CS414 – Object Oriented Design

Take Home Midterm Exam

Taken By: Bryan Saunders

Honor Pledge

"I have not given, received, or used any unauthorized assistance."
Signature: *Bryan Saunders*
Date: 10/9/2013

CS414 Midterm Exam
Bryan Saunders

Question 1

Representational Gap is a measure of how different the domain model of an application is from its actual design implementation. Low Representational Gap means that the domain model and design are very closely aligned. The primary example of this is the having the classes in the design use the same names and functionality as the domain.

The advantage of LRG is that it leads to code that is easier to read, comprehend, and modify because it closely matches the real world domain model. For instance people will immediately know that the class named "Airplane" is supposed to be an airplane and do airplane things.

Domain modeling is critical to achieving LRG. The domain model provides a general guide for the designer to follow when building the system and designing the classes of the system. Without the domain model, it would be extremely difficult to achieve a LRG without the developer also being a subject matter expert in the field.

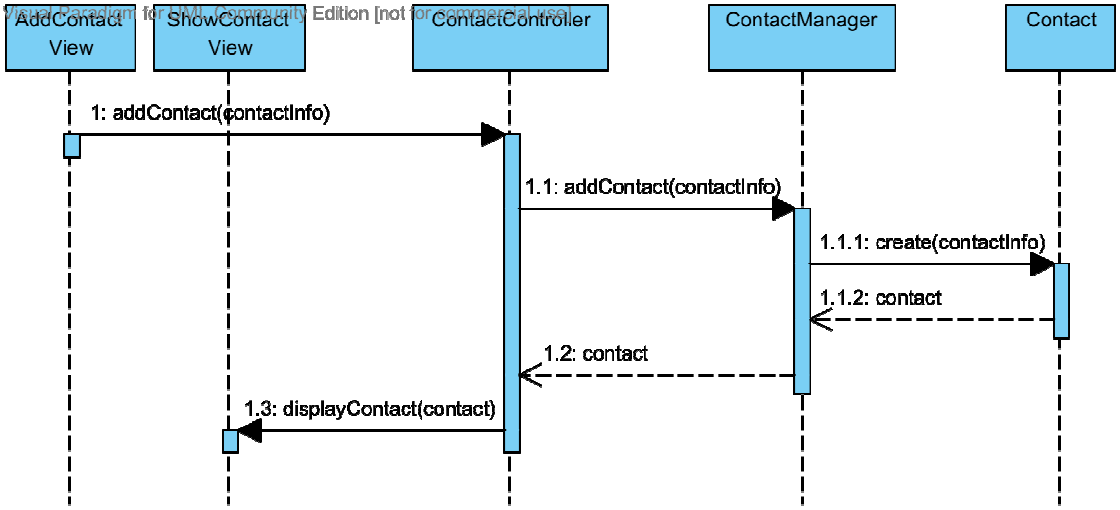
CS414 Midterm Exam
Bryan Saunders

Question 2

The Controller GRASP pattern uses a single non-view class to handle one or more specific incoming system events. This class could represent a single use case, or the system as a whole. The controller class serves as the middle man between the view and model layers of the system, and is generally considered to be part of the service layer.

Because it sits in the middle of the view and model layers, it provides a separation of the two. This way the model does not need to know anything about the view, and the view needs to know nothing about its underlying model. This separation decreases the coupling between the layers, and allows them to be changed without impacting the other layers.

One example of this separation is a simple CRUD system for managing contact information. Consider a use case for adding a contact where the user fills out a form on one view, submits it, and is redirected to a second view to see the created contact. In this scenario the system needs 2 views, one for adding the contact and one for displaying the contact, a controller, a helper/manager, and a model.



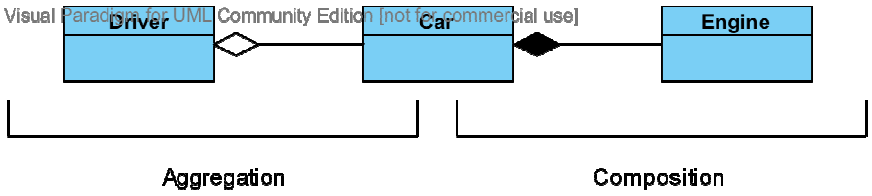
As demonstrated by the Sequence Diagram above, the ContactController class receives a command from the AddContactView telling it to add a new contact. It then delegates that command to the appropriate helper method in the business layer (ContactManager) who then creates the Contact from the model layer. Once the Contact has been created and returned, it sends the Contact information to the ShowContactView to be displayed back to the user. Because the view and model layers are only talking to the ContactController, they are separated from one another.

Question 3

Composition can be thought of as a 'part-of' relationship and is demonstrated with a filled diamond in UML. For example when modeling a Car, we would say that an Engine is part-of a Car. In Composition, the lifetime of the child object is controlled by the lifetime of the parent object so when the parent object is destroyed, so is the child object. In our example, if the Car object is destroyed, the Engine object will cease to exist as well.

Aggregation can be though of as a 'has-a' relationship and is demonstrated with an empty diamond in UML. Sticking with our Car example, a Driver has-a Car, but a Car is not part-of a Driver. When a Driver is destroyed its Car will still exist. This is because the lifetime of the child object, is not controlled by the parent object, and the child object will still exist even after the parent object is destroyed.

Below is a combined class diagram showing both Composition and Aggregation using our above car example.



Question 4

You should express system events at the level of intent rather than the physical input medium or interface widget to better capture the intent of the operation. It also allows you to remain abstract about the design implementation and stay uncommitted to the input interface that is used. This allows you to avoid implementation details while designing and modeling.

Consider the use case of an ATM machine that allows you to deposit checks. A good name for the system event could be 'inputCheck' because there are multiple ways you could enter a check into the ATM. It could be using a scanner, a camera, or manually entered. Some poor names for the system event would be 'scanCheck' or 'readCheck'.

Question 5

Diagram 1

Diagram 1 has low coupling because class A only communicates with one other class, class B. Even though class B calls class C, who then calls class D, this does not have any impact on A's coupling. Class A is also cohesive because it is focused on a single goal. Because of class A's low coupling and high cohesion, diagram 1 is the ideal design.

Diagram 2

Diagram 2 has no coupling at all since class A does not communicate with any other class but itself. I would also say that class A has very low cohesion since it only calls itself. This is based on the fact that the other two diagrams call out to other classes.

Without knowing what class A actually does and having more useful method names, it is hard to say for sure what level of cohesion it has. It is entirely possible that class A only has a single public operation and the other 3 calls are private helper methods, in which case it would have high cohesion.

Diagram 3

Diagram 3 is the worst of all of the diagrams in terms of coupling. Class A calls out to all three other classes, increasing its level of coupling significantly over the version of class A in diagram 1. However class A does have a high level of cohesion because it is relying on the other classes to do there works.

Question 6

Diagram A

Diagram A violates the constraint that 1 customer shops with a single cart because the Customer object ghosh as multiple Carts c1 and c2.

Diagram B

Diagram B violates the constraint that 1 cart contains 0 or more items because all 10 carts have the same Item i. Multiple customers could purchase the same type of Item, but they can not all purchase the same instance of that item.

Diagram C

Diagram C violates the constraint that 1 customer shops with 1 cart because all Customers g0 – g9 are sharing the same Cart c. Multiple Customers can not all use the same cart.

Diagram D

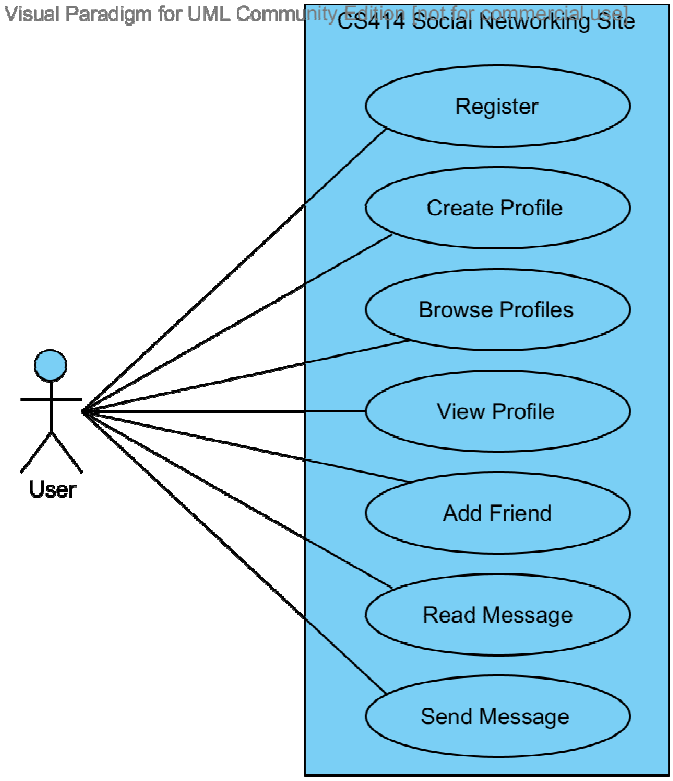
Diagram D meets all constraints.

CS414 Midterm Exam
Bryan Saunders

Question 7

Selected Application: Social Networking Site

Use Case Diagram



CS414 Midterm Exam
Bryan Saunders

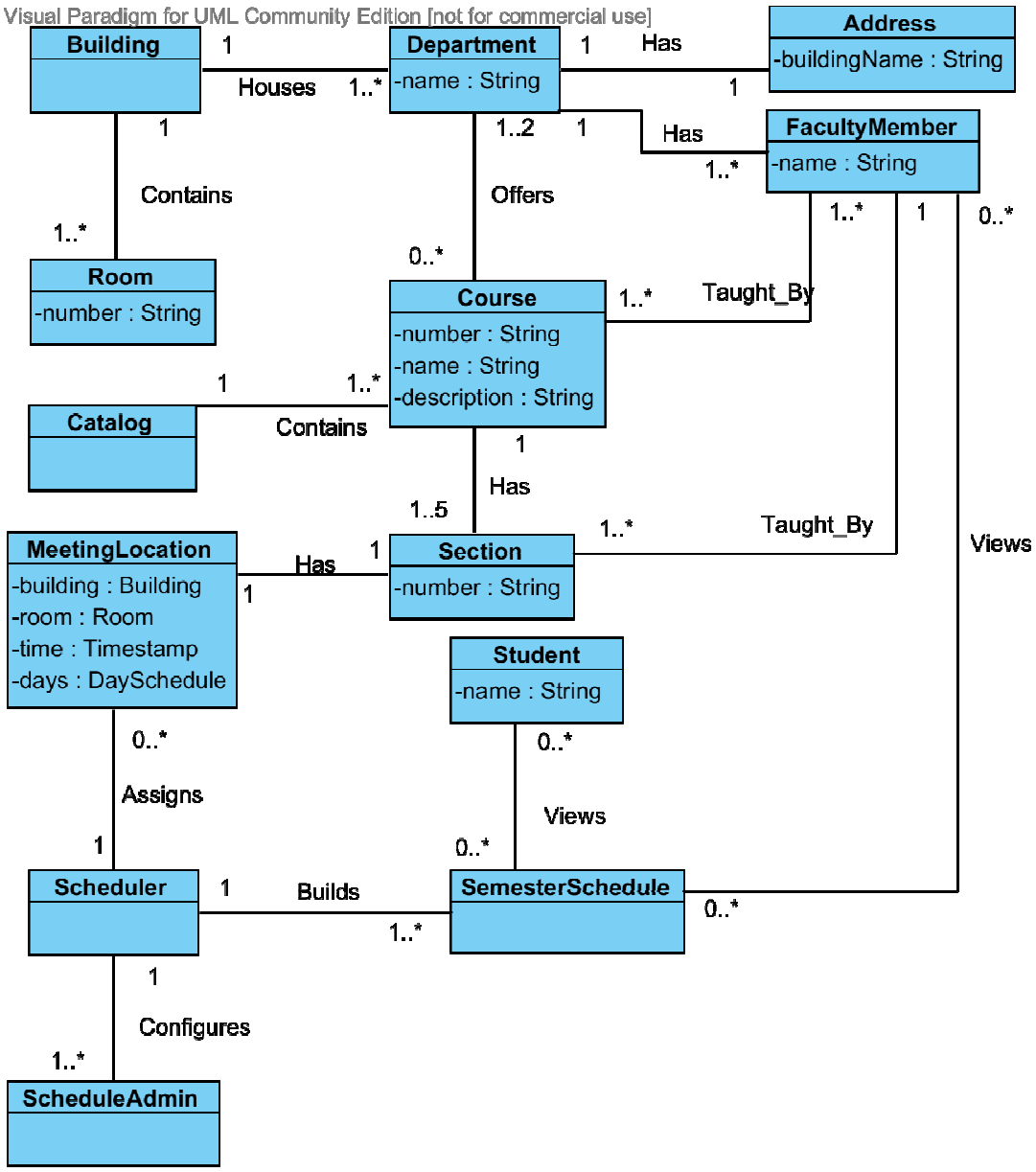
Use Case Scenario

Name	Create a Profile
Scope	CS414 Social Networking Site
Level	User Goal
Primary Actor	User
Stakeholders and Interests	User: Create profile to network with friends Owner: Users to create profiles to drive site traffic
Preconditions	User account created and user is logged in
Success Guarantee	User has created profile successfully
Main Success Scenario	1. User requests to create new profile 2. System loads create profile form 3. System pre-populates name and email field from account 4. User enters display name 5. User enters address 6. User enters phone number 7. User chooses to upload photo 8. System prompts user to select photo 9. User selects photo 10. System uploads photo 11. System displays photo on create profile form 12. User submits form 13. System validates user information 14. System displays success page
Extensions	Step 13: Invalid information entered 1. System loads create profile form 2. System pre-populates form with user data 3. System displays appropriate error messages 4. User corrects invalid fields 5. System continues at step 12
Special Requirements	None
Variations in Technology & Data	Step 13 Extension, Step 3: Error messages should be displayed next the field in error
Frequency of Occurrence	Every time a user creates a new account
Miscellaneous	None

CS414 Midterm Exam
Bryan Saunders

Question 8

Domain Model



CS414 Midterm Exam
Bryan Saunders

Domain Glossary

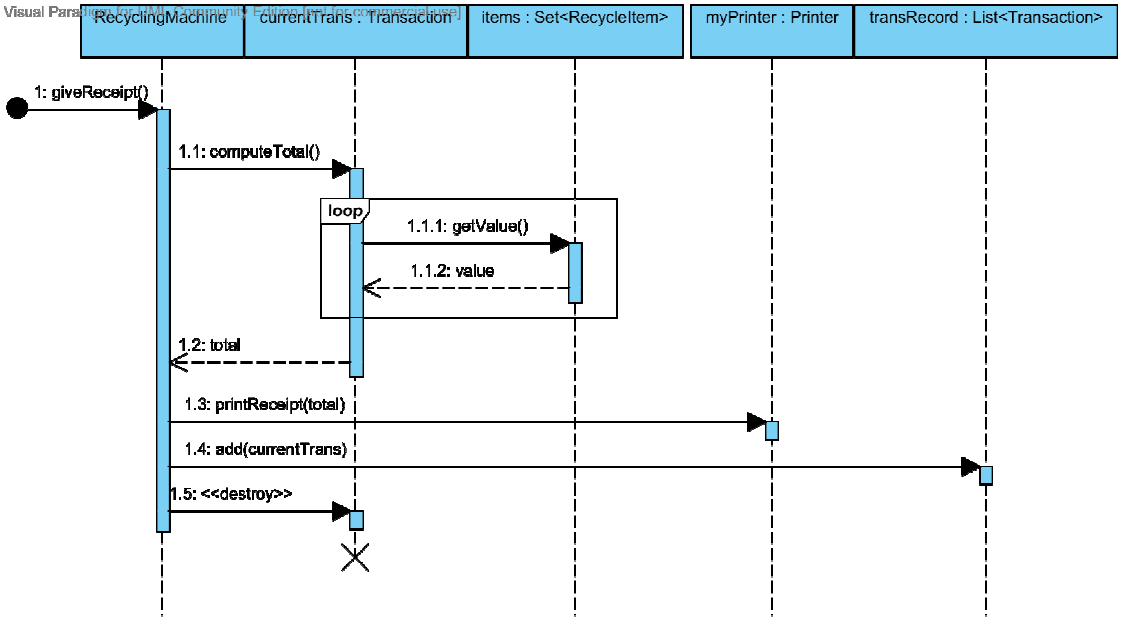
- **Building**
  - A physical building somewhere at the University
  - ***Associations***
    - Houses 1 or more Departments
    - Contains1 or more Rooms
- **Room**
  - A physical room inside of a building
    - Has a Number
  - ***Associations***
    - Housed by 1 Building
- **Department**
  - University department responsible for offering degrees within a specialized field
    - Has a Name
  - ***Associations***
    - Housed in 1 Building
    - Has 1 Address
    - Has 1 or more Faculty Members
    - Offers 0 or more Courses
- **Address**
  - A Campus location
    - Has a Building Name
  - ***Associations***
    - Belongs to 1 Department
- **FacultyMember**
  - A professor or other staff member at the University
    - Has a name
  - ***Associations***
    - Belongs to 1 Department
    - Teaches 1 or more Courses
    - Teaches 1 or more Sections
    - Views 0 or more Semester Schedules
- **Course**
  - Description of a course that is taught at the University. The actual teaching is done in course Sections.
    - Has a Name, Number, and Description
  - ***Associations***
    - Taught By 1 or more Faculty Members
    - Has 1 to 5 Sections
    - Contained in 1 Catalog
- **Catalog**
  - A listing of all courses offered at the university
  - ***Associations***
    - Contains 1 or more Courses
- **Section**
  - Teaching of a Course
    - Has a Number
  - ***Associations***
    - Belongs to 1 Course
    - Taught By 1 Faculty Member

        - Has 1 Meeting Location
- **MeetingLocation**
    - Campus location and time where a section meets to be taught
        - Has a Building, Room, Time, and Day Schedule
    - ***Associations***
        - Belongs to 1 Section
        - Assigned by 1 Scheduler
- **Scheduler**
    - Used to schedule meeting locations to sections automatically
    - ***Associations***
        - Assigns 0 or more Meeting Locations
        - Configured by 1 or more Schedule Admins
- **SemesterSchedule**
    - Schedule of all courses for a semester
    - ***Associations***
        - Viewed by 0 or more Students
        - Viewed by 0 or more Faculty Members
        - Built by 1 Scheduler
- **Student**
    - Anyone who attends the University
    - ***Associations***
        - Views 0 or more Semester Schedules
- **ScheduleAdmin**
    - Configures the Scheduler with Course information such as what courses are offered, how many sections are required, and what faculty will teach them
    - ***Associations***
        - Configures 1 Scheduler

CS414 Midterm Exam
Bryan Saunders

Question 9

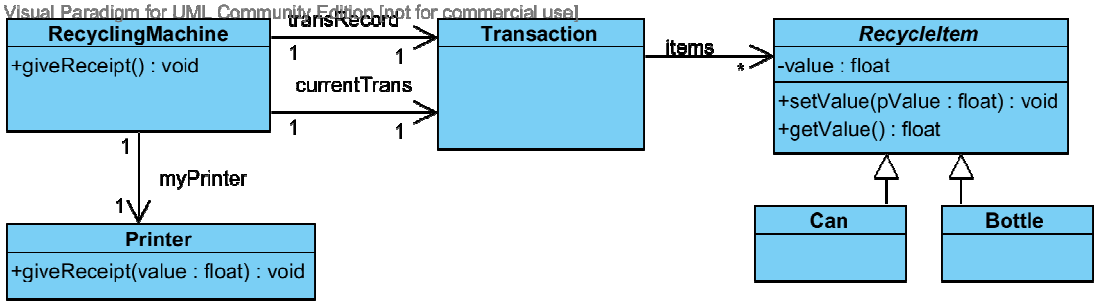
UML Sequence Diagram



Operation Calls

- **RecyclingMachine**
  - Public void giveReceipt()
- **Transaction**
  - Public float computeTotal()
- **RecycleItem**
  - Public float getValue()
- **Printer**
  - Public void printReceipt(float)
- **List<Transaction>**
  - Public void add(Transaction)

Class Diagram



CS414 Midterm Exam
Bryan Saunders

GRASP Patterns Used

- **High Cohesion**
  - The classes are all Highly Cohesive and focus on a single task. This allows the code to be easier to understand, allows for better separation of concerns, and makes the code more reusable.
- **Low Cohesion**
  - The classes have fairly low cohesion because they are only coupled to the classes they must be coupled with to function.
- **Polymorphism**
  - The RecycleItem abstract class and its children classes use Polymorphism so that a Transaction can take any Recyclable Item and be able to get its value. Using this pattern we could easily extend the system to take Paper or other recyclables simply by extends RecycleItem.
- **Controller**
  - The RecyclingMachine class serves as the System Controller since it is receiving the system events triggered by pressing the Receipt Button on the machine.