derek Quach (dlquach2)

Eunsoo Roh (roh7)

Han Jiang (hjiang25)

Jiangchuan Zhou (jzhou31)

Kirill Mangutov (manguto2)

Table of Contents

1. Introduction

1.1 Conference Management System Overview

The goal of the project was to create a "Conference Management System." A conference manage system allows people to register for the conference, set preferences (food, topics interested, sessions to attend, hotel room, etc), and submit abstracts of their publications. The system also allows conference administrators to contact the attendees via email, where email address and recipient's title can be pre-populated (only body needs to be filled out). The system has a user management subsystem that can manage the users by assigning them to different privilege levels, e.g. normal users can only view certain information, while some privileged users can view more information and modify it.

The system is expected to be used for the management of the annual Railroad Environmental Conference (RREC, about 500 attendees each year) and potentially some other conferences. Thus, the system is supposed to be very useful. Currently, the management of the RREC is not efficient. Since it doesn't have a integrated system for information processing, at least one full-time staff and 3 student workers are working on dealing with the information management, phone calls, and emails. If we had created this system, the efficiency for the conference management would've been improved significantly.

1.2 The Team

Our team is a group of undergraduate students majoring in Computer Science and Computer Engineering. The team members are as follows:

- Alexander Hadiwijaya (hadiwij2)
- Jiangchuan Zhou (jzhou31)
- Derek Quach (dlquach2)
- Eunsoo Roh (roh7)
- Kirill Mangutov (manguto2)
- Han Jiang (hjiang25)

1.3 Explanation Of Terms

- PHP: a server-side scripting language designed for web development but also used as a general-purpose programming language, according to definition on Wikipedia.
- PHPUnit: a testing framework for PHP, allowing PHP code to be tested automatically by writing test definitions
- Mockery: a mocking framework for PHP, allowing to decouple dependent modules and components within automated tests.
- Symfony: a PHP web application framework for MVC applications. Symfony is free software and released under the MIT license, according to definition on Wikipedia.
- Apache HTTP Server: commonly referred to as Apache, is a web server application notable for playing a key role in the initial growth of the World Wide Web, according to Wikipedia.
- Extreme Programing: also known as XP, is a software process intended to improve software quality and responsiveness to changing customer requirements, according to Wikipedia.
- Capture: request to the payment server for crediting funds after successful authorization.

1.4 Note to the Readers

The concepts in the document assume familiarity with the following programming languages, techniques, technologies, standards, libraries and frameworks:

PHP, Symfony, UML, Web server, basic knowledge about software development, etc.

2. Software Process

We used a slightly modified version of the development process we learned in previous class (CS 427), which is called Extreme Programming, or XP. One reason we chose XP was because it has certain characteristics that help improving software quality and meeting customer demands. As an iterative agile software development process, its fixed-size iterations and frequent customer interactions prevent large deviations, which could be costly. Also, continuous integration along with refactoring allows improving code quality with minimal regression. Another reason was that we were all familiar with XP, since everyone in the team has taken CS427 before and finished project with using XP. To us, XP is the process proved to be practical, not to mention it has led to numerous successful projects. Also, following XP saved us a lot of time as it doesn't involve too much documentation work. With heavy coursework in parallel with the project, saving the time for separate documentation to make the code more readable was a tremendous advantage of using XP.

2.1 Iterative Development

We picked a few user stories every iteration then tried to implement them within the iteration. Sometimes it went according to plan, but sometimes it was delayed. There were various reasons for the delay, such as coursework overload e.g. exams and homeworks, too optimistic time estimation and third party integration bottleneck e.g. waiting for e-mail from U of I Merchant Card Services. Iterative development still proved to be useful, as we could properly re-evaluate the risks and time estimations to have a better idea what has to be/could be done. Without iterative development, we wouldn't have had a clear idea on what would be the set of features that we could deliver in final iteration and possibly ended up with half-finished user stories.

2.2 Refactoring

Refactoring was generally performed each iteration as clunky and repeated code was rewritten to increase maintainability and readability. As each pair worked separately, we found that pairs often create code that is often functionally identical with code written by another pair working on a different part of the project. At the group meeting after each iteration, these replications are often found and refactored out into a more universal function/module. A good example of this would be extracting authentication and data loading logic within individual test classes to a unified test fixture base class. By refactoring those private functions into a new class that every bundle would use instead, we were able to dramatically clean up the test fixtures. Other common forms of refactoring included coding standards enforcement and shifting controller logic away from the view and vice versa.

2.3 Testing

We performed unit testing, system testing, and graphical interface testing to the CMS system. The tests were written in PHPUnit framework, as well as Mockery and Symfony DOM Crawler for mock-based testing and user interface testing. Following the guidelines of XP, we continuously added new

unit tests during the development process. We also set up Jenkins CI to automatically run the tests on github push, then deploy to the development server if test succeeds. Whenever there was a failed unit test, our developers would receive an Email from Jenkins, which enables the developers to deal with the failed test promptly. Having the CI and unit/functional tests set up helped us perform the refactoring process without introducing many new bugs (regression).

After the implementation of the system, we also manually tested the system according to its specifications. This proved to be quite useful, as unit/integration tests alone cannot solve all problems in a software. For example, our manual interface testing helped us fix some problems of our user interface design, making the system's user interface less crude and more user-friendly.

2.4 Collaborative Development

We laxed the pair programming requirement from traditional XP process, where two programmers develop together on one workstation. Because each member of the group has quite different schedule, it was difficult for pairs to meet up outside of the group meetings. Therefore, while user stories were still assigned to a pair, members of the pair were allowed to work separately. We also had approx. 4-hour group meeting every weekend, so we had enough opportunities to share ideas and make project decisions together.

3. Requirements and Specifications

3.1 User Stories:

- As a user, I want to register as an attendee
- As a user, I want to view and edit my information such as email, password, etc
- As a user, I want to be able to sign up to attend a conference from a list of available conferences
- As an administrative user, I should be able to send mass emails to everyone signed up to attend a specific conference because I need to send updates about the conference
- As an administrative user, I should be able to create a new conference such that attendees can select to attend this conference from a list
- As a user, I want to be able to reset my password using my email should I forget my password
- As a user, I want to use this system to make a payment to cover the attendance fee of the conference
- As an administrator, I want to be able to modify administrative status of other users and delete them.

3.2 Use cases:

- Administrator Create a conference instance
- Administrator Send mass email to all attendees of a conference
- Administrator View list of attendees and modify administrative status
- Attendee Register an account
- Attendee Sign up to attend a conference
- Attendee Edit information such as email, name, and password
- Attendee Pay the cover fee
- Attendee Reset password using email
- Presenter Add abstract when enrolling in conference

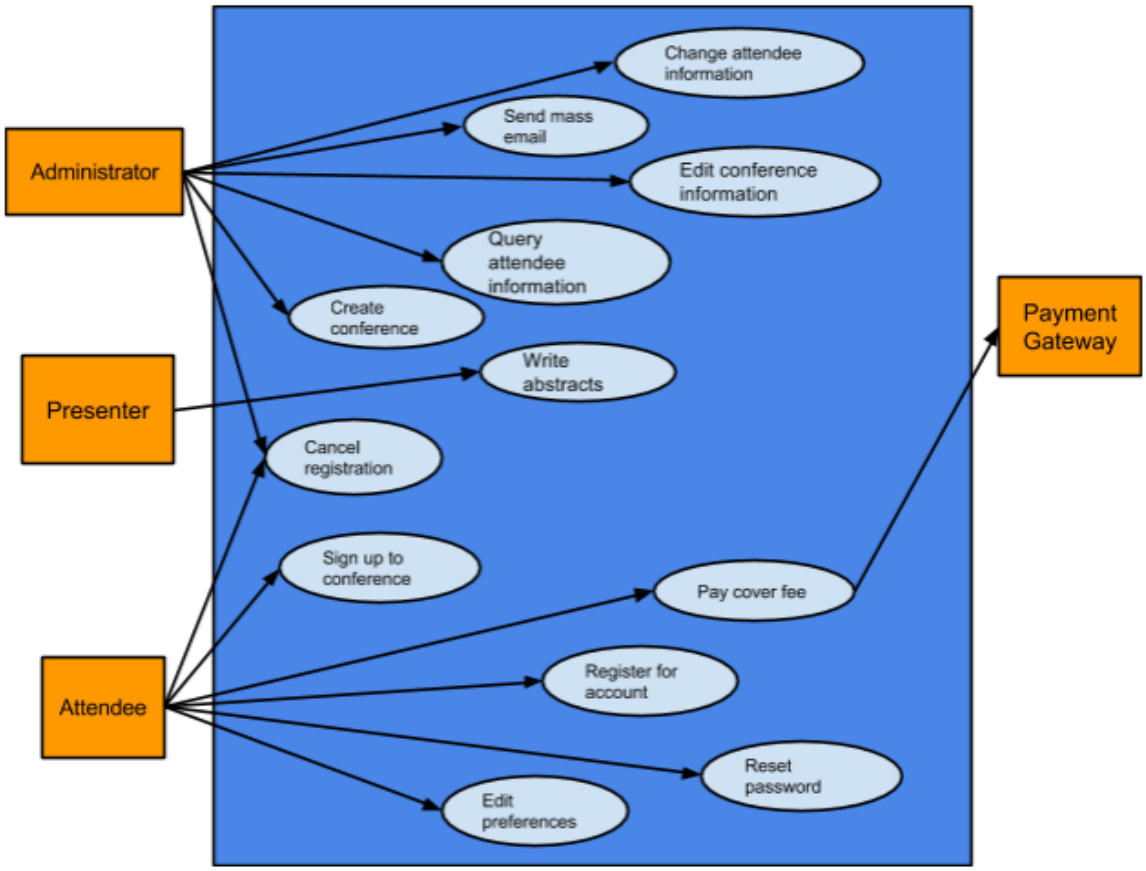

Figure 3.1 Use case diagram

4. Architecture and Design

4.0 Languages and Frameworks

- *PHP* 5.5+
- *jQuery*
- *Bootstrap* 3
- *Symfony* 2.4 with *Twig* and *Doctrine*
- *MySQL* and *SQLite* for database backend
- `FOSUserBundle` for user subsystem
- `JMSPaymentCoreBundle` for payment integration
- *PHPUnit* and *Mockery* for testing

4.1 System Architecture

Our system is built on top of *Symfony* 2, which implements Model-View-Controller pattern for web services. We also used *Doctrine* ORM to persist/access model objects from/to database using Active Record pattern.

The overall system is written as seven *Symfony* bundles (packages) as well as master configuration/views residing in "app" directory. The breakdown is as follows:

"app" directory

- **app/config/config.yml**: master configuration file; sets up parameters for frameworks and bundles
- **app/config/parameters.yml**: per-site configuration file; contains parameters that must be configured per site e.g. database credentials. automatically generated from "app/config/parameters.dist.yml".
- **app/config/routing.yml**: defines routing for the website; hooks up URL to controllers
- **app/Resources**: contains css/javascript and master twig template files. Master templates are extended and overridden in each bundle.

`UiucCmsSiteBundle`

Contains controller for the index page and base twig templates for other bundles.

- `DefaultController`: minimal controller serving the index page

`UiucCmsUserBundle`

Provides home page for logged in users, as well as user account subsystem by integrating/overriding `FOSUserBundle` features.

- `DefaultController`: minimal controller serving the user home page (/user)
- `User`: model object for a user. Two additional fields, `User::firstName` and `User::lastName`, are added atop of `FOSUserBundle`-provided fields: `User::username`, `User::email`, `User::password`, etc.
  Privileges are granted by adding a role to `User::roles` field.

`UiucCmsConferenceBundle`

Controls the logic for creating, viewing, and enrolling in conferences. Also handles management of conferences.

- `ConferenceController`: controller for all conference related sections. Handles all actions under the "/conf/" tree.
  - `listAction()`: Lists all conferences and the user's status with them (enrolled, closed, open).
  - `createAction()`: Generate a form with the needed fields to create a conference.
  - `submitAction(Request $request)`: Validates the conference creation form and informs the user of any infractions. Persists the conference to the database.
  - `displayAction(Conference $conference)`: Displays a particular conference and all of its fields. Checks whether or not a user is enrolled in that particular conference and if enrollment is possible.
  - `payFeesAction(Conference $conference)`: Prepares a Payment object to be handed off to the Payment bundle to handle.
  - `enrollAction(Request $request, Conference $conference)`: Validates a particular enrollment for a conference and persists it to the database.
  - `listEnrolledAction()`: Queries for all conferences a user is enrolled in.
  - `viewCreatedAction()`: Queries for all conferences an admin has created.
  - `manageAction($id)`: Creates the management page for a specific conference.

    - `enrollInfoAction(Conference $conference)`: Generates a form with the necessary fields to complete and enrollment.
    - `viewEnrolledAbstractAction($confId, $attendId)`: Views the abstract and chosen food option for a conference the user has enrolled in.
- `Conference`: model object for a conference.
    - `Conference::createBy`, the user id of the admin who created this conference.
    - `Conference::name`, the name of the conference.
    - `Conference::year`, the year the conference is set to occur.
    - `Conference::city`, the city the conference is set to take place in.
    - `Conference::registerBeginDate`, the date registration opens.
    - `Conference::registerEndDate`, the date registration closes.
    - `Conference::topics`, the listed topics for the conference.
    - `Conference::maxEnrollment`, the maximum occupancy of the conference.
    - `Conference::coverFee`, the cost of attendance for the conference.
- `Enrollment`:model object for an enrollment which connects a user a conference they are enrolled in.
    - `Conference::conferenceId`, the id of the conference in question.
    - `Conference::attendeeId`, the id of the user enrolled in the conference.
    - `Conference::enrollmentDate`, when the user enrolled in the conference.
    - `Conference::coverFeeStatus`, whether or not the cover has been paid.
    - `Conference::food`, vegetarian or non-vegetarian food selection for the user.
    - `Conference::paperAbstract`, the abstract of the attendee's paper.

`UiucCmsPaymentBundle`

Payment middleware between `JMSPaymentCoreBundle` and `ConferenceBundle`.

- `PaymentController`: controller for payment integration. The "client" bundle must create an `Order`, then forward to `PaymentController::choosePaymentAction()` for initiating payment process with one of the registered providers.
    - `PaymentController::choosePaymentAction()`
      Presents user with payment method dialog, then forwards to `completePaymentAction()` on POST.
    - `PaymentController::completePaymentAction()`
      Processes the payment request by interacting with `JMSPaymentCoreBundle:PaymentPluginController`. Redirects the user to the external URL if requested by the plugin controller.
- `DemoController`: controller for demo pages; allows setting up a test order with desired payment amount for testing.

`UiucCmsAdminBundle`

Provides features to admins such as sending mass mail and changing user status.

- `DefaultController::demoteAction()`: Demotes the selected user if the selected user status is admin and the current user logged in is a super admin

- `DefaultController::promoteAction()`: Promotes the selected user if the selected user status is normal user
- `DefaultController::removeAction()`: Removes the selected user if the selected user status is normal user
- `DefaultController::mailAction()`: Asks user to input subject and body and sends the user using Swiftmailer
- `Mail`: model object for mail. Stores sender, recipients, Mail subject, and Mail content

`UiucCmsUiPayPaymentBundle`

Provides payment backend plugin for `JMSPaymentCoreBundle:PluginInterface`. This bundle is maintained in a separate repo for confidentiality concerns (https://github.com/toukoaozaki/UiucCmsUiPayPaymentBundle).

- `UiPayClient`
  Abstraction for interacting with iPay backend servers.
  `UiPayClient::registerPayment()` is first used to register payment to the iPay backend, which in turn returns payment tokens and the url to redirect users on success.
  `UiPayClient::capturePayment()` is called after successful authorization to finalize the transaction. Performing such capture is an important part of payment security, as authorization status needs to be verified; return endpoint can be reached arbitrarily by user.
- `UiPayPlugin`
  `JMSPaymentCoreBundle:PluginInterface` for University of Illinois iPay payment system. This plugin implements `approveAndDeposit()` method using `UiPayClient` services.
- `Payment`
  Model for payment data specific to iPay payment system. This includes token, payment type and transaction id.
- `ReturnController`
  Controller to handle returns from iPay payment service. Upon return receipt, the controller updates transactions then forwards to the appropriate return url requested by the initiator through `JMSPaymentCoreBundle:PaymentInstruction`.
- `CurlHttpClient`
  Provides abstractions for HTTP requests using cURL. Given request type, url and arguments, `CurlHttpClient::request()` returns the response body as string
- `TimeProvider`
  Provides current time via `TimeProvider::now()`.

`UiucCmsTestUtilityBundle`

Contains utility/base base classes for the entire project.

- `IdentityTranslator`
  Implements Symfony `TranslatorInterface` that performs no translation. This class is useful for unit tests as the resource identifiers are relatively stable compared to the text.
- `FixtureBase`
  Base class for test data fixtures using Doctrine `FixtureInterface`. Guarantees that the fixtures are only loaded on test environment.

- `FunctionalTestCase`
  Base class for functional tests requiring authentication and data fixtures. Provides helper functions to simplify authentication and fixture loading within the test.

4.2 UML diagrams

4.2.1 Class diagram

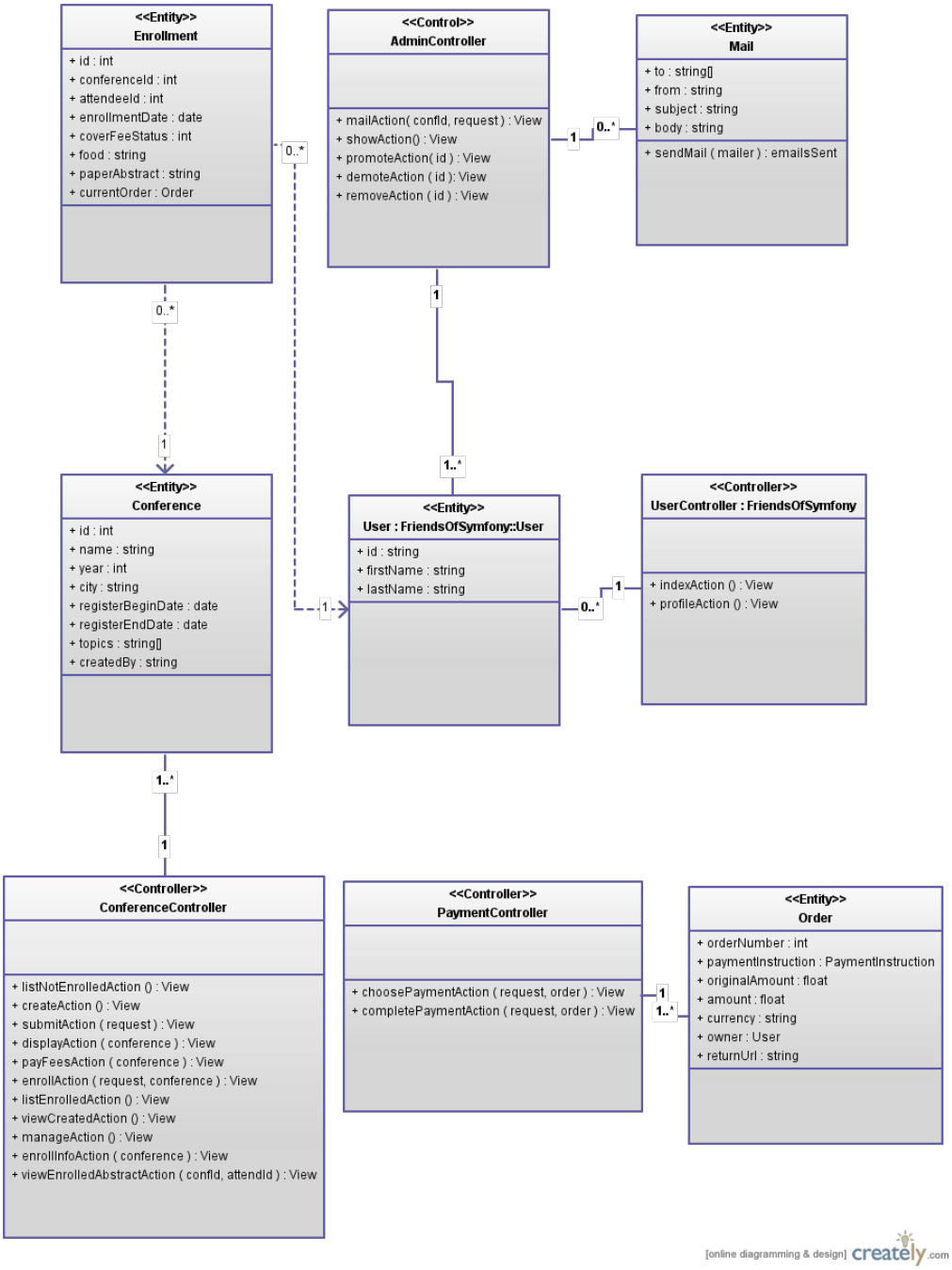

Figure 4.1 Class diagram

4.2.2 Sequence Diagrams:

- Conference registration:

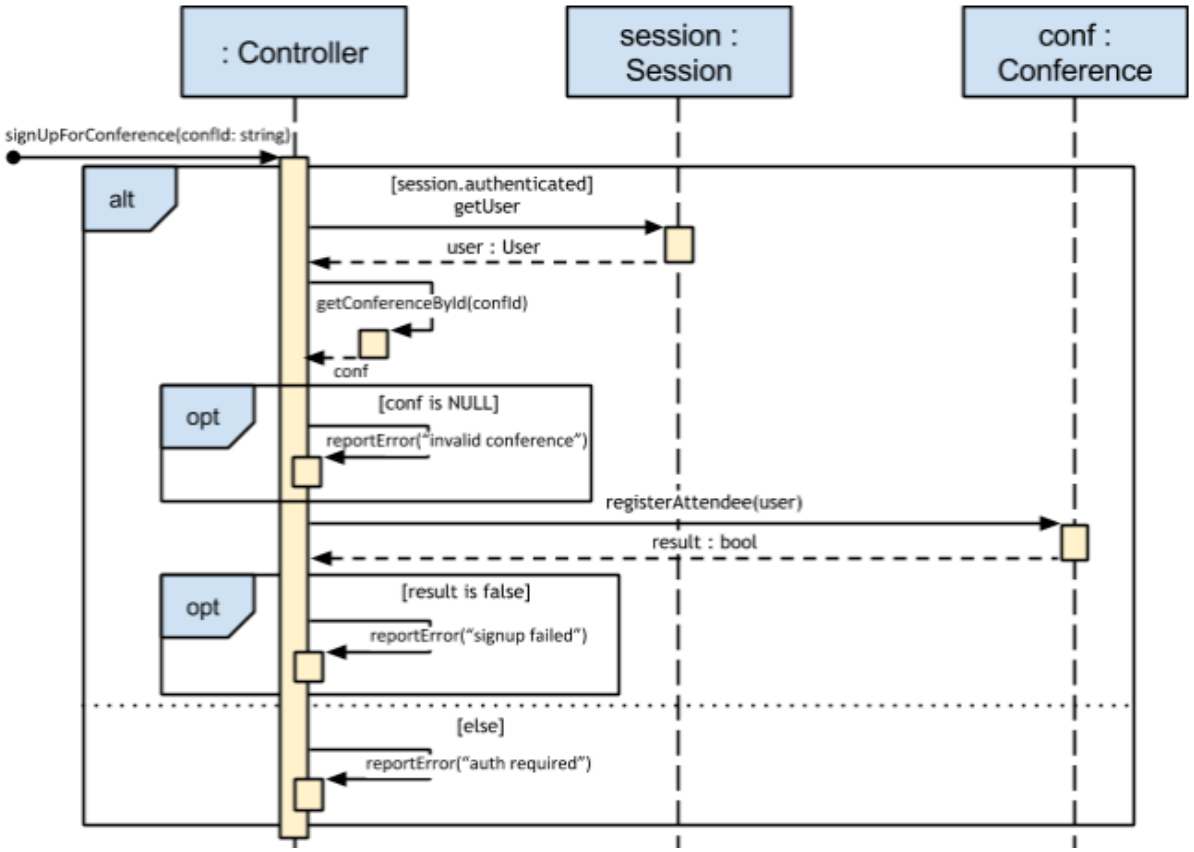

Figure 4.2 Sequence diagram of conference registration

- Make Payment:

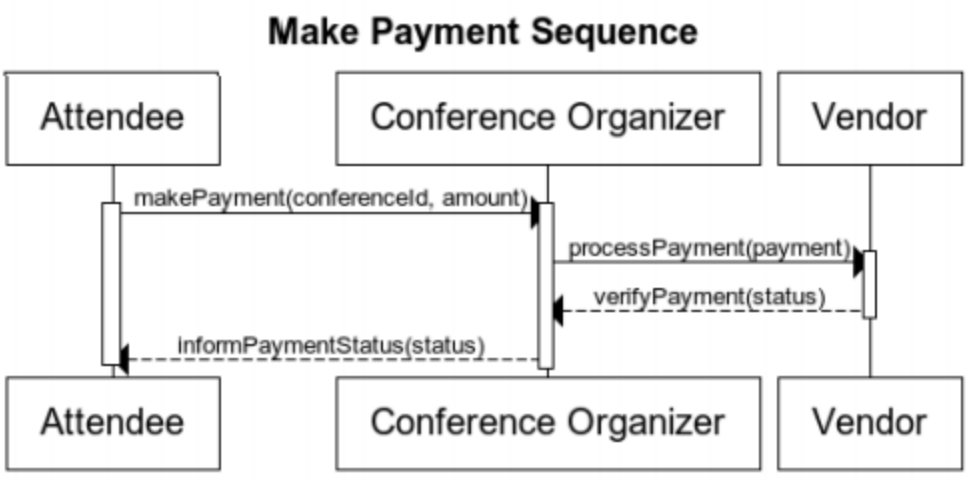

Figure 4.3 Sequence diagram of making payment

- Add Abstract:

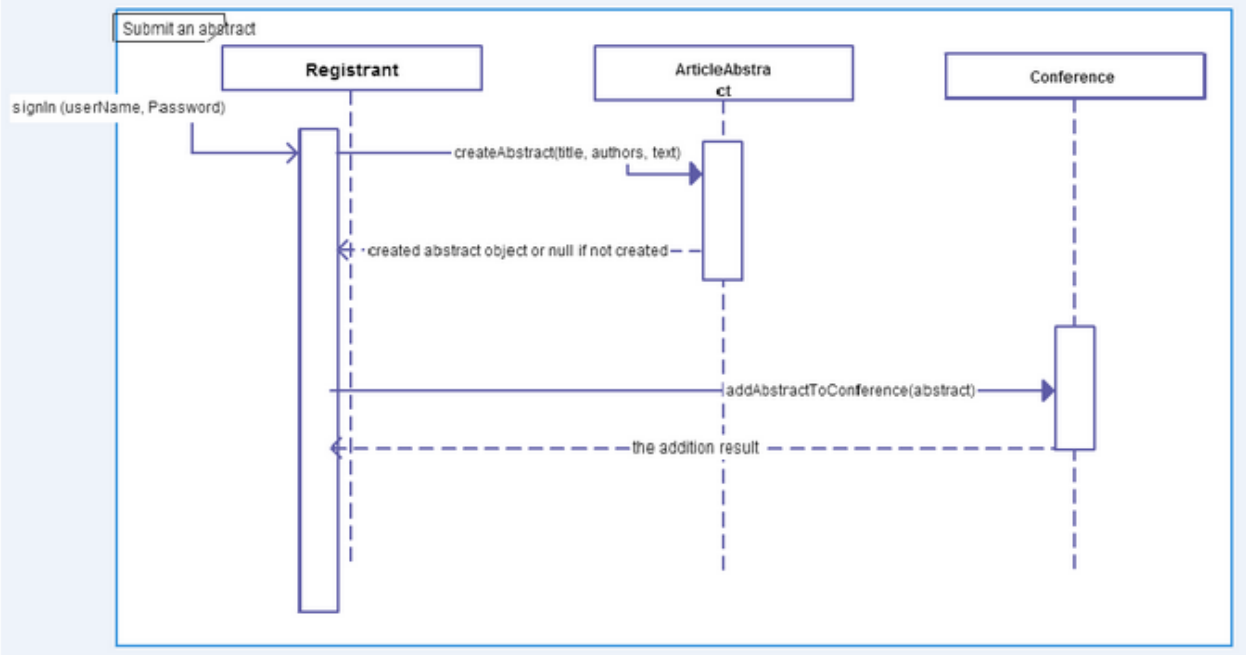

Figure 4.3 Sequence diagram of adding abstract

4.3 Major components of the CMS.

The major components of the CMS system are the individual bundles of our project. These include:

- AdminBundle: Administrator functionality including adding/removing admin privileges and sending mass emails to conference subscribers
- ConferenceBundle: Conference management composed primarily of conference creation/modification and user sign up for conferences
- PaymentBundle: Cover fee payment system that interfaces with the university's financial transaction system iPay
- SiteBundle: Implementation of the user interface
- TestUtilityBundle: Tools to assist testing including fixtures and base test classes
- UserBundle: User profile management

4.4 Framework's influence on the design

The biggest components of Symfony that influenced our design were the use of bundles, Doctrine, MVC model, and twig. A bundle is a well defined directory provided by Symfony that enabled the integration of the model-view-controller pattern as well and the well-defined separation of the various components of our system. Furthermore, Symfony comes built in with twig, a template engine for PHP. Thanks to twig, we were able to separate business logic from view logic. In order words, it allowed us to not have to include php code that generates html code in our controller classes. Finally, Symfony is integrated with Doctrine, a library that provides powerful tools to easily persist and read information from the database. With Doctrine, we were able to enforce ORM without having to write raw SQL queries, whose syntax slightly varies between different database systems.

5. Future Plans

5.1 Immediate Extensions

To make the Conference Management System more powerful, we are considering adding the following extensions if time permits.

1. Display the presentations to public after the conference.
2. Allow administrators to select a previous conference and do query of previous conference information, such as number of attendees.
3. Search of attendee's historical preferences.
4. Allow attendees to download calendar to to their smart phones.
5. Add conference schedule alert features.
6. Allow administrators to close conferences from further registrations.

RailTEC will have a developer continuously work on this conference management system to add new features. For all complex system, this kind of long term evolution is inevitable. Giving it more work, this might become a sophisticated system in the future.

5.2 High-Level Goals

Following ideas might help the long term evolution of the system:

1. Allow attendees to use social network accounts to register.
2. Providing online virtual exhibition service to supporters.
3. Allow registrants to reserve hotel rooms online.

5.3 Personal Reflections

Jiangchuan Zhou - Being my first time using a framework, this project provided a great experience to work with them. In the beginning, it was intimidating seeing all the Symfony automatically generated files and folders. However, as I followed the numerous tutorials provided, I developed an understanding of Symfony and frameworks in general. While I initially thought that the learning was time consuming and just making a simple implementation take a lot longer than without a framework, I also realized the many conveniences it brought to the project. Many thanks to the course and group members for providing such a great learning experience!

Eunsoo Roh - As a backend-inclined person, my experience into the web hasn't been quite deep. This project gave me an opportunity to practice web development in all aspects, including both frontend—Twig, Bootstrap and jQuery—and backend—PHP and Doctrine. Also, this project provided me with valuable experiences dealing with technical/management troubles, so I can be better prepared for my next project. Finally, learning and reverse-engineering how various payment systems work gave me more insights on how to integrate a web service with a payment backend.

Derek Quach - In the past I used inline PHP to accomplish what the Conference Management System did with an actual framework. The Symfony PHP framework was challenging to learn since I had zero experience but in the end produced much more manageable and readable code. In spite of its start up cost, Symfony made extending our website and project much easier down the road. Learning to use a PHP framework was definitely the most valuable lesson I learned during my time working on the Conference

Management System. Of course, this has also allowed me to gain more experience with working in groups and following a more structured software development process. Maintaining good practices while developing really does create more maintainable and more manageable code. I have learned a great deal during the course and give my thanks to my team who taught me a great deal as well.

Kirill Mangutov - Working on the Conference Management System exposed me to the many responsibilities involved in working on a project for a client. When working on class MPs or personal projects, there are few tasks besides hammering out code. For our client facing application, I had to design a wiki page hierarchy to both best reflect our design process and make it easy for people outside our team to learn about our project and process. As we wrote code, we would add any challenges we faced to the user stories they correspond to make it as easy as possible to maintain the code.

Alexander Hadiwijaya - This project has given worth experience in using framework and MVC. Conference Management System has also given me deeper understanding between users and administrator regarding permission and security. Often time we stumbled upon security breaches between users and administrators. However, that helped us to always remember privileges between users and administrators. This course and RailTEC has given us an opportunity to do this project and made us experience great things. Thank you for all team members for the best experience.

Han Jiang - It's been a great experience for me to work on this project using MVC model and a powerful framework, symfony. Having worked on a multi tier project before, I find that using MVC model has its tremendous advantages, such as security and scalability. Though the learning curve of symfony is steeper than I thought, it worths the effort because of a great number of features supported by symfony, which in turn will improve the efficiency of the development in the long run. Most importantly, I am really excited about this project of making a useful conference management system. I am also extremely satisfied with the people in the CMS team, who demonstrated strong motivation and talents.

6. Appendix

6.1 Installing Conference Management System

1. See https://github.com/uiuc-cms/uiuc-cs428-cms for installation instructions.