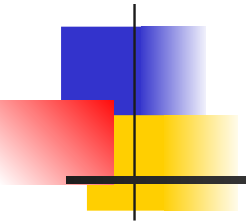


Introduction to the Course

Individual Software Process

Course Description

กระบวนกำรพัฒนำซอฟต์แวร์สมัยใหม่ กำรพัฒนำแบบ วนรอบและแบบค่อยเป็นค่อยไป กำรวำงแผนและประมำณ
โครงกำรเดี่ยว กำรจัดกำรเวลำ กำรติดตำมเวลำ คุณภำพรหัสโปรแกรม กำรปรับปรุงรหัสโปรแกรม กำรตรวจสอบรหัสโปรแกรม กำรควบคุมรุ่นของรหัสโปรแกรม กำรทดสอบซอฟต์แวร์เบื้องต้น กำรพัฒนำซอฟต์แวร์ภำยใต้กรอบงำน

Modern software development process, iterative and incremental development, individual project planning and estimation, time management, tracking time, code quality, code refactoring, code review, source code version control, introduction to software testing, software development under a modern framework.

How This Course Fits In

Developers work on projects in teams.

They use technology, tools, skills, knowledge to build products (using a process).

Individual Software Process - skills, knowledge, and habits for an effective developer

Workgroup Software Process - how to work effectively as a team. Apply a team process to a project.

Group Dynamics - ?

SKE technical courses - the knowledge you need

Topics in this Course

Conceptual Knowledge	Skills	Knowledge of Technology	Habits
Software processes, Iterative and Incremental Dev, Agile concepts	Planning Estimation Tracking Work Testing Code Review Build Management Refactoring Retrospective	Git JUnit, PyTest Mock Objects Task boards Issue tracking Tools for automation Ant, Maven, CI	Clean Code Self-learning Focus

Goal of the Course

Learn and apply basic software development skills needed for an individual developer

Prerequisite for this Course

1. Ability to code in Java, at level of Programming 2.

2. Git basics: create or clone a repo, update files, push changes.

3. How to use Github and Github Classroom.

See: **`https://skeoop.github.io/`**

Work and Grading

1. Weekly assignments - in lab and homework
2. Quizzes
3. Exams

Grading scale announced later.

Locations

Google Classroom. Assignments & announcements. Your feedback, discussion, and questions.

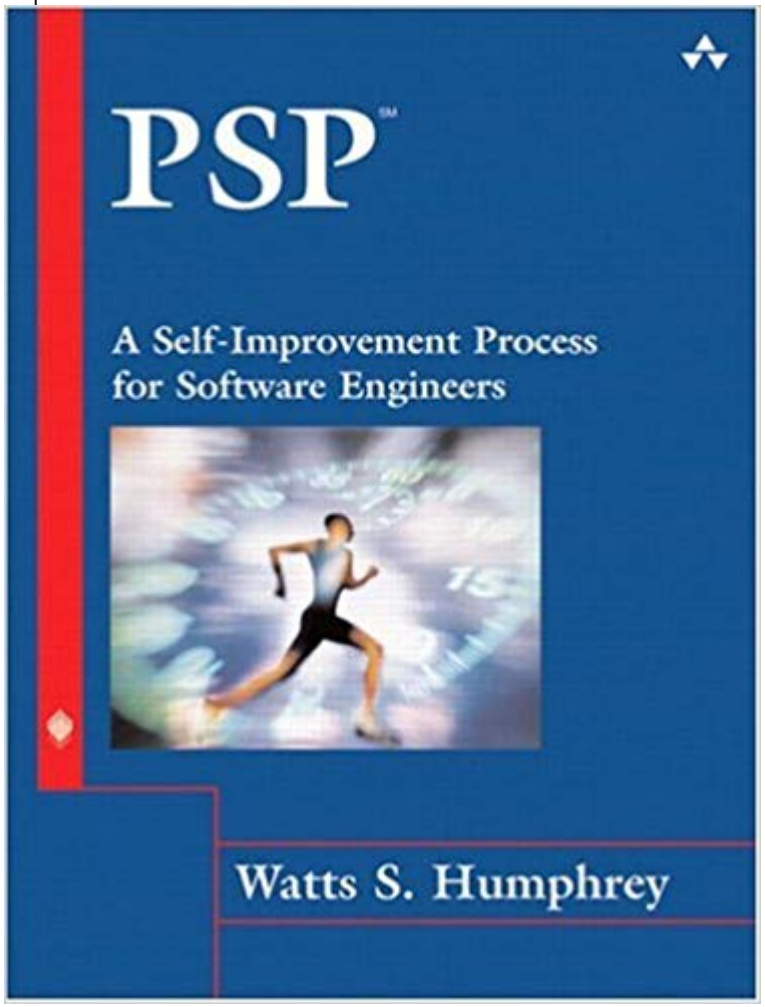
1. Goto classroom.google.com
2. Add this course using code: **`minx6nj`**

Course Material: **`https://cpske.github.io/ISP`**

Submit Assignments:

Some on Google classroom. Code on Github.

Original Syllabus of the Course



Step-by-step practices to build a personal process for:

planning

defect tracking

estimation

measuring quality & efficiency

evaluation

process improvement

Personal Software Process (PSP)

Objective: provide a disciplined process for SEs to manage their own work

- improve estimation and planning skills
- reduce defects in their products
- manage their own schedule & work quality
- improve their own software process

PSP progress through levels

PSP0: [baseline] measure time you spend on planning, design, coding, test, and *post mortem* (retrospective)

PSP0.1: measure output LOC. Add a coding standard and process improvement proposal (PIP).

PSP 1.0: Estimate program size using level 0 data. Make a test plan.

PSP 1.1: Add planning. Estimate time from program size.

PSP 2.0: Add design & code review. Emphasis on defect removal and prevention.

PSP 2.1: Add design specification.

PSP 3: Apply an iterative process to PSP2.1.

PSP Is Very Beneficial, but...

doesn't match what companies expect of developers.

Process

process -

a series of actions to achieve a particular result

What is a Software Process?

Software process - a sequence of actions for producing software.

Do You Have a Software Process?

What is your software process?

(discussion)

What did you do (activities, tasks, etc) in...

- Exceed Camp project?
- OOP PA5 project?

Summary of Student Software Processes

What are common activities in everyone's process?

Do You Have a Software Process?

Yes!

Everyone who develops software uses a process.

"Never thought about it" ...

- Process may be *implicit* or *informal*.

... "*its different for each project*"

- *ad hoc process*

The Most Common Software Process?

- "Code and fix"
- *ad hoc* (chaos)

Code and Fix

1. think about the problem, doodle ideas on paper
2. start coding
3. try to run what you've written so far.
   - if it doesn't work, fix the code
   - if it works, then add more stuff (step 2)
   - "more stuff" requires rewriting the old code -- modify the old code, run again, fix again, then add more stuff

"Code and Fix" is Inefficient

- No clear statement of what the result should be

- No design or "its in my head".

- Lots of rework.