CS 361 Software Engineering I

HW4: Design Assignment

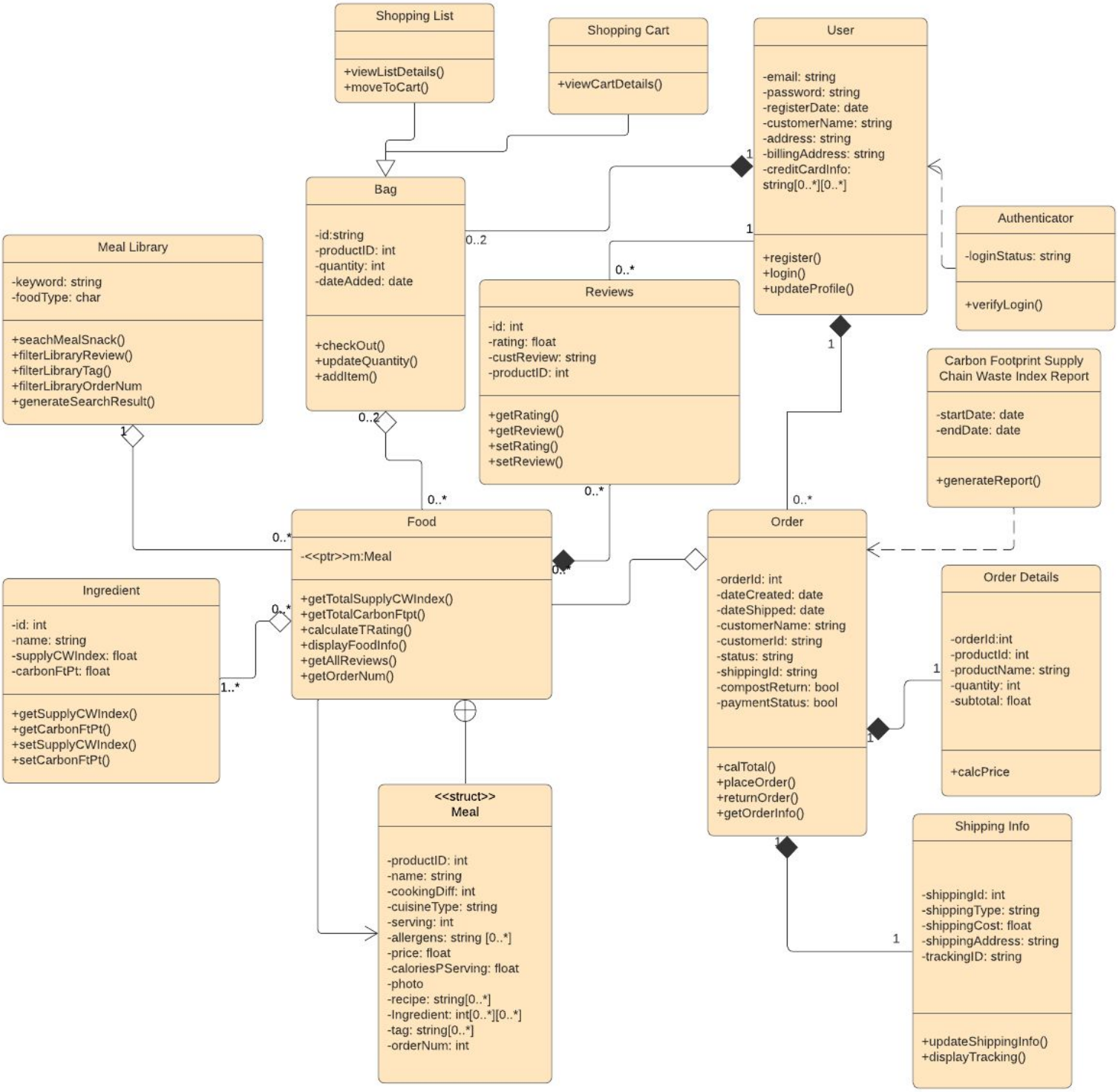
Meal planner to promote healthy meals with low waste

11/10/2019

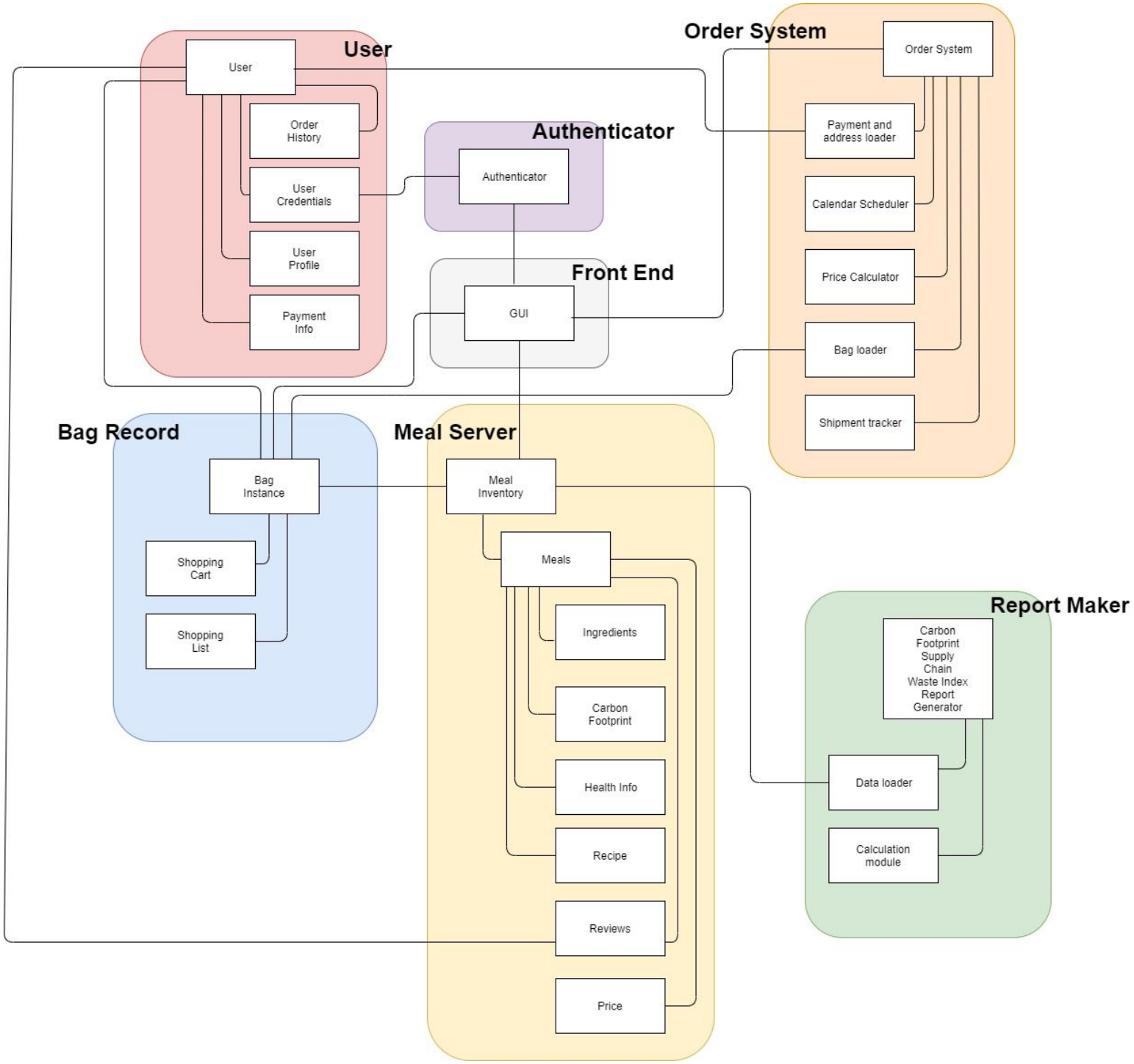
Group 24

John Casey III
Raymond Lieu
Todd Radin
NianJun Shi
Brian Shim

UML Class Diagram



Packaging


User
User
Order History
User Credentials
User Profile
Payment Info
Authenticator
Authenticator
Front End
GUI
Order System
Order System
Payment and address loader
Calendar Scheduler
Price Calculator
Bag loader
Shipment tracker
Bag Record
Bag Instance
Shopping Cart
Shopping List
Meal Server
Meal Inventory
Meals
Ingredients
Carbon Footprint
Health Info
Recipe
Reviews
Price
Report Maker
Carbon Footprint Supply Chain Waste Index Report Generator
Data loader
Calculation module

Coupling

Data Coupling
- Meal Inventory's price provides unstructured data to order system's price calculator
- User provides unstructured data to meal's reviews

Stamp Coupling
- Meal inventory provides structured data to bag instance
- Order system provides structured data to user's order history
- Bag instance provides structured data to order system
- Bag record, meal server, and order system provide structured data to the GUI

Control Coupling
- Payment and address loader calls user's payment info
- Order system's bag loader calls bag record
- Carbon footprint supply chain waste index generator calls meal's carbon footprint

Content Coupling
- Authenticator can modify user credentials

Cohesion

Functional Cohesion
- User encapsulates all the users' relevant data
- Meal Server encapsulates all the meals' relevant data
- Front end provides the user an interface to interact with the software
- Report maker creates the carbon footprint supply chain waste index report
- Authenticator validates users' credentials and the GUI

Procedural Cohesion
- Order System's modules are encapsulated to be a specific order (load bag, schedule the meal, calculate the price, load payment/address info, provide shipment tracking info)

Logical Cohesion
- Bag record holds shopping list and shopping cart. When a user transfers the meal to the bag, the user selects one or the other to be used.

Code Development

Our design will best support incremental development. Since users do not have to pay to use our application or purchase curated foods from the company, the core value in the application is in our meal/snack library. Other major functionalities such as order-placing, order-tracking and report-generation are "add-ons" and are dependent on the meal library.

For example, order-placing is dependent on the user's selection of a meal or snack; order-tracking is dependent on the order's existence and the generation of the supply waste and carbon footprint report is dependent on the meals or snacks the user has ordered. These dependencies require the meal library to be logically in place before they can be functional.

Design Patterns

Adapter

The adapter pattern will be helpful in allowing our system to interface with other components that might otherwise be difficult. One example of this is with the user interface and how we receive Pre-condition from the user. Different phones and browsers can render Pre-condition differently, so an adapter that will provide an interface to our system will be really valuable. Another example would be interfacing with third party vendors such as payment and shipping providers. We will need to be able to work with various providers and this pattern should help with that.

Facade

Since our system provides a front end user interface to the user, the facade pattern will obviously be valuable. The user should not need to know the inner workings of our system, but rather the complexities of the system should be abstracted away by the GUI. This pattern will also be useful internally. For example, communicating with the database involved many complicated interactions. The complexities should be abstracted away by various APIs that will hide the complexity from the developer.

Memento

The memento pattern would be useful in allowing our system to return to a previous state. Since our system allows user to add items to their cart and remove them, it would be very useful to be able to be able to return to a previous state. This would also help state changes when users add items to their shopping list or remove them.

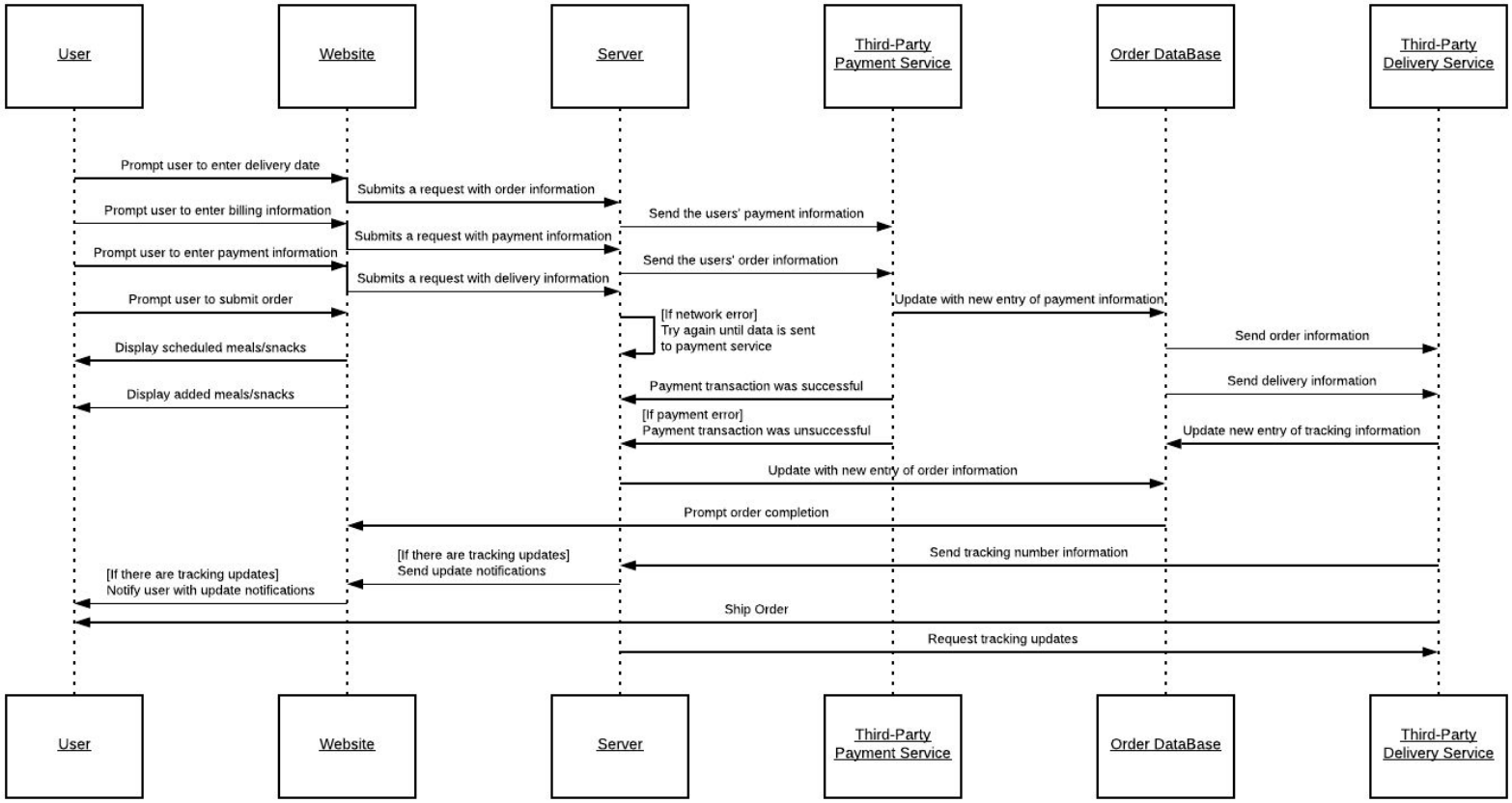
Interpreter

The interpreter pattern will be useful to our system as we will be making many calls to the backend database. In addition we will need to make many translations from the various languages used in web development. All of this will require language translation and parsing that this pattern should help address.

Observer

The observer pattern will be very helpful to our system in providing updates to the user interface as the user interacts with our system. An example of this is when a user adds scrolls down a meal listing page. We will need to know when the user reaches the end of the page so that we can load additional items when the bottom has been reached.

Use Case #2 Sequence Diagram



Interfaces

Meal order system

- Pre-condition: Meals to be ordered, calendar date for scheduling, payment info
- Post-condition: Tracking information for meals, meal added to user's order history, third-party delivery service notified of order

Carbon footprint supply chain waste index report system

- Pre-condition: User has placed orders and selects range of dates
- Post-condition: Carbon footprint supply chain waste index report during the date range

Meal review system

- Pre-condition: Meal to be reviewed, text for the review, number to indicate the rating
- Post-condition: Text review with rating listed under the meal and user profile, meal rating adjusted

View tracking information

- Pre-condition: Order number
- Post-condition: Tracking information

Authenticator

- Pre-condition: Username and password
- Post-condition: Log in to the site if successful, redirect to re-enter login info if failed

Filter meals

- Pre-condition: User selects filters such as cooking difficulty, types of cuisine (Italian, Mexican, etc.), serving size, allergens, and rating, calorie range, total order number from all users.
- Post-condition: Website displays filtered meal results

Exception Handling

Ordering ingredients to prepare scheduled meals

- The user does not fill in or incorrectly fills in all the required data needed by the server before submitting order.
    - In the case that the user enters incorrect billing or delivery information, an exception would be thrown at the server, from the client, to require the user to enter valid information. The exception handler would inform the client to request valid data from the user. The exception handler would also log the transgression and send the user a warning notification.
    - In the case where the user ignores the warning notification, the administrator would also receive an exception handler warning to contact the user directly to correct their invalid information before submitting an order.

Third-party service access database for user information

- The third-party service requests data from the database which they do not have permission to access.
    - In the case that the third-party service attempts to gain access to the database, an exception handler compares the users' clearance to the clearance needed to access the data and sees a discrepancy. A warning notification is displayed to the third-party service warning them to not attempt to access data they do not have the clearance for. The exception handler logs the incident for the database system administrators to review and report.

General exceptions

- User provides incorrect login info
    - If the user provides incorrect login info, an error will be displayed to the user indicating that the information is incorrect. The user will be redirected to the login page with the error message up to five times until the user inputs the correct information or the user is locked out of the account.
- Network failure which the client(client application) cannot reach the server.
    - In the case of the user, the data will be saved by the client as continual attempts are made to send the data to the server. The exception handler will display a popup that informs the administrator of the issue and instructs them not to fill any orders until the server can be successfully reached.
    - In the case of the third-party service, the exception handler displays a message that the client could not reach the server. The third-party service can continue to

ship out orders before the network failure and update the tracking notification to the database but cannot ship any orders that were affected by the network failure until further instructions from the client.

- In the case of the client cannot connect to the database, the exception handler will inform the client of this issue and asks to try again after the issue resolved.

Team Member Contributions

Here is a list of team member contributions for HW4:
All - Communicated on Slack throughout the week and added contributions to Google doc
Brian Shim - sequence diagram for use case, exceptions, review
John Casey III - unavailable this week due to family emergency
NianJun Shi - UML class diagram showing the key OO entities, code development, packaging feedback, review
Raymond Lieu - Packaging diagram, coupling/cohesion, interfaces, UML feedback, review
Todd Radin - Design patterns, packaging feedback, review