A COMPETITIVE-COLLABORATIVE APPROACH FOR INTRODUCING SOFTWARE ENGINEERING IN A CS2 CLASS

Swapneel Sheth, Jonathan Bell, Gail Kaiser
Department of Computer Science
Columbia University
{swapneel, jbell, kaiser}@cs.columbia.edu

@swapneel, @_jon_bell_
MOTIVATION

• Students at universities are increasingly competitive
• Grades are typically a zero-sum game
• This competition can discourage communication among students
• Competition may not lead to better learning
APPROACH

• Use competition to help students learn material better

• Encourage more communication by introducing collaborative aspects
COMS 1007

• Object Oriented Programming and Design In Java

• Second course for CS Majors and Minors at Columbia University (required for other engineering majors; also fulfills general science, CS requirement for non-engineering)

• Taught in Spring 2012

• Enrollment 129 (largely freshmen and sophomores)
COMS 1007

• Course Material
  - Object Oriented Design, Design Patterns, Interfaces, Graphics Programming, Inheritance and Abstract Classes, Networking, and Multithreading and Synchronization

• 5 biweekly assignments (theory and programming)

• In-class midterm and final
COMPETITION

- Battleship Tournament (Software Design, Interfaces)
- Gamification using HALO (Software Testing)
LEARNING GOALS

• Good Design

• Using Interfaces

• “Program to an interface, not an implementation”
/**
 * The game interface — this will control the Battleship game.
 * It will keep track of 2 versions of the "board" — one for each player.
 * It will let players take turns.
 * It will announce hits, misses, and ships sunk (by calling the appropriate methods in the Player interface/class).
 * @author swapneel
*/

public interface Game {

    int SIZE = 10;

    int CARRIER = 5;
    int BATTLESHIP = 4;
    int SUBMARINE = 3;
    int CRUISER = 3;
    int DESTROYER = 2;

    /**
     * This method will initialize the game.
     * At the end of this method, the board has been set up and the game can be started
     * @param p1 Player 1
     * @param p2 Player 2
     */
    void initialize(Player p1, Player p2);

    /**
     * This is the start point of playing the game.
     * The game will alternate between the players letting them take shots at the other team.
     * @return Player who won
     */
    Player playGame();
}
/**
 * The Location interface to specify how x and y coordinates are represented.
 * This can be used to represent the location of a ship or a shot.
 * If the location is a shot, the isShipHorizontal() method can return an arbitrary value.
 * @author swapneel
 */

public interface Location {

    /**
     * Gets the x coordinate
     * @return the x coordinate
     */
    int getX();

    /**
     * Gets the y coordinate
     * @return the y coordinate
     */
    int getY();

    /**
     * This method will indicate whether the ship is horizontal or vertical.
     * Can return an arbitrary value if the location is used to indicate a shot (and not a ship)
     * @return true if ship is horizontal, false otherwise
     */
    boolean isShipHorizontal();
}
public interface Player {

    /**
     * This method will place a ship on the grid.
     * This method should guarantee correctness of location (no overlaps, no ships over the edge of the board, etc.)
     * @param size the size of the ship to place
     * @param retry if an earlier call to this method returned an invalid position, this method will be called again with retry set to true.
     * @return The Location of the ship
     */
    Location placeShip(int size, boolean retry);

    /**
     * This method will get the new target to aim for
     * @return The Location of the target
     */
    Location getTarget();

    /**
     * This method will notify the Player of the result of the previous shot
     * @param hit true, if it was a hit; false otherwise
     * @param sunk true, if a ship is sunk; false otherwise
     */
    void setResult(boolean hit, boolean sunk);
}
TOURNAMENT

• 1000 games against simple AI player written by TA

• Used as seeding for Single Elimination Bracket

• Players with good strategies would win, proceed to next round against other students

• Quarterfinals, Semifinals, Finals in class

• Extra Credit points based on results of the tournament
RESULTS
RESULTS

Wanted to take part in the tournament

- 10.08%
- 89.92%
RESULTS

Wanted to take part in the tournament

- 10.08%
- 89.92%

Able to take part in the tournament

- 7.76%
- 92.24%
GAMIFICATION USING HALO

• Introductory CS classes do not focus on software testing [Jackson et al.; Elbaum et al.; Jones]

• “If it compiles and run without crashing, it must work fine”

• Low student interest and engagement in software testing [Elbaum et al.]
HALO SOFTWARE ENGINEERING

• Highly Addictive socially Optimized Software Engineering

• Use game-like elements and motifs to make software testing more fun, social, and engaging

• Quests disguise testing techniques like boundary value analysis

• Students complete quests => code gets tested better
HALO IN COMS 1007

• Halo Quests optional for HW2 and HW3

• For HW4, a small amount of Extra Credit could be earned

• For HW5 (Team Projects), students had to create HALO quests for Extra Credit

• “Learning by example”
CIA WORLD FACT BOOK
CIA WORLD FACT BOOK

ASSIGNMENT

• Use Java Networking to fetch data
• Parse HTML
• Answer Jeopardy-style questions
• List countries in South America that are prone to earthquakes

• Find the country with the lowest elevation point in Europe

• A landlocked country is one that is entirely enclosed by land. For example, Austria is landlocked and shares its borders with Germany, Czech Republic, Hungary, etc. There are certain countries that are entirely landlocked by a single country. Find these countries.
HALO QUESTS

• **TARDIS** - java program that can Transfer And Read Data from Internet Sites

• New Earth — This will probably be your first program that talks to the Internet. While this isn’t as complex as creating a new Earth, you should test out the basic functionality to make sure it works. Can you program read one page correctly? Can it read multiple pages? Can it read all of them?
HALO QUESTS (2)

**EXTERMINATE! EXTERMINATE!** - The CIA factbook has some unstructured data - not all of it is organized properly. This may not be as annoying (or life threatening) as Daleks, but your programs should be able to deal with this correctly and not crash (or get exterminated).

**Blink** — Your program doesn't need to be afraid of the Angels and can blink, i.e., take longer than a few seconds to run and get all the information. However, this shouldn't be too long, say 1 hour. Does your program run in a reasonable amount of time?
RESULTS - HW 5

Attempted the Extra Credit

37.90%

62.10%
RESULTS - HW 5

- Attempted the Extra Credit: 37.90%
- Perfect Score on the Extra Credit: 92.21%

7.79%
COLLABORATION

• Team Projects

• Lectures in Class
TEAM PROJECTS

• Most introductory CS courses allow little/no collaboration

• Real world projects are typically done in (large) teams

• Learn good (and bad) aspects about working in teams

• Added pedagogical incentive - give students the freedom to choose and deal with ambiguity that results
TEAM PROJECTS

Formal Design (CRC, Class diagrams, Sequence/State Diagrams)
Interfaces (Define and use)
Inheritance
Design Patterns

Java Graphics
Networking
Multithreading
Advanced Java (Data Structures, Reflection, External Libraries, etc.)
TEAM PROJECTS
TEAM PROJECTS

- 9.09%
- 33.88%
- 57.02%
TEAM PROJECTS

- Default Individual Assignment: 9.09%
- Custom Project, Team Size 2+: 33.88%
- Custom Project, Team Size 1: 57.02%
SAMPLE TEAM PROJECTS

• **Meal Planner** - use the USDA nutrition database, allow users to create and share recipes, learn about percentage values of food

• Navigate and read **SparkNotes** book summaries, allowing for offline use on a subway

• **What I should wear** - Clothes-recommendation system based on local weather

• Games like Guess Who?, Multiplayer Minesweeper, Mafia
LECTURES IN CLASS

• Two 75-minute lectures per week

• Stereotypes about CS being “geeky” and “boring” are not reinforced
  - Informal, collaborative classroom environment
  - Students encouraged to participate/comment
  - Group exercises
LECTURES IN CLASS

• Assignments are well-defined and precise, but real-world projects are typically not

- All assignments had elements of vagueness

- Team projects

- Lectures reinforced the importance of being flexible and accepting uncertainty is part of CS
FEEDBACK - QUALITATIVE

• Sources
  - Midterm and final course evaluations
  - Surveys filled out by students
  - Email
FEEDBACK - HALO

- Made it more fun, 3
- Made sure that I did everything, but would have anyway, 2
- Clarified assignment and what was expected of me, 4
- It helped, but I had other programming difficulties, 1
- It made sure that I did everything important and was more fun, 2
- Made sure that I did everything important for the assignment, 16
- Put me on the right track to complete the assignment, 11
The tasks were too trivial, and didn't help me, 12

Only benefit was extra credit - tasks were too trivial for me, 12

The quests got in the way and were unhelpful, 4

I had general programming difficulties aside, 1
FEEDBACK - COMPETITIVE ASPECTS

• “I really liked the class tournaments. If only there was a way to make them like mandatory.”

• “I think it’s awesome that you’re sneaking your taste in music into the HALO quests. The Coldplay references are hilarious. PLEASE make every HALO quest music-themed. It keeps me awake and happy as I do my homework.”
FEEDBACK - COLLABORATIVE ASPECTS

• “It’s really rare to feel like learning is a collaborative experience between the professor and the students, and even rarer to feel like it’s a collaborative experience amongst the students themselves, and you’ve been awesome at fostering that kind of environment.”
CONCLUSION

• Competition
  - Battleship Tournament (Software Design, Interfaces)
  - Gamification using HALO (Software Testing)

• Collaboration
  - Team Projects
  - Lectures in Class
A COMPETITIVE-COLLABORATIVE APPROACH FOR INTRODUCING SOFTWARE ENGINEERING IN A CS2 CLASS

Swapneel Sheth, Jonathan Bell, Gail Kaiser
Department of Computer Science
Columbia University
{swapneel, jbell, kaiser}@cs.columbia.edu

@swapneel, @_jon_bell_