Object-Oriented Design Examples

CSSE 574: Session 4, Part 2

Steve Chenoweth

Phone: Office (812) 877-8974

Cell (937) 657-3885 Email: chenowet@rose-hulman.edu

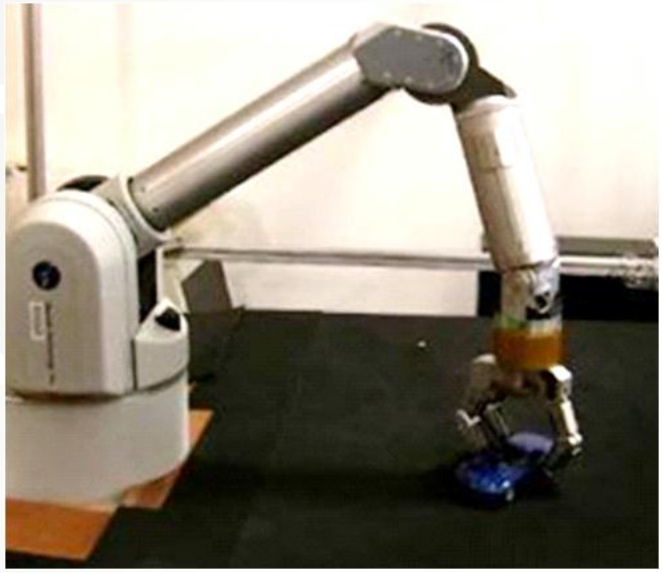
ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Plan for Today

- Quick recap of GRASP
- Command-Query Separation (CQS) principle
- Design examples

General, Responsibility Assignment Software Patterns Principles

Robot arm grasping at MIT. "Our system can move the robot arm to successfully grasp the object, even in situations where the part to be grasped is not visible in the input image. The input to our system is only one single 2D image (no stereo or range data) and one known 3D ground plane. It is an open-loop grasp (the mechanical action is only based on the initial command) without using any touch sensors or feedbacks from the robot during the grasp process." From http://people.csail.mit.edu/chiu/demos.htm.



Example: Design *makeNewSale*

Operation:	**makeNewSale()**
Cross References:	**Use Case: Process Sale**
Preconditions:	**none**
Postconditions:	○ **A *Sale* instance *s* was created** ○ ***s* was associated with the *Register*** ○ **Attributes of *s* were initialized**

Command-Query Separation Principle

- Each method should be either a command or a query
- Command method: performs an action, typically with side effects, but has no return value
- Query method: returns data but has no side effects

Why?

Extended Example: Grading System

(Warning – this could be on the Midterm!)

Problem Statement

The system will help instructors and teaching assistants provide thorough, timely feedback to students on assignments. The system will make grading more efficient, allowing students to more quickly receive feedback and course staff to devote more time to improving instruction.

The system will take a collection of student solutions to an assignment as PDF files or some other convenient, open standard. It will allow the grader to "write" feedback on student submissions. It will keep track of the grader's place in each assignment so that he or she can grade every student's answer to question 1, then question 2, and so on. Finally the application will create new PDF files including comments for return to the students.

Besides feedback, the system will help with calculating grades. The grader can associate points with each piece of feedback, so that the application can calculate points earned on the assignment. The grader will be able to drag remarks from a "well" of previous feedback to give the same feedback to multiple students (and deduct or add the same number of points). The points associated with a particular piece of feedback can be edited, causing the system to update the score calculations for every student that received that feedback.

A Sampling of Use Cases

- Create assignment
- Import student submissions
- Create feedback item
- Edit feedback item
- Add feedback to a submission
- Export graded student submissions

Domain Model and SSDs

- see handout…

Create New Assignment

Operation	*createNewAssignment(title, description, dueDate, authors)*
Cross References	Use Case: Create Assignment
Preconditions	none
Postconditions	▪ an *Assignment* instance, *assignment*, was created ▪ the attributes of *assignment* were set from the corresponding arguments ▪ a list, *instructors*, of new *Instructor* instances were created for each *author* in *authors* ▪ for each *instructor* in *instructors*, *instructor.name* was set to the corresponding *author* in *authors* ▪ *assignment* was associated with *instructors*

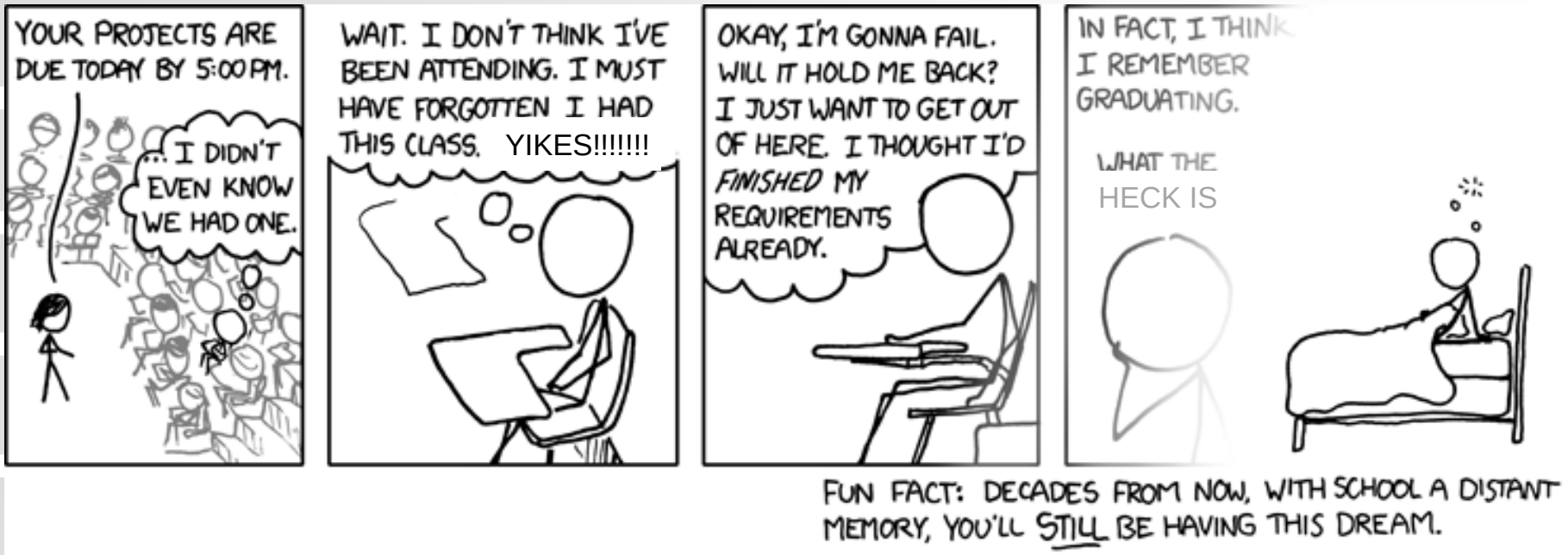
Create New Rubric

Operation	*createNewRubric(assignment, pointsAvailable, initialRequirements, authors)*
Cross References	Use Case: Create Assignment
Preconditions	*assignment* is an existing *Assignment* in the system
Postconditions	▪ a *Rubric* instance, *rubric*, was createdthe attributes of *rubric* were set from the corresponding argumentsa list, *instructors*, of new *Instructor* instances was created for each *author* in *authors*for each *instructor* in *instructors*, *instructor.name* was set to the corresponding *author* in *authorsrubric* was associated with *instructorsrubric* was associated with *assignment*

Add Requirement

Operation	*addRequirement(rubric, requirement)*
Cross References	Use Case: Create Assignment
Preconditions	*rubric* is an existing *Rubric* in the system
Postconditions	• *requirement* was appended to *rubric.requirements*

Students



The same goes for the one where you're falling out of the helicopter into the swamp. You guys all have that dream, right? It's not just me. Right?

Edit Feedback Item

Operation	*editFeedbackItem(item, title, points, comments)*
Cross References	Use Case: Edit Feedback Item
Preconditions	*item* is an existing *FeedbackItem* in the system
Postconditions	▪ the attributes of *item* were updated based on the other arguments

Your Turn…

- What would the Operational Contract look like for “Add Submission”?

Add Submission

Operation	*addSubmission(assignment, studentName, submissionData, submissionDate)*
Cross References	Use Case: Import Student Submissions
Preconditions	*assignment* is an existing *Assignment* in the system
Postconditions	▪ a new *Submission* instance, *submission*, was created*submission.studentAnswers* was set to *submissionDatasubmission.submissionDate* was set to *submissionDatesubmission* was associated with *assignment*a new *Student* instance, *student*, was created*student.name* was set to *studentNamesubmission* was associated with *student*