Software Engineering

(Engineering of Software Subsystems)

Spring 2015 - Course Overview

Instructor: Raghu Reddy

raghu.reddy@iiit.ac.in

Courses/Experiences so far…

- Undergrads at IIIT: You have learned technologies and low-level OO principles in workshop courses and some high level design in SSAD

- Grads at IIIT: Problem Solving course and Computer Scripting course

- All others (includes PGSSP): Some software development experience or an course at the undergrad level in Software Engineering/Systems Engineering or any other variant of it.

Underlying Assumptions (Pre-requisites)

- SE principles: Abstraction, Modularization/Decomposition, Coupling, Cohesion, etc.
- Some Technologies (for example, python, JavaScript, web2py, IDE's etc.) : at least 1 OOP language, & 1 RDBMS.
- Basic OO principles and implementations
  - Find the nouns → objects/state
  - Find the verbs → behaviors; methods/functions
  - Encapsulation, Inheritance, Polymorphism, etc.
- Introduction to static and dynamic modeling
- SDLC (Iterative Incremental process knowledge)
- Minimal SE practices (Version control, bug tracking, task management, etc.) and any associated tools.

Bottom Line: You should be able to CODE !!!

This Course is About…

Software Design

Course Details…

http://faculty.iiit.ac.in/~raghu.reddy/Software_Eng/CourseDetails.html

How do You Design?

- What do you think about?
- What considerations are important?
- When have you done enough?

What This Course is About

- Standard *patterns* of interactions between classes/ sub-systems?
  - Design patterns & Architectural patterns
- How to apply them to your application
  - Deal with subsystems at the higher level of abstraction provided by the patterns
- What to do when it does not fit exactly
  - Evaluate options and analyze the trade-offs
- How do you document the design knowledge (very important, given the focus on AGILE delivery models)

Do You Always Reinvent the Wheel?

- Consider code level patterns

  How do you walk through an array in Java?

```
for (i = 0;i < array.length;i++) {
    // use the array element
}
```

Our Design Level

- Higher than what we've done before
  - Not specific data structures
  - Not algorithmic approaches
- Lower than complex system level architectures
  - Not financial systems
  - Not air-traffic control
- Interactions of 8-15 classes in solution domain.

  i.e., the small sized subsystem

Problem-based learning methodology

- Solving problems motivates your learning
- Lecturing is minimal
- This is better because
  - Learner actively engages the material
  - Deeper learning when learner motivates need for knowledge
  - More closely resembles true career situation
- Over the past few years students were very positive about this teaching approach and seemed to learn a lot more