Project Kickoff, Team Dynamics, OO Principles

Dr. Isaac Griffith Idaho State University

Outcomes

Idaho State University | Computer Science

After today's lecture you will be able to:

- Start on the Team Project
- Understand the basics of team dynamics
- Understand my high level grading for the project
- Begin to understand OO Design Principles

ROAR

K
I
C
K
O
F
F

Project Teams

CS 2263

Team Roles

Each team will need one of the following

- A Team Leader (Parts 1 - 5)
- A Person in charge of the Git (Parts 2 - 5)
- A Person in charge of design (Part 3)
- A Person in charge of development (Parts 3 - 5)
- A Person in charge of testing (Parts 3 - 5)
- A Person in charge of deployment (Part 5)
- A Person in charge of UI (Parts 2 - 5)
- A Person in charge of requirements (Parts 2 and 3)
- A Person in charge of build (Parts 3 - 5)

What?!?

But Professor, that’s way more jobs than there are team members…

ROAR

What?!?

But Professor, that's way more jobs than there are team members…

Welcome to Software Engineering… we all wear many hats

Team Dynamics

CS 2263

Team Dynamics

- A key to a great team project, is a great team
- Yet, these things don't happen by accident
- Typically, when forging a team or adding new members we look at how they interact with current team members
  - Normally a part of the interview process
- Unfortunately, that is not going to be the case here

Team Dynamics

- Your teams will start out as a random group of students

- It should be your goal to:
  1. Get to know each other personally (if you don't already), in a setting outside of class
  2. Get to know each other's strengths (technically)
  3. Get to know each other's weaknesses (technically)
  4. Start to create a "**Jelled**" team

Creating a “Jelled” Team

- A team of people so strongly knit that the whole is greater than the sum of its parts

- Characteristics of a jelled team:
  - Very low turnover rate
  - Strong sense of identity
  - A feeling of eliteness
  - Team vs. individual ownership of the project
  - Team members enjoy their work

Motivating People

- Motivation is the greatest influence on performance
- Monetary rewards usually do not motivate
- Suggested motivating techniques:
  - 20% time rule
  - Peer-to-peer recognition awards
  - Team ownership (refer to the team as “we”)
  - Allow members to focus on what interests them
  - Utilize equitable compensation
  - Encourage group ownership
  - Provide for autonomy, but trust the team to deliver

Handling Conflict

- Preventing or mitigating conflict:
  - Cohesiveness has the greatest effect
  - Clearly defining roles and holding team members accountable
  - Establish work & communications rules in the project charter
- Additional techniques:
  - Clearly define plans for the project
  - Make sure the team understands the importance of the project
  - Develop detailed operating procedures
  - Develop a project charter
  - Develop a schedule of commitments in advance
  - Forecast other priorities and their impact on the project

Other Difficulties

- This project will require time and planning
- It will also require that you and your team work together to ensure the work is done and is of high quality
- This means that as a team you need to:
  - Hold regularly scheduled meetings
  - Assign jobs to one another, and hold each other accountable for those assignments
  - Utilize technologies such as Zoom, Slack or Discord, and GitHub to coordinate your project

Project Grading

- The project is divided into 5 Parts
  - Week 06: Part 1 - Team Formation (5%)
  - Weeks 07 - 08: Part 2 - Planning and Specification (20%)
  - Weeks 09 - 10: Part 3 - Design and Initial Implementation (20%)
  - Weeks 11 - 12: Part 4 - Implementation and Test (20%)
  - Weeks 13 - 14: Part 5 - Release and Deploy (20%)
  - Week 15: Part 6 - Project Presentation (10%)
  - Team Reviews (5%)
- The team will receive a grade value, but the individual score for each team member will be weighted by a multiplier.
  - This multiplier will be derived from the feedback reports submitted after each part
  - Thus, as a team you may have done well, but individually you can still fail.
  - This acts a means to ensure accountability of each team member to the team.

Design Principles

CS 2263

ROAR

Design Principles

- This lecture is a collection of design principles for making better software.
- Every great programmer has a toolbox of design principles they use to help them produce great code
- Yes, these principles are admittedly fuzzy and not mutually exclusive
- They must be learned by specific coding examples/experiences

The Principles at a Glance

- SOLID Principles
  - **SRP** - Single Responsibility Principle
  - **OCP** - Open-Closed Principle
  - **LSP** - Liskov Substitution Principle
  - **ISP** - Interface Segregation Principle
  - **DIP** - Dependency Inversion Principle
- Others
  - **DRY** - Don’t Repeat Yourself
  - **LC** - Loose Coupling
  - **HC** - High Cohesion
  - **EV** - Encapsulate What Varies
  - **SoC** - Separation of Concerns

Solid Principles

CS 2263

Single Responsibility Principle (SRP)

Classes should not have more than one focus of responsibility

- Classes can reasonably be involved in different interactions, it is the *focus* that is the issue.
- This principle is similar to the cohesion principle and Separation of Concerns (SoC) principle
- Data-centric designs always violate this principle: the fat class in the middle with all the methods has many different foci.

SRP Example

- A way to figure out if certain methods belong in a given class:
- If not, move them to another existing, or new, class

```
class Automobile
    def start()
    def stop()
    def changeTires()
    def drive()
    def wash()
    def checkOil()
    def getOil()
end
```

- for each method **X**, ask, “does the **Automobile** have primary responsibility for **X**-ing?” If the answer is no, the method doesn’t belong.

SRP Violations

Finding

- Look for large classes with a lot of methods
    - Inspect these methods to determine if they satisfy the same responsibility
- Tend to cause interference between developers on same project
    - Leads to merge conflicts and other problem Areas

Handling

- If you have a class that violates this principle
    - extract methods belonging to one of the responsibilities into their own class
    - continue doing this until only one responsibility remains

For Next Time

- Review this Lecture
- Complete, as a team, Part 1 of the Project
- Come to Class

Are there any questions?