CS3012 - INTRODUCTION TO SOFTWARE ENGINEERING

DR. VIVEK NALLUR

VIVEK.NALLUR@SCSS.TCD.IE

https://www.scss.tcd.ie/vivek.nallur/teaching/cs3012/

OUTLINE OF THIS TALK

- What is this module about?
- What do you need to do to pass?
- How will you be graded?

WHAT IS SOFTWARE ENGINEERING?

Engineering principles and practice that convert the art of programming into reliable software products/projects

THE 'ART' OF PROGRAMMING

Virtues of a good programmer (according to Larry Wall)

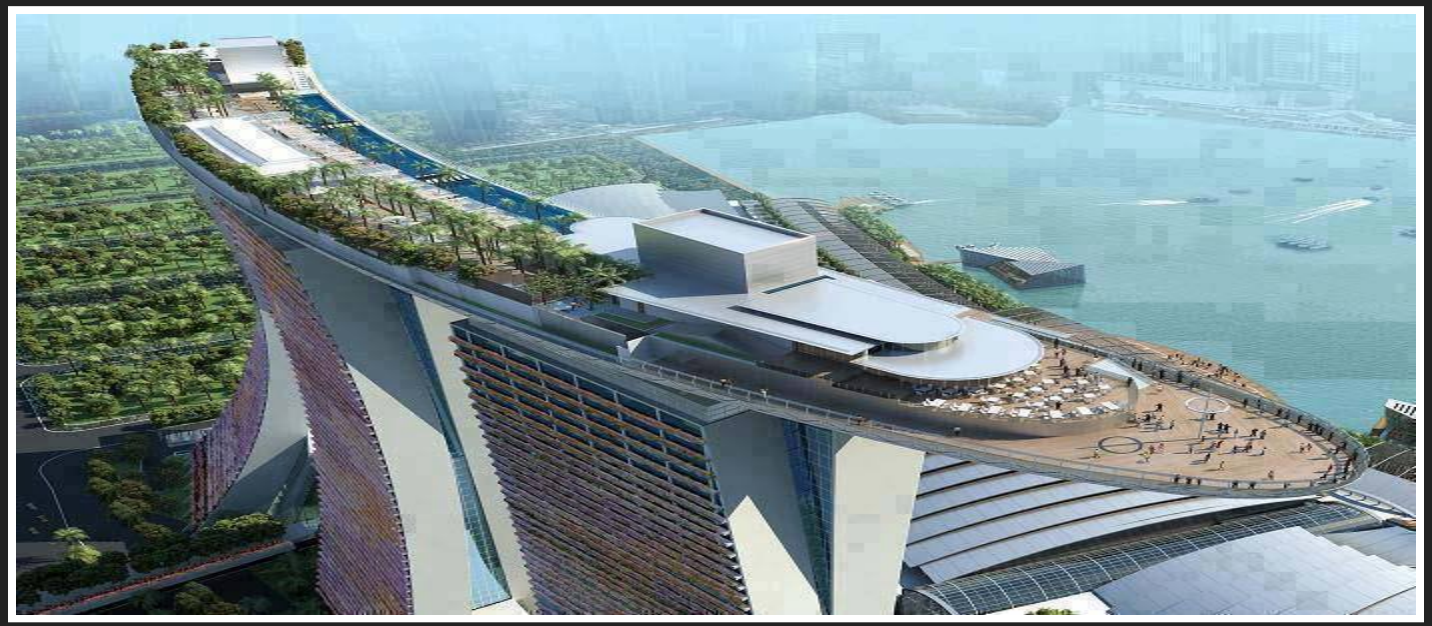
- *Laziness*
- *Impatience*
- *Hubris*

WHAT'S WRONG WITH THE 'ART'?

- I'll design it as I go along
- I don't need to write down my ideas
- It'll be done when I'm done

IF YOU'RE ENGINEERING IT

- You need to know WHAT to build
- You need to know HOW to build it
- You need to build it
- You need to DOCUMENT what you're doing
- You need to do it on time!



TOPICS COVERED IN THIS MODULE

- Requirements
  - Elicitation, specification, validation
- Design
  - General concepts, key issues
- Architecture
  - Structures, styles, patterns
- Testing
  - Techniques, processes
- Software Engineering Processes
  - For companies, teams, and individuals

ACTIVITIES EXPECTED FROM YOU

- Listen in class (your exam will be directly based on discussion in class)
- Complete your assignments (*every* week)
- Reflect on your assignment feedback
- Reading list (Read these *before* you come to class)

ASSIGNMENTS - DETAILS

- Account for 70% of your final grade (so do them)
- Programming assignments
  - Well-defined input and output
  - Submission via source-control (https://gitlab.scss.tcd.ie/)
- Your code will be tested against test inputs

MEASUREMENT, MEASUREMENT, MEASUREMENT

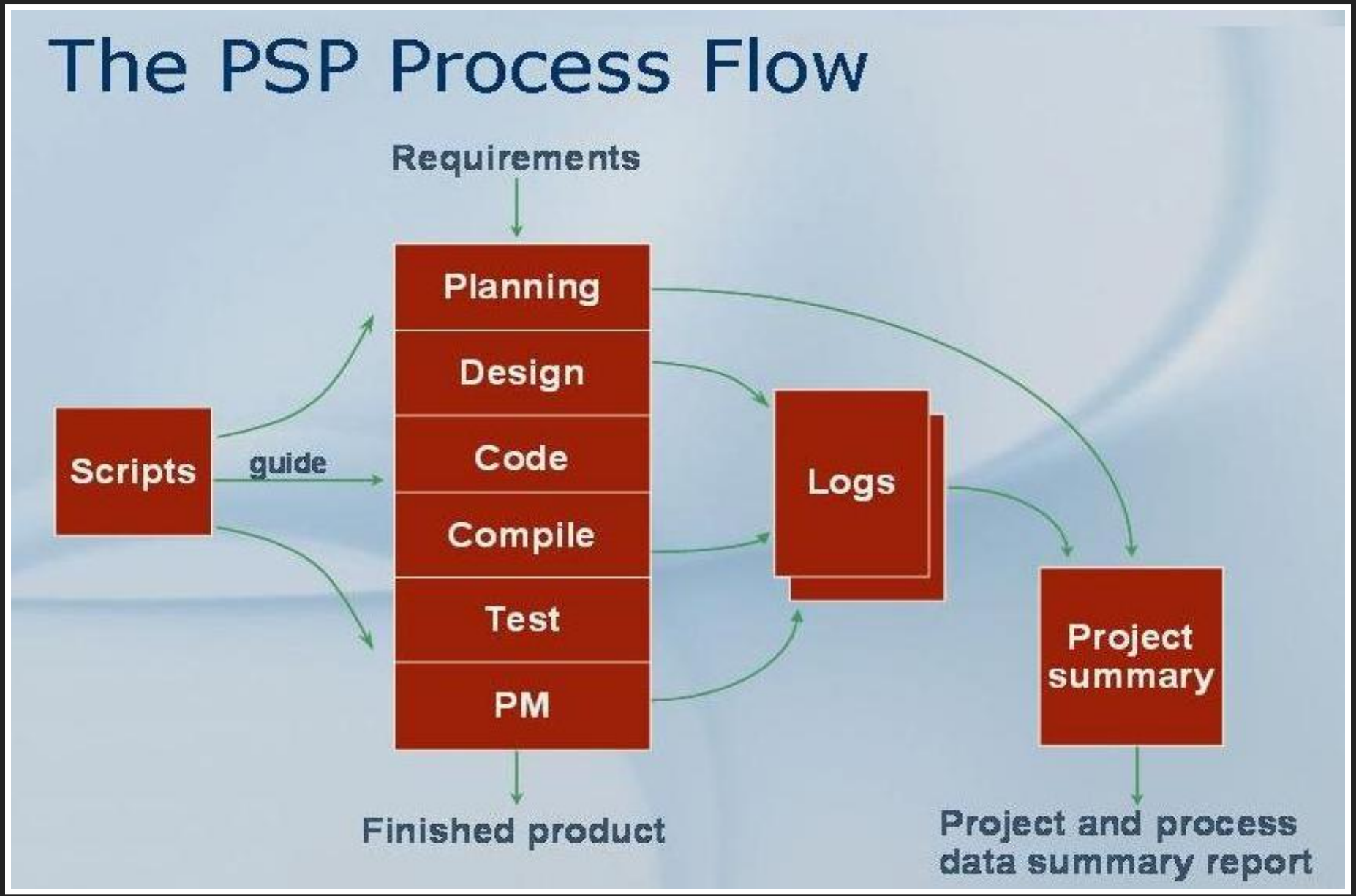
To measure is to know

If you cannot measure it, you cannot improve it

PERSONAL SOFTWARE PROCESS (PSP)

- Invented by Watts Humphrey at Software Engineering Institute
- Structured programming process intended to allow programmers to better understand and improve themselves.
- Main components:
  - Scripts
  - Tracking
  - Postmortem

PSP HAS MANY LEVELS, BUT THE MOST BASIC ONE LOOKS LIKE THIS



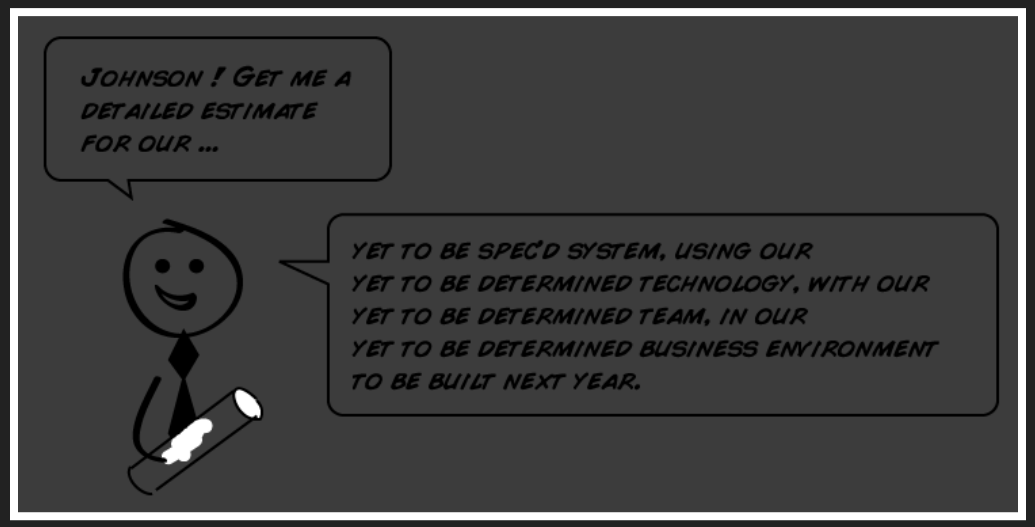
PERSONAL SOFTWARE PROCESS (BUT NOT REALLY)

- The full process is a little cumbersome. We will use a modified version
- Estimating / Planning
- Designing
- Code
- Test
- Postmortem (for the next assignment)

EVIDENCE-BASED SOFTWARE DEVELOPMENT

- PSP got one thing right: no evidence, no improvement
- So, for every phase of the assignment, record *estimated* and *actual* time
- Calculate *velocity* of your own process
- So, you get a personalized view of how *you* code

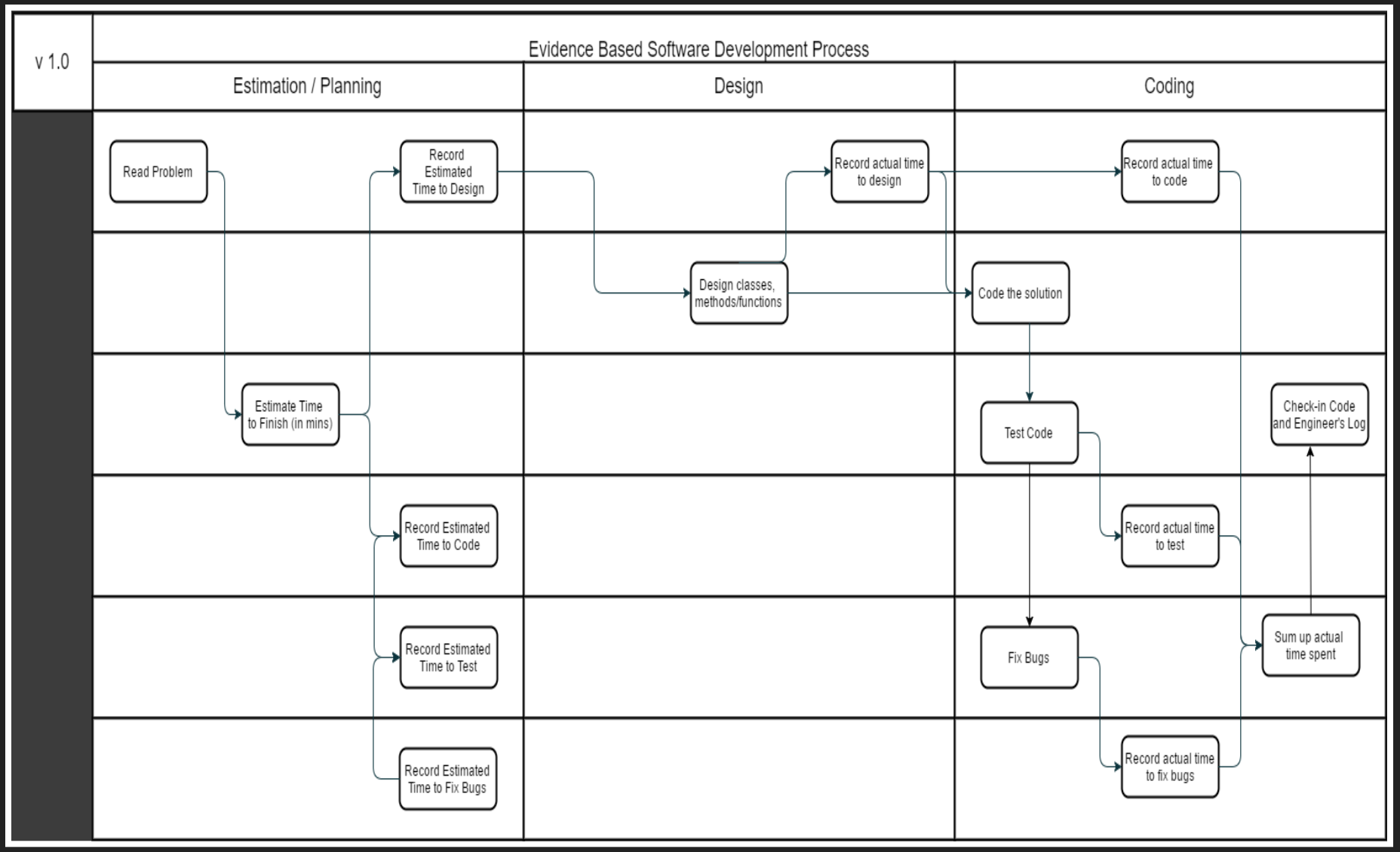
ESTIMATING - WHAT NORMALLY HAPPENS



ESTIMATING - WHAT YOU WILL DO

Evidence-based estimation!

YOUR PROCESS - 1



YOUR PROCESS (IN TEXT)

- Plan
- Design
- Code
- Test

PLANNING

YOUR PROCESS - 2

- Keep an Engineer's Log *throughout* your coding process
- Each assignment consists of a programming problem
- For each assignment, there are four tasks:
  - Design your solution
  - Code it
  - Test it as much as you can
  - Fix the bugs you found

YOUR PROCESS - 3

- For each task:
  - Estimate how much time (in minutes) it will take, *before* the task.
  - Record estimate in the Engineer's Log
  - Do the task
  - Record the actual time taken, in the Engineer's Log

YOUR PROCESS - 4

- Each one of you will have a personalized repository: https://gitlab.scss.tcd.ie/vivek.nallur/cs3012_[your-user-id]/
- For each assignment:
  - Create a directory for it (e.g., 1, 2, 3...)
  - Check-in your code
  - Check-in your Engineer's Log

YOUR PROCESS - 5

- Why are you doing this?
  - Creates a structured process for you
    - You can measure yourself
    - You can improve
  - Familiarity with basic tools of the industry (*i.e.*, version-control, testing)

GRADING OF ASSIGNMENTS - 1

- Submitted code will be checked against the specification given in the problem
- The ***Input*** and the ***Output*** specification will be checked strictly
- Every mis-match will be counted as a defect
- Your programming score will be:
  $Number\ of\ Test\ Cases - Number\ of\ Defects$

GRADING OF ASSIGNMENTS - 2

From your Engineer's Log:

$$Velocity_{task} = \frac{ActualTimeforTask}{EstimatedTimeforTask}$$

$$AverageVelocity = \frac{\sum Velocity_{task}}{NumberofTasks}$$

GRADING OF ASSIGNMENTS - 3

Your improvement over all the assignments:

- At the end of all assignments, calculate the trendline for *your personal* $AverageVelocity$
- The slope of your trendline should be negative.

HOW TO CHEAT?

- Google for solutions to assignment (your Engineer's Log will be messed up)
- Doctor your Engineer's Log (too easy to be caught out. If you're caught, you're automatically reported to the Module director)
- Doctor your Engineer's Log wisely + google for solutions (not worth your time)

ASSIGNMENT FOR NEXT WEEK

- Log on to https://gitlab.scss.tcd.ie/ with your scss-id
- Learn how to use Git [http://rogerdudler.github.io/git-guide/]
- Request access to the CS3012 repository
  - Navigate to https://gitlab.scss.tcd.ie/vivek.nallur/cs3012.git
  - Poke around the website till you find the "Request Access" button
  - If you don't request access, you cannot submit (this is usually very harmful to your grade)

THAT'S ALL FOLKS!

Any questions?