Unit 4 Seminar

Title: Estimating Tools and Risk Assessment

Activity 1

1. **Review the NIST Privacy tools. How do these fit with the risk assessment methods and tools described in last week's Lecturecast?**

The National Institute of Standards and Technology (NIST) Privacy Framework is an indispensable tool for organizations to identify and manage privacy risks while creating innovative products and services. It comprises privacy risk management and assessment, critically analysing potential future events that could impact individuals, assets, processes, or the environment. NIST provides resources like typical profiles and crosswalks to map regulatory requirements and standards to the Privacy Framework outcomes. (NIST, 2020).

In light of last week's lecture cast, the Risk Management Process (RMP) plays a crucial role in the success of software engineering projects, and software engineering project managers (SEPMs) must take responsibility for it. Information risk is a significant business element that affects all aspects of a company, and implementing a new risk management system as a program may be necessary. Proper governance, buy-in from senior management, and involvement from heads of all central departments are essential to ensure a successful RMP (My-course, 2021).

SEPMs must follow the PDCA mandate, including the plan, do, check, and act phases. The planning phase involves creating a plan, adopting new procedures, roles, and responsibilities, and reviewing the process. The do phase requires gathering and reviewing feedback, while the act phase modifies procedures, roles, and responsibilities.

Risk identification is a core aspect of the SEPM role, and the project manager must collect and analyse risks from various perspectives. Frameworks such as Open FAIR, OCTAVE, and NIST can help with this process.

Risk analysis involves assigning numerical or qualitative assessment values to risks. Qualitative assessments involve asking staff to evaluate risks as minor, medium, or significant, while quantitative assessments use statistical or historical data to assess and assign a weight to a risk.

The mitigation phase involves classifying risks into four categories: eliminate, tolerate, reduce, or transfer. Mitigation mechanisms can include using software products for editing, storing documents on secure servers, or using version control systems to audit access and quickly revert to old versions.

The risk management process (RMP) is essential to ensuring the success of software engineering projects. By strictly adhering to the PDCA mandate, SEPMs can confidently guarantee their projects' success and maintain a robust reputation that perfectly aligns with the NIST Privacy Framework tools.

2. **Create a Python program that implements one of the estimation methods covered in the Lecturecast.**

One of the methods covered in the lecture cast is functional points. Functional Point Analysis (FPA) is a software engineering tool that provides dimensionless function points for estimating work, time, and materials needed for software development. It aids in project management, benchmarking, and cost-benefit analysis (GeeksforGeeks, 2019).

Blueoptima (2023) notes that the function points are widely used in software engineering, allowing engineers to accurately measure project size, identify optimisation areas, and analyse development performance. They offer a more holistic

view than lines of code metrics, considering key factors like data elements, files, user inputs, and outputs. However, they have drawbacks, like difficulty assigning accurate complexity values.

Below is an example of a Python program that includes FPA calculations based on the ideas of Wakoli (2021), assuming that each project result (delivered, ongoing, failed) corresponds to a specific functionality point. The program assumes that each project outcome (delivered, ongoing, or failed) is associated with a specific functionality point. These points are then used to analyse the team's performance:

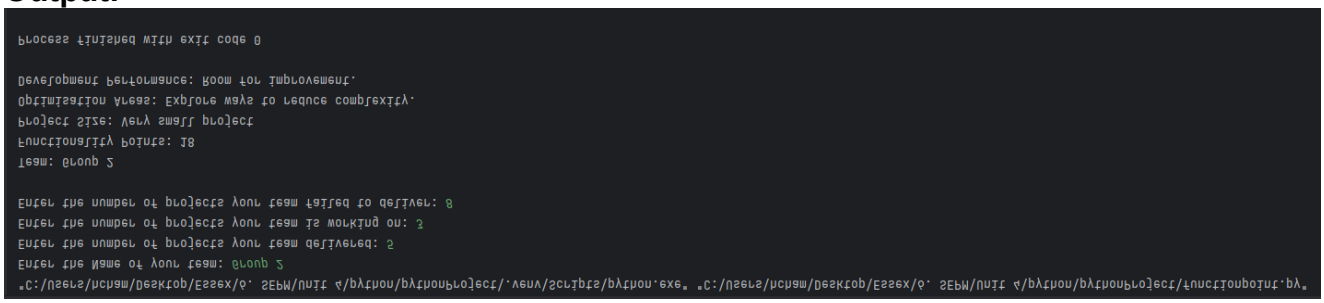
GitHub repository:

https://github.com/hchamane/hchamane.github.io/compare/main...Functional_Point_Analysis

```
"""Enhanced Software Project Analysis Tool"""

# Constants for FPA
DELIVERED_PROJECT = 3
ONGOING_PROJECT = 1
FAILED_PROJECT = 0

def calculate_functionality_points(delivered, ongoing, failed):
    """
    Calculates the total functionality points based on project results.
    """
    return (delivered * DELIVERED_PROJECT) + (ongoing * ONGOING_PROJECT) +
(failed * FAILED_PROJECT)

def analyse_project_size(points):
    """
    Determines project size based on functionality points.
    """
    if points >= 50:
        return "Large project"
    elif points >= 40:
        return "Medium project"
    elif points >= 30:
        return "Small project"
    else:
        return "Very small project"

def identify_optimisation_areas(points):
    """
    Suggests areas for optimisation based on functionality points.
    """
    if points >= 50:
```

```
        return "Consider optimising resource allocation and scalability."
    elif points >= 40:
        return "Focus on improving efficiency and code quality."
    elif points >= 30:
        return "Look for opportunities to streamline processes."
    else:
        return "Explore ways to reduce complexity."

def analyse_development_performance(points):
    """
    Evaluates development performance based on functionality points.
    """
    if points >= 50:
        return "Excellent performance!"
    elif points >= 40:
        return "Good performance."
    elif points >= 30:
        return "Moderate performance."
    else:
        return "Room for improvement."

def main():
    """
    The main function to validate and calculate the team's performance
    """
    teamname = input("Enter the Name of your team: ")
    delivered = int(input("Enter the number of projects your team
delivered: "))
    ongoing = int(input("Enter the number of projects your team is working
on: "))
    failed = int(input("Enter the number of projects your team failed to
deliver: "))

    # Validate input (assuming a maximum of 40 matches)
    if delivered + ongoing + failed > 20:
        print("Error: You entered too many matches. Please enter valid
numbers.")
        return

    # Calculate total functionality points
    totalpoints = calculate_functionality_points(delivered, ongoing,
failed)

    # Output results
    print(f"\nTeam: {teamname}")
    print(f"Functionality Points: {totalpoints}")
    print(f"Project Size: {analyse_project_size(totalpoints)}")
    print(f"Optimisation Areas:
{identify_optimisation_areas(totalpoints)}")
    print(f"Development Performance:
{analyse_development_performance(totalpoints)}")


if __name__ == "__main__":
    main()
```

Output:



```
"C:\Users\hcham\Desktop\Essex\6. SEPM\Unit 4\python\pythonProject\.venv\Scripts\python.exe"
Enter the Name of your team: Group 2
Enter the number of projects your team delivered: 5
Enter the number of projects your team is working on: 3
Enter the number of projects your team failed to deliver: 8

Team: Group 2
Functionality Points: 18
Project Size: Very small project
Optimisation Areas: Explore ways to reduce complexity.
Development Performance: Room for improvement.

Process finished with exit code 0
```

3. **Based on the requirements you have gathered for your assignment, create an estimate of the total effort and time to complete the planned demonstration of your system.**

3.1. Below are the requirements I have gathered from the transcript:

3.1.1. Hardware Requirements:

- **Portable and lightweight**: weight of around 2 kg, including the computer, batteries, screen, and peripherals.
- **Central Processing Unit (CPU):** Motorola 68k series CPU for power and future-proofing.
- **Memory:** 512Kb of RAM, a high capacity for the time.
- **Storage:** Built-in solid-state storage.
- **Screen:** Built-in low-power screen.
- **Expansion Slots:** To accommodate future upgrades and peripherals.
- **Input/Output (I/O) Ports:**
    - Serial ports for networking and communication (potentially supporting RS 422 and RS 485 standards).
    - Keyboard connector.
    - Joystick port for game emulation.
    - Centronics printer port.
    - SCSI port.

3.1.2. Software Requirements:

- **Operating System (OS):** Custom multi-tasking, Unix-like OS under development by Syn Computing.
- **Programming Language:** HyperBasic (HB), a structured, modular superset of BASIC, designed in-house to replace TeleBasic.
- **Business Suite Application:** A third-party supplier is developing a business suite that will be bundled with the system. This suite should include basic functionalities like word processing, spreadsheet, database, and graphics.
- **Backward Compatibility:** The ability to run existing Syn Computing software through emulation, potentially using a TeleBasic converter to allow TB programs to run on the new machine without modification.

3.1.3. Additional Requirements:

- **Cost-effective design:** A target cost price of £250 per unit.
- **User-friendliness:** An external keyboard for easier typing and the potential for different keyboard layouts in various European countries.

3.2. Effort and Time Estimate for System Demonstration

Malsam (2022) states that accurately estimating the total effort and time required to complete a planned system demonstration is crucial for effective project management. Precision in estimating effort and duration is essential to prevent Schedule Padding (Twproject Staff, 2018).

The project comprises complex tasks, including hardware development, software development, system integration, business suite integration, user interface design, and cost management. We must source lightweight components for hardware development, design a custom motherboard with a Motorola 68k CPU, 512Kb RAM, and solid-state storage, and implement expansion slots and multiple I/O ports. Software development includes creating a custom multi-tasking OS, creating a new programming language (HyperBasic), and implementing emulation for TeleBasic programs. System integration requires significant time and effort to ensure drivers work and basic functionality. Business suite integration can be time-consuming, and user interface design requires considerable effort. Cost management involves sourcing

affordable components and controlling manufacturing costs to reach the £250 target price (My-course, N.D.).

Watt (2019) emphasises that the five tools used for resource estimation in a successful project include expert judgment, alternative analysis, published data, bottom-up estimating, activity duration estimating, and reserve analysis, which consider realistic, optimistic, and pessimistic estimates.

The 3 Point Software Estimation Test is a method that breaks down tasks into smaller sub-tasks and estimates each using three possible scenarios: best case, most likely, and worst case. The best-case scenario assumes the best resources and a skilled team, while the most likely scenario assumes adequate resources and a skilled team. The worst-case scenario assumes limited resources and an improperly skilled team. The estimate is then calculated using the double-triangular distribution formula, which measures the effort to complete the task. This test provides a more precise estimation and reduces the risk of failures, as explained by Bhadoria (2022). The effort to complete the task is calculated using the double-triangular distribution formula –

E = (b+4m+w)/6 – E is the value for the estimate

SD = (w – b)/6 – SD stands for standard deviation, which measures the variability or uncertainty in the estimate.

To estimate the time it may take for the system demonstration, here is an example based on the ideas of Bhadoria (2022):

- In the best-case scenario, let's assume **b** = 30 days,
- The most likely scenario may take 60 days, **m** = 60 days,
- And in the worst case, it may take 180 days, so **w** = 180 days.

Ence:

*E = (30 + (4*60) + 180) / 6 = 75 days*

SD = (180 – 30)/6 = 25 days

Depending on the team size and access to pre-built components, we estimate that the system demonstration will take between 25 and 75 days to complete.

References:

- NIST (2020). *PRIVACY FRAMEWORK: A TOOL FOR IMPROVING PRIVACY THROUGH ENTERPRISE RISK MANAGEMENT.* Available from: https://www.nist.gov/system/files/documents/2020/01/16/NIST%20Privacy%20Framework_V1.0.pdf [Accessed 10 Apr. 2024].
- My-course (2021). *Estimating, Planning and Risk*. [online] Available from: https://www.my-course.co.uk/Computing/Computer%20Science/SEPM/SEPM%20Lecturecast%204/content/index.html [Accessed 10 Apr. 2024].
- Blueoptima (2023). *What Are Function Points in Software Engineering?* [online] BlueOptima. Available from: https://www.blueoptima.com/what-are-function-points-in-software-engineering/ [Accessed 10 Apr. 2024].
- Wakoli, V. (N.D.). *Python-to-Calculate-Team-Points.py*. [online] Gist. Available at: https://gist.github.com/wakoliVotes/5b58477e2fd8fd7e168a70ccde382e53 [Accessed 10 Apr. 2024].
- Malsam, W. (2022). *Software Development Estimation: A Quick Guide*. [online] ProjectManager. Available from: https://www.projectmanager.com/blog/software-development-estimation [Accessed 10 Apr. 2024].
- Twproject Staff (2018). *Twproject: project management software*. [online] Twproject: project management software, bug tracking, time tracking, planning. Available at: https://twproject.com/blog/effort-duration/ [Accessed 10 Apr. 2024].
- Watt, A. (2019). *Resource Planning – Project Management*. [online] Opentextbc.ca. Available from: https://opentextbc.ca/projectmanagement/chapter/chapter-11-resource-planning-project-management/ [Accessed 10 Apr. 2024].
- Bhadoria, L. (2022). *7 Software Test Estimation Techniques*. [online] Available at: https://www.browserstack.com/guide/software-test-estimation-techniques.
- My-course (N.D.). *SEPM New Assessment 2023: Project Management Case Study (based on real world events) | UoEO.* [online] Available from: https://www.my-course.co.uk/pluginfile.php/1120210/mod_assign/intro/2.%20SEPM%202023%20Case%20Study.pdf [Accessed 10 Apr. 2024].