CSCI 2120U
Software System Development & Integration

Lecture 1:
Introduction

Rohollah Moosavi

Email: rohollah.moosavi@ontariotechu.ca

Outline

- Get to know each other.
- Review the Course Syllabus.
- An Introduction.

- Professor: Dr. Rohollah Moosavi
  - **E-mail:** rohollah.moosavi@ontariotechu.ca
  - **Room:** UAB 440
  - **Office hour:** (UA 2029)
    - Monday 5pm – 6pm
    - Wednesday 5pm – 6pm
- Teaching Assistants:

Jessica Jessica	jessica@uoit.net
Gavin Gosling	gavin.gosling@uoit.net
Ibrahim Mushtaq	ibrahim.mushtaq@uoit.net
Dikachi Kalu	onyedikachi.kalu@uoit.net
Sumeet Dhillon	sumeet.dhillon@ontariotechu.net

- Lectures:
  Mon. 3:40pm – 5 pm (UA 1120)
  Wed. 3:40pm – 5 pm (UA 1120)

- Laboratories:

Jessica jessica	M	9:40am – 11am
Gavin Gosling	T	2:10pm – 3:30pm
Ibrahim Mushtaq	W	2:10pm – 3:30pm
Dikachi Kalu	T	2:10pm – 3:30pm
	T	12:40pm – 2pm
Sumeet Dhillon	W	9:40am – 11am

Note:

- Labs officially start on January. 13th

About You

1. Your name?
2. Your favorite Course?
3. Which programming language do you know?
4. How many years experience do you have in programming? And which programming languages?
5. How much are you interested in programming?
6. Your Java proficiency?
7. Your OOP proficiency?
8. What future career you’d like to pursue?

About Me

- **Bachelor** in Computer Engineering.
- **Master** in Computer Engineering.
- **Ph.D.** in Computer Science – Computer Vision

- Previous:
  - Azad University, Tehran, Iran (15 years)
  - Limkokwing University, Malaysia (1 year)
  - Senior Software Development Engineer in Canada, Malaysia & Iran (10 years)
- Ontario Tech University

Expectations from you

1. **Attendance**
   - Come to class on time and prepared
   - If you miss a lecture, it is up to you to **make up** for it
2. **Regular work**
   - You **can’t learn** only **by watching**
   - You need to put in regular practice
3. **Time management**
   - A big pitfall, even for high performing students, is to leave work until too close to its **due date**
   - Cramming is a roadblock to learning

Evaluation

Component	Due Date	Weight
Labs	Every week	10%
Quizzes	Surprise/Announced	5% ~ 10%
Assignment	March 5, 2020 (due by 11:59pm, Release: 14 February)	10%
Test #1	February 26, 2020 (in class)	10%
Test #2	March 18, 2020 (in class)	15%
Group project	March 23, 2020	25%
Final exam	TBA, April 2020	25%

Late Submissions

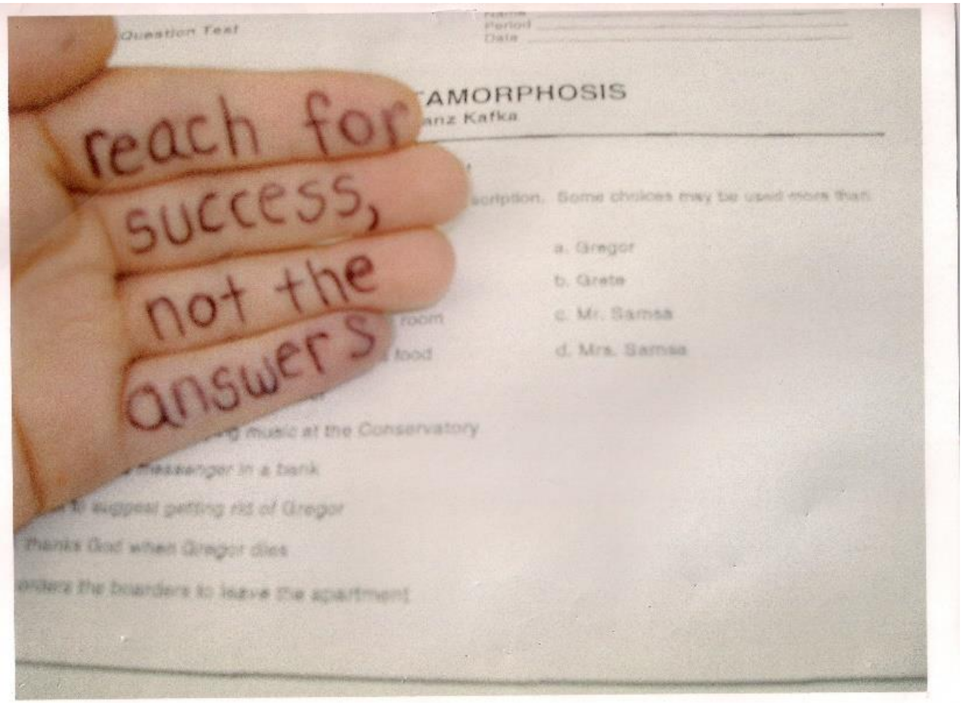
- Any student who misses an examination without a valid medical reason and documentation will receive zero for that examination.
- Those who submit medical documentation will either be given a **makeup exam** or will have the **weight** of the examination redistributed.
- For assignments and lab assignments, **late submissions will not be accepted**.

Academic Integrity

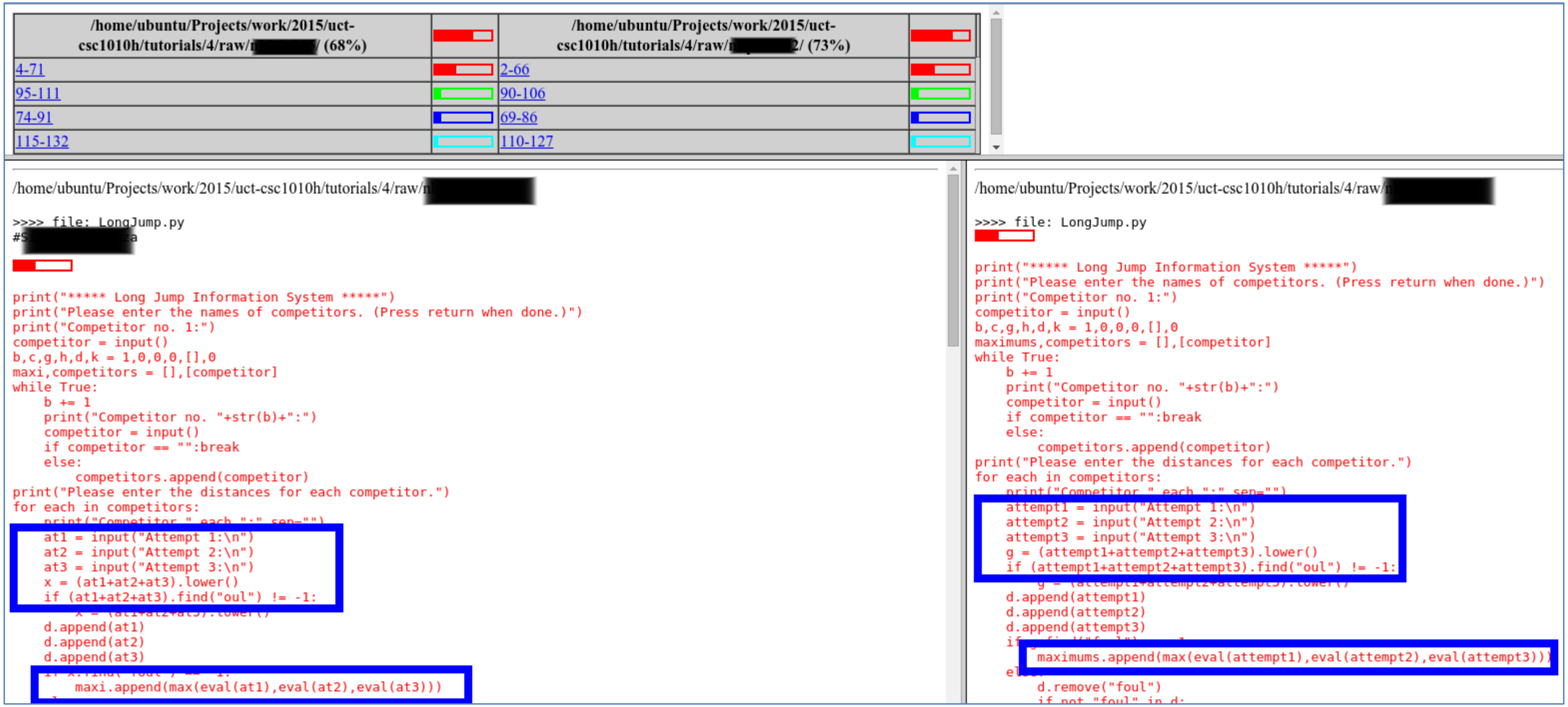
- All assignments are group-based assignments.
- You may discuss assignment questions with your fellow students but please do not exchange your code or use another group’s and/or student’s work.
- Please read the academic integrity section of the Ontario Tech University webpage.

Academic Integrity Test

- Everybody MUST do this test, and if you get less than 100, you can repeat it as many times as you want.



Plagiarism Checker



/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/4/raw/n[redacted]/ (68%)	🟥	/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/4/raw/n[redacted]2/ (73%)	🟥
4-71	🟥	2-66	🟥
95-111	🟩	90-106	🟩
74-91	🟦	69-86	🟦
115-132	▭	110-127	▭

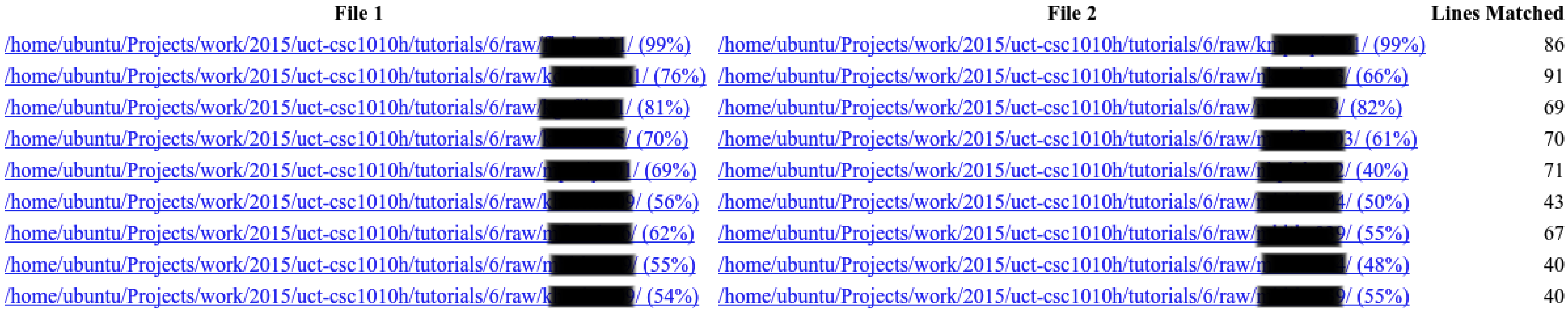
/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/4/raw/n[redacted]

```
>>>> file: LongJump.py
#S[redacted]za

print("***** Long Jump Information System *****")
print("Please enter the names of competitors. (Press return when done.)")
print("Competitor no. 1:")
competitor = input()
b,c,g,h,d,k = 1,0,0,0,[],0
maxi,competitors = [],[competitor]
while True:
    b += 1
    print("Competitor no. "+str(b)+":")
    competitor = input()
    if competitor == "":break
    else:
        competitors.append(competitor)
print("Please enter the distances for each competitor.")
for each in competitors:
    print("Competitor ",each,":",sep="")
    at1 = input("Attempt 1:\n")
    at2 = input("Attempt 2:\n")
    at3 = input("Attempt 3:\n")
    x = (at1+at2+at3).lower()
    if (at1+at2+at3).find("oul") != -1:
        x = (at1+at2+at3).lower()
    d.append(at1)
    d.append(at2)
    d.append(at3)
    if x.find("foul") == -1:
        maxi.append(max(eval(at1),eval(at2),eval(at3)))
```

/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/4/raw/n[redacted]

```
>>>> file: LongJump.py

print("***** Long Jump Information System *****")
print("Please enter the names of competitors. (Press return when done.)")
print("Competitor no. 1:")
competitor = input()
b,c,g,h,d,k = 1,0,0,0,[],0
maximums,competitors = [],[competitor]
while True:
    b += 1
    print("Competitor no. "+str(b)+":")
    competitor = input()
    if competitor == "":break
    else:
        competitors.append(competitor)
print("Please enter the distances for each competitor.")
for each in competitors:
    print("Competitor ",each,":",sep="")
    attempt1 = input("Attempt 1:\n")
    attempt2 = input("Attempt 2:\n")
    attempt3 = input("Attempt 3:\n")
    g = (attempt1+attempt2+attempt3).lower()
    if (attempt1+attempt2+attempt3).find("oul") != -1:
        g = (attempt1+attempt2+attempt3).lower()
    d.append(attempt1)
    d.append(attempt2)
    d.append(attempt3)
    if g.find("foul") == -1:
        maximums.append(max(eval(attempt1),eval(attempt2),eval(attempt3)))
    else:
        d.remove("foul")
        if not "foul" in d:
```

Plagiarism Checker



File 1	File 2	Lines Matched
/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (99%)	/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (99%)	86
/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (76%)	/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (66%)	91
/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (81%)	/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (82%)	69
/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (70%)	/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (61%)	70
/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (69%)	/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (40%)	71
/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (56%)	/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (50%)	43
/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (62%)	/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (55%)	67
/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (55%)	/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (48%)	40
/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (54%)	/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/6/raw/[redacted]/ (55%)	40

Course Outline

1. Software engineering
2. Version control
3. Build tools
4. Code and design best practices
5. Graphical user interfaces
6. File input and output
7. Socket input and output
8. Multi-threaded programming
9. Software libraries

Software engineering

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled.
- Software engineering is concerned with theories, methods and tools for professional software development.

Software Costs

- Software costs often **dominate** computer system costs. The costs of software on a PC are often **greater than** the hardware cost.
- Software costs more to maintain than it does to develop.
- For systems with a long life, maintenance costs may be **several times** development costs.
- Software engineering is concerned with cost-effective software development.

Software project failure

1- Increasing system complexity

- As new software engineering techniques help us to build larger, more complex systems, as the demands change.
- Systems have to be built and delivered more quickly; larger, even more complex systems are required;
- Systems have to have new capabilities that were previously thought to be impossible.

Software project failure

2- Failure to use software engineering methods

- It is **fairly easy** to write computer programs without using software engineering methods and techniques.
- Many companies have drifted into software development as their products and services have evolved.
- They do not use software engineering methods in their everyday work. Consequently, their software is often more expensive and less reliable than it should be.

Professional Software Development

- History of software engineering:
  - In fact it is proposed in 1968 and through 1970 and 1980, several software engineering techniques were developed, such as structured programming and object-oriented development.

Professional Software Development

- Software engineering is intended to **support** professional software development **rather than** individual programming.

- It includes techniques that support program specification, design, and evolution, none of which usually relevant for personal software development.

Professional Software Development

What is **software?**

- Software is not just the programs themselves, but also all associated **documentation**, **libraries**, support **websites**, and **configuration data** that are needed to make these programs useful.
- Software products may be developed for a particular customer or may be developed for a general market.

What are the **attributes** of **good software**?

- Good software should **deliver the required** functionality and performance to the user, and should be maintainable, dependable and usable.

Professional Software Development

- What is **software engineering**?
  - Software engineering is a discipline that is concerned with all aspects of software production.

- What are the fundamental software engineering **activities**?
  - Software specification, software development, software validation and software evolution.

Professional Software Development

What is the **difference** between **software engineering** and **computer science**?

- Computer science focuses on **theory** and **fundamentals.**
- Software engineering is concerned with the **practicalities** of **developing** and **delivering useful software**.

What is the **difference** between **software engineering** and **system engineering**?

- System engineering is concerned with **all aspects** of **computer-based systems development** including **hardware**, **software** and **process engineering**.
- Software engineering is part of this more general **process**.

Professional Software Development

What are the **costs** of software engineering?

- Roughly 60% of software costs are development costs, 40% are testing costs.
- For custom software, evolution costs often exceed development costs.

Professional Software Development

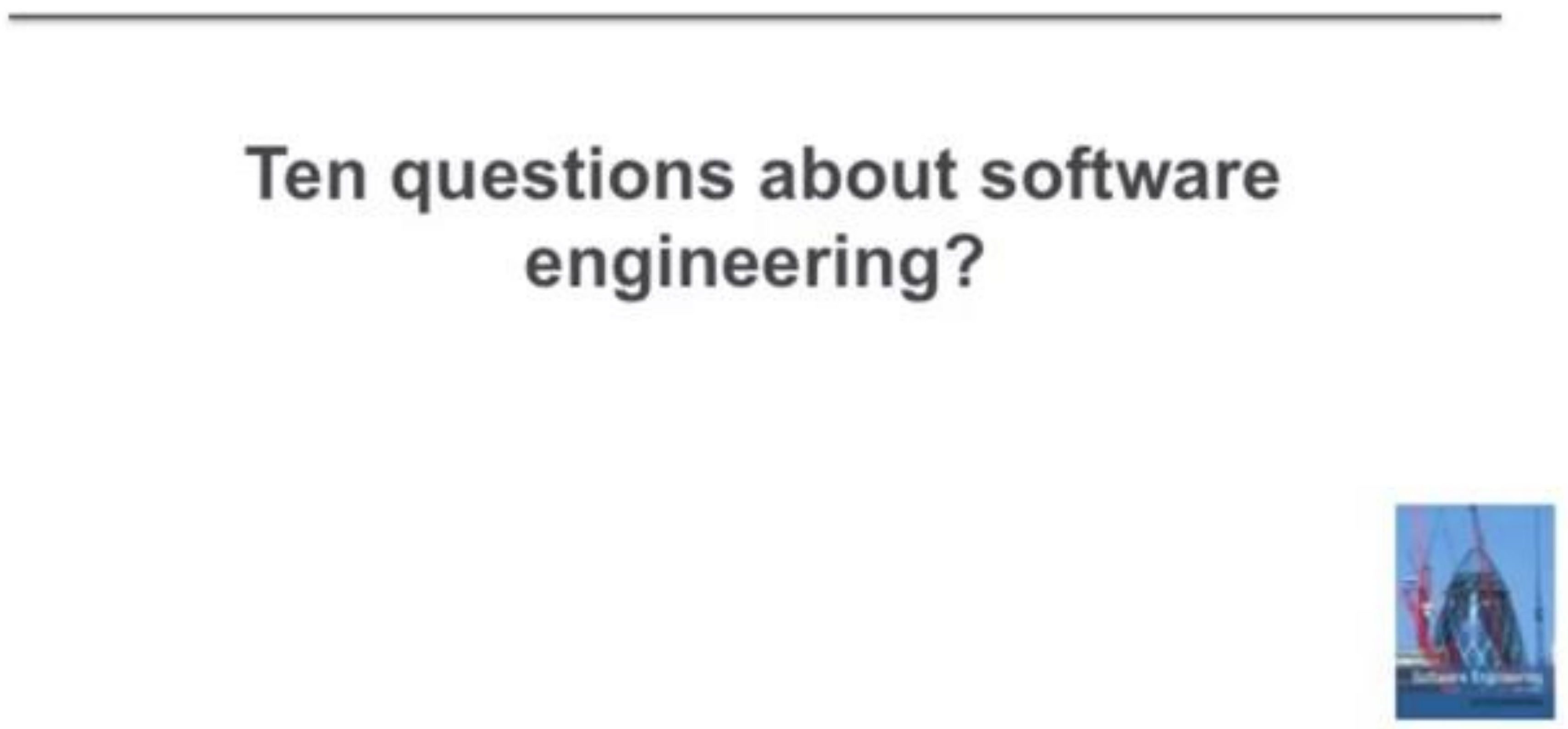
What are the **best software engineering techniques** and **methods**?

- While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system.
- For example, games should always be developed using a **series of prototypes** whereas safety critical control systems require a **complete** and **analyzable specification** to be developed.
- We **can’t**, therefore, say that one method is better than another.

What differences has the web made to software engineering?

- The web has **led to** the availability of software services and the possibility of developing **highly** distributed service-based systems.
- Web-based systems development has **led to** important advances in programming languages and software reuse.

Ten Questions about Software Engineering



Ten questions about software engineering?

Essential Attributes of Good Software

Maintainability

- Software should be written in such a way so that it can evolve to meet the changing needs of customers.
- This is a **critical attribute**, because software change is an inevitable requirement of a changing business environment.

Dependability and security

- Software dependability includes a range of characteristics such as reliability, security and safety.

Essential Attributes of Good Software

Efficiency

- Software should not make wasteful use of system resources such as memory and processor cycles.
- Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.

Acceptability

- Software must be acceptable to the type of users for which it is designed.
- This means that it must be understandable, usable and compatible with other systems that they use.

Software Engineering

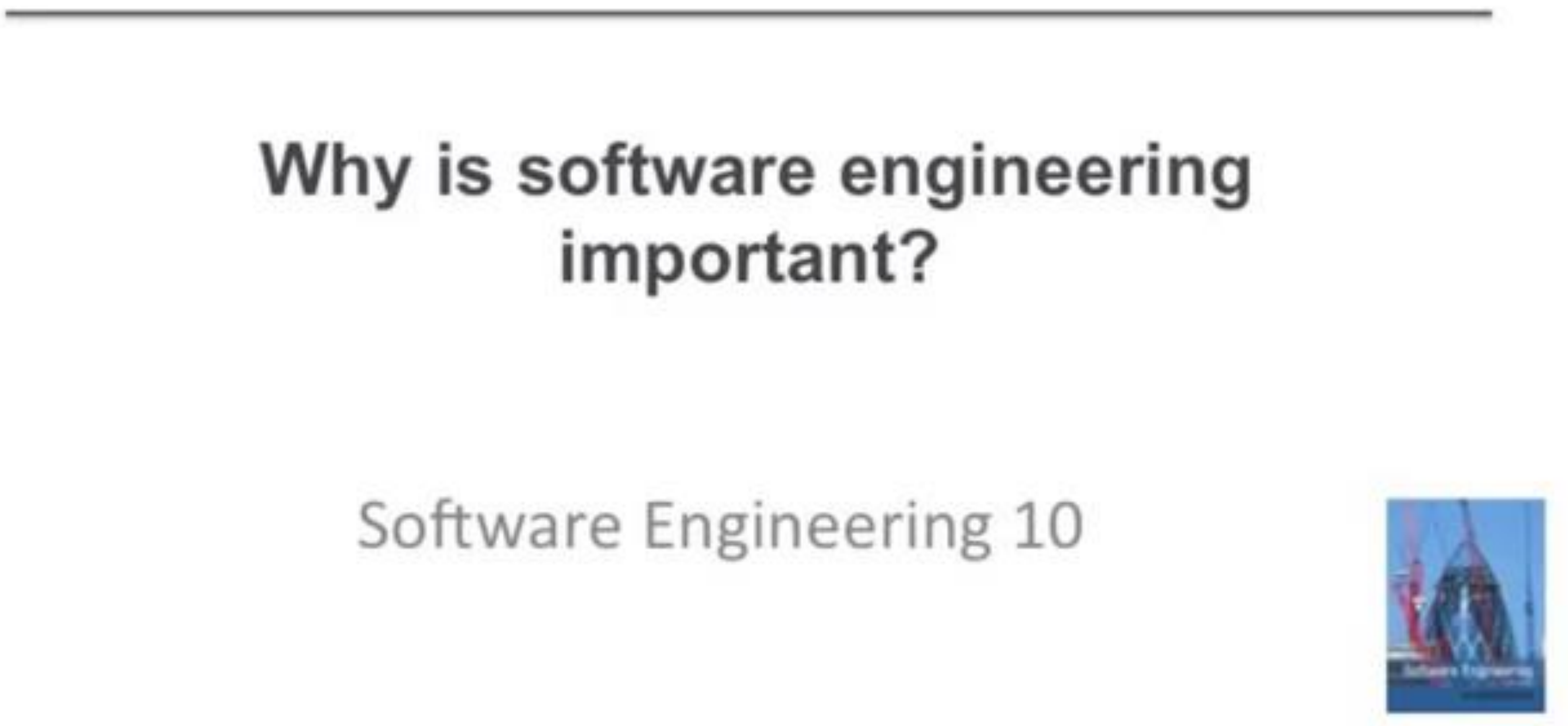
- **Software engineering** is an **engineering discipline** that is concerned with **all aspects** of software production from the early stages of system specification through to maintaining the system after it has gone into use.

- **All aspects** of software production means, not just technical process of development. Also project management and the development of tools, methods etc. to support software production.

Importance of Software Engineering

- It is usually **cheaper**, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project.

Why is Software Engineering Important?



Why is software engineering important?

Software Engineering 10

Software Process Activities

- **Software specification**
  - Where **customers** and **engineers** **define** the software that is to be produced and the **constraints on its operation**.
- **Software development**
  - Where the software is **designed** and **programmed**.
- **Software validation**
  - Where the software is **checked** to ensure that it is what the customer requires.
- **Software evolution**
  - Where the software is **modified** to reflect changing customer and market requirements.

Software Engineering Diversity

- There are many different types of software system, there is no universal set of software techniques that is applicable to all of these.
- The software engineering methods and tools used **depend on**:
  - The **type of application** being developed,
  - The **requirements** of the **customer** and
  - The **background** of the **development** team.

Application Types

- **Stand-alone applications**
  - These are application systems run on a local computer, such as a PC.
  - They include all necessary functionality and do not need to be connected to a network.
- **Interactive transaction-based applications**
  - Applications that execute on a remote computer and are accessed by users from their own PCs or terminals.
  - These include web applications such as e-commerce applications.
- **Embedded control systems**
  - These are software control systems that control and manage hardware devices.

Application Types

- **Batch processing systems**
  - These are business systems that are designed to process data in large batches.
  - They process large numbers of individual inputs to create corresponding outputs.

- **Entertainment systems**
  - These are systems that are primarily for personal use and which are intended to entertain the user. Example: games

- **Systems for modelling and simulation**
  - These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.
  - These are often computationally intensive and require high-performance parallel systems for execution.

Application Types

- **Data collection systems**
  - These are systems that:
    - Collect data from their environment using a set of **sensors** and
    - Send that data to other systems for processing.
  - The software may have to interact with sensors and often is installed in a hostile environment such as inside an engine or in a remote location.

Software Engineering Fundamentals

- Some fundamental principles apply to all types of software system, irrespective of the development techniques used:
  - Systems should be developed using a managed and understood development process. Of course, different processes are used for different types of software.
  - Dependability and performance are important for all types of system.
  - Understanding and managing the software specification and requirements (what the software should do) are important.
  - Where appropriate, you should reuse software that has already been developed rather than write new software.

Internet Software Engineering

- The Web is now a platform for running applications.
- Organizations are increasingly developing web-based systems rather than local systems.

- **Web services** allow application functionality to be accessed over the web.
- **Cloud computing** is an approach to the provision of computer services where applications run remotely on the ‘cloud’.
    - Users do not buy software, buy pay according to use.

Web-based Software Engineering

- Web-based systems are complex distributed systems
  - But the fundamental principles of software engineering discussed previously are **as applicable to them** as they are to any other types of system.

- The fundamental ideas of software engineering apply to web-based software in the same way that they apply to other types of software system.

Web-based Software Engineering

- Software reuse
  - Software reuse is the dominant approach for constructing web-based systems.
  - When building these systems, you think about how you can **assemble** them from pre-existing software components and systems.

- Incremental and agile development
  - Web-based systems should be developed and delivered incrementally.
  - It is now generally recognized that it is **impractical** to specify all the requirements for such systems in advance.

Web-based Software Engineering

- Service-oriented systems
  - Software may be implemented using service-oriented software engineering
  - Where the software components are stand-alone web services.

- Rich interfaces
  - Interface development technologies such as **AJAX** and **HTML** have emerged that support the creation of rich interfaces within a web browser.

Summary

- Software engineering is an engineering discipline that is concerned with all aspects of software production.

- Essential software product attributes are maintainability, dependability and security, efficiency and acceptability.

- The high-level activities of specification, development, validation and evolution are part of all software processes.

- The fundamental notions of software engineering are universally applicable to all types of system development.

Summary

- There are many different types of system and each requires appropriate software engineering tools and techniques for their development.

- The fundamental ideas of software engineering are applicable to all types of software system.