Software Engineering-I

Bridging the gap between Computer Programming and Software Engineering

Learning Objectives

The learning objectives are to

- Understand the differences between Software Engineering and Computer Engineering
- To identify the engineering needs of software-driven companies

I

Software Engineering happens in organizational context

Programming is individual activity

Implement sorting algorithms

Creating a simple web page and displaying greeting message

A mobile app to calculate Body Mass Index (BMI) given height and weight.

Software Engineering is an organizational activity

To develop an
E-Commerce Platform

Trading platform

Movie streaming
platform

Programming as Academic vs Organizational Activity

To learn a specific skill

To apply an acquired skill.

To implement smaller scale applications

To implement large scale applications

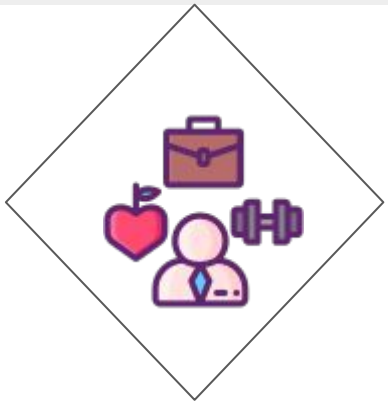
Operational issues are not focussed

Operational issues are central

Consequences of SW Engineering as an organizational activity

Need of effective communication tools

Proper interface to divide work units among developers.



Tools to enhance developers productivity

Tools to monitor and respond operational issues.

II

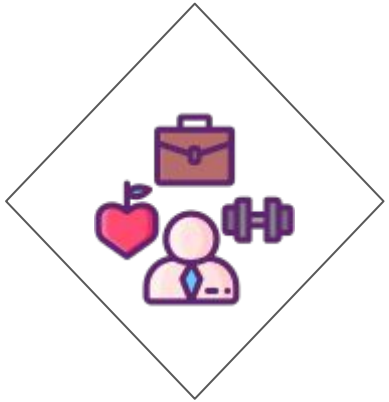
Software Engineering is programming integrated with time.

Long live the software

Programming assignments are confined to a limited time frame

Software such as YouTube and Linux kernel endure indefinitely

Rigorous testing



Continuous adaptation to emerging technologies

Operational efficiency

III

Software Engineering is multi-version and multi-person Programming

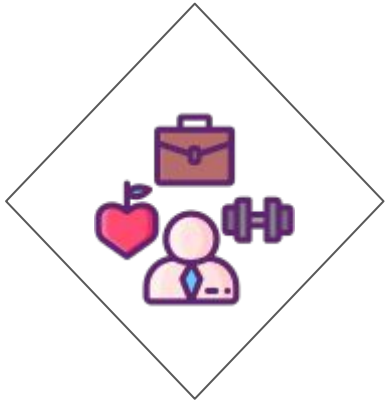
Complexity of Modern Software

	Developers	Code Size	Language	Version
Gmail	1000+	10M+	Java, JavaScript	20+
YouTube	1000+	10M+	Java, Python, Go	40+
Map	1000+	15M+	Java, C++	30+
Drive	500+	8M+	Java, Python,Go	50+

Consequences

Software Engineering is multi-version and multi-person Programming

Effective tools for collaboration



Product management techniques

Version Management

IV

Software Engineering is writing clean code and not clever code

$$F_0 = 0$$
$$F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2}$$

Clean Code

python Copy code

```python
def fibonacci(n):
    if n <= 0:
        raise ValueError("Input must be a positive integer")
    elif n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        prev, curr = 0, 1
        for _ in range(n - 2):
            prev, curr = curr, prev + curr
        return curr
```

Binet's Formula

Let $F_n = F_{n-1} + F_{n-2}$, with $F_0 = 0$, $F_1 = 1$, be the Fibonacci sequence, then

$$F_n = \frac{\varphi^n - (-\varphi)^{-n}}{\sqrt{5}} = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

Where $\varphi = \dfrac{1+\sqrt{5}}{2} = 1.6180339875\ldots$ is the golden ratio

Clever Code

python

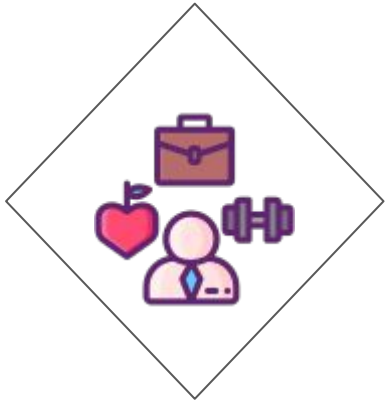
Copy code

```python
def fibonacci(n):
    sqrt5 = 5 ** 0.5
    phi = (1 + sqrt5) / 2
    return round((phi ** n - (-phi) ** -n) / sqrt5)
```

Consequences

Clever code writing might be allowed in academic setting.	Clean code is typically the norm in industry.

Rigorous **code review** is required to ensure code quality



Code needs to be written for human readability

Organization specific guidelines need to developed

Hyrum's Law

With a sufficient number of users of an API, it does not matter what you promise in the contract: all observable behaviours of your system will be depended on by somebody.

Observable Behaviour vs API

Hyrum's Law

Output on Google Colaboratory:

```
def display(myset):
    for i in myset:
        print(i)
myset={"Aditya","Bal","Chavan","Devendra","Eknath"}
display(myset)
```

```
Chavan
Eknath
Devendra
Aditya
Bal
```

Output on Pycharm:

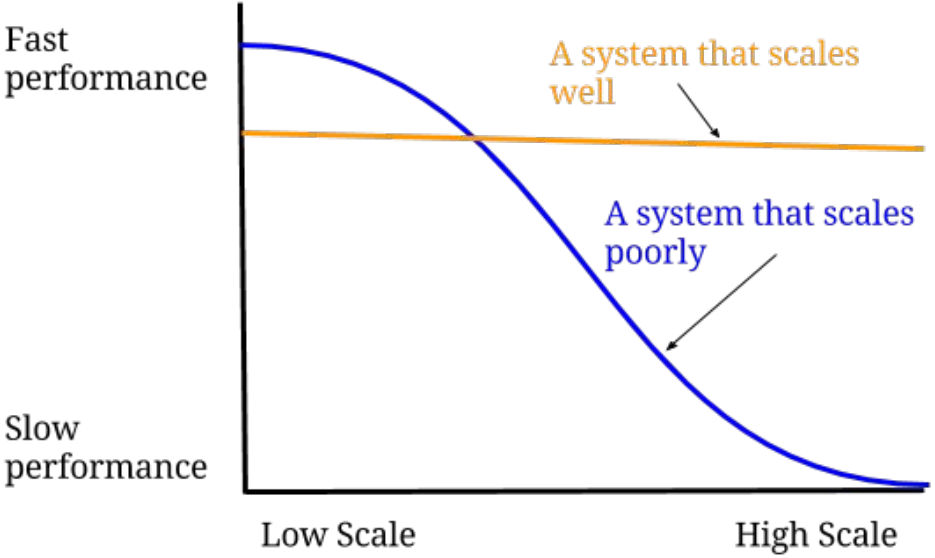
```
def display(myset):
    for i in myset:
        print(i)
myset = {"Aditya", "Bal", "Chavan", "Devendra", "Eknath"}
display(myset)
```

```
"C:\Users\Shree Ganesh\PycharmProjects\hello\venv\Scripts\python.exe" "C:/Users
Bal
Chavan
Aditya
Devendra
Eknath
```

V

Addressing operational issues is one of the major concerns of Software Engineering.

Software Engineering is about designing scalable applications.



the ability of a computing process to be used to meet the increasing demands.

VI

Addressing tradeoffs is a central activity while designing software.

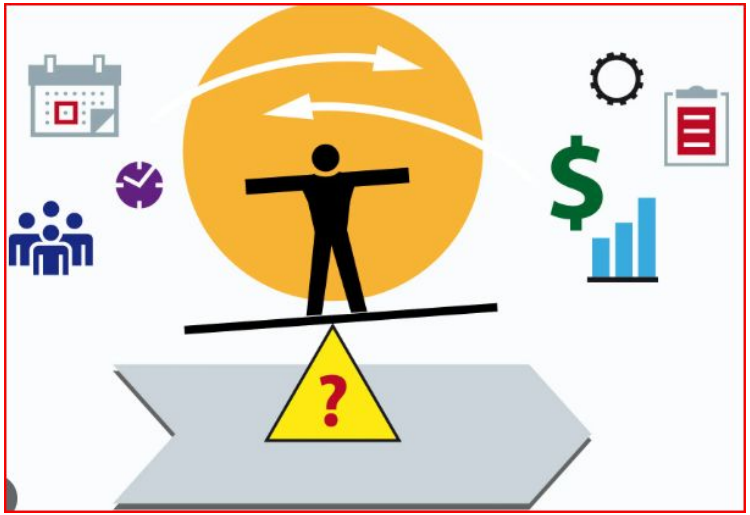
Addressing tradeoffs is a central activity while designing software

Security Vs Usability

Strength of password

Captcha

Cost of operation vs number of computing resources



Conclusion

Software Engineering differs from computer programming in terms of

1. Individuality
2. Evolvability
3. Code quality
4. Addressing operational issues

Modern software needs tools for

1. Effective Collaboration
2. Version Management
3. Assure code quality
4. Monitoring and feedback mechanism for operational issues

Quiz time

Which aspect of software engineering is *<u>completely missing</u>* when software is developed in an academic setting

1. Acquiring new technical skills
2. Monitoring the performance of developed applications
3. To demonstrate the working of programming abstractions
4. To interact with teachers and supervisors.

Quiz time

What are the three significant needs of modern software development when it takes place in organizational context

1. Tools for version management
2. Tools for monitoring operating environment
3. An Interactive Development Environment
4. Tools to integrate development and operation issues

Quiz time

Which of the TWO dominant concerns that need to be handled in organizational context as opposed to academic setting

1. Scalability
2. Tradeoff
3. Acquiring new skills
4. Meeting project deadlines

Quiz time

Hyrum's law refers to which of the two developmental concerns

1. API and Observable behaviour
2. API and Portability
3. API and Tradeoff
4. Observable behaviour and Tradeoff.