Seminar 4: Activity 1

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

NIST Privacy tool	Risk Assessment Method
NIST Privacy Management Framework • Mostly for use by governments or organisations that work with governments. (Whelan, 2022) • Concerned with privacy risks	***Open Factor Analysis of Information Risk (FAIR)*** • Quantitative international standard for cybersecurity and operational risk. • FAIR model quantitifies vulnerabilities and other risk factors (Copeland, 2019) • Helps to understand risks in financial terms. • the framework is mainly concerned with establishing accurate probabilities for the frequency and magnitude of data loss events.
	Operationally Critical Threat, Asset and Vulnerability Evaluation (OCTAVE) • Identifies and manages information security risks. • Security is considered from human, technology and physical perspectives. • Identifies mission critical assets and uncovers threats and vulnerabitilities (Violino, 2021)
NIST Risk Management Framework • A set of processes for government entities to integrate	

information security and risk management into their systems development life cycles. • It links several NIST standards and guidelines to support risk management that meets requirements of the Federal Information Security Modernization Act (FISMA). • Provides processes that integrate activities such as security, privacy, and supply chain risk management into a system development lifecycle, • Can be applied to new and legacy systems, any type of system or technology, and be used within organisations of any type, size or sector.	

Looking at the NIST Privacy Tools, it seems both the Privacy Framework and the Risk Management Framework are generally geared towards government entities, although the the RMF framework can be adapted for use in any organisation. Open FAIR's biggest contribution is that it allows organisations to consider risks in terms of financial outcomes, while OCTAVE is good to identify mission-critical assets and their related threats and vulnerabilities.

Create a python program that implements one of the estimation methods covered in the lecturecast. You can use the Jupyter Notebook workspace in Codio and save your work to your GitHub repository.

```
def calculate_function_points(function_units, factor_rate):
    """A function point is a unit of measurement used to express the amount of
business functionality an information system provides to a user.
```

```
    Functional user requirements of software are identified and categorized
into one of five types: outputs, inquiries, inputs, internal files,
     and external interfaces. Each is then assessed for complexity and
assigned a number of function points.

    Suitable for scratch development projects and when requirements and all
the external interfaces are identified.
    Unsuitable for maintenance releases, migration projects, complex business
logic, OLAP applications.
    Please see https://www.fingent.com/blog/function-point-analysis-
introduction-and-fundamentals/ for more details."""

    # 1. FP = FC(PCA)
    # 2. PCA = 0.65 + 0.01Σ ci

    # function_units = [
    #     "External Inputs",            # elementary process in which data
crosses the boundary from outside to inside. This data may come from a data
input screen or another application.
    #     "External Outputs",           # elementary process in which derived
data passes across the boundary from inside to outside.
    #     "External Inquiries",         # elementary process with both input
and output components that result in data retrieval from one or more internal
logical files and external interface files.
    #     "Internal Logical Files",     # user identifiable group of logically
related data or control information maintained within the boundary of the
application.
    #     "External Interface Files"    # user identifiable group of logically
related data or control information referenced by the application, but
maintained with in the boundary of another application
    # ]

    # weight_rates = [ "Low", "Average", "High" ]

   # General system characteristic that influence the final result
   #
   # Data Communications
   # Distributed Data Processing
   # Performance
   # Heavy Configuration Use
   # Transaction Rate
   # Online Data Entry
   # End-user Efficiency
   # Online Updates
   # Complex Processing
   # Reusability
   # Ease of Installation
   # Ease of Operation
```

```
    # Supports multiple sites
    # Facilitates Change

    # Step 1
    degrees_of_influence = factor_rate * 14

    #  Step 2: Complexity adjustment factor
    CAF = 0.65 + (0.01 * degrees_of_influence)

    # Step 3: Unadjusted function point
    UFP = 0

    for i in range(5):
        for j in range(3):
            freq = function_units[i][j]
            UFP += freq * function_units[i][j]

    # Step 4: calculate final function point value
    FP = UFP * CAF

    return (UFP, CAF, FP)


if __name__ == '__main__':
    function_rates = [
        # External Inputs
        [0, 10, 0],

        # External Outputs
        [0, 2, 0],

        # External Queries
        [0, 15, 0],

        # Internal Logical Files
        [0, 5, 0],

        # External Interface Files
        [0, 0, 0],
    ]

    # Degrees of influence
    # 0 - No influence
    # 1 - Incidental
    # 2 - Moderate
    # 3 - Average
    # 4 - Significant
    # 5 - Essential
    influence = 3
```

```
(UFP, CAF, FP) = calculate_function_points(
    function_rates, influence)

# Output Values
print("Pre-processed Function Points (UFP): {}".format(UFP))
print("Complexity Adjustment Factor (CAF): {}".format(CAF))
print("Function Point Hours (FP): {}".format(FP))
```

Running the sample Python code yields the following results:

```
Pre-processed Function Points (UFP): 354
Complexity Adjustment Factor (CAF): 1.07
Function Point Hours (FP): 378.78000000000003
```

This means that given an “Average” amount of influence with an adjument factor of 1.07, and a moderately high reliance on external queries and external inputs, requires at least 378 function points of effort.

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.

Using the Function Point Estimation algorithm described above, with a “Low” influence, “Essential” outputs, and “Significant” inputs, results in 38.13 function points.

```
Pre-processed Function Points (UFP): 41
Complexity Adjustment Factor (CAF): 0.93
Function Point Hours (FP): 38.13
```

References

Copeland, J.B. (2019). Frank Kim of SANS Explains How FAIR Fits with NIST, ISO, CIS and Other Cybersecurity Frameworks. Available from https://www.risklens.com/resource-

center/blog/frank-kim-of-sans-explains-how-fair-fits-with-nist-iso-cis-and-other-cybersecurity-frameworks [Accessed 12 May 2022]

Violino, B. (2021). 5 IT risk assessment frameworks compared. Available from https://www.csoonline.com/article/2125140/it-risk-assessment-frameworks-real-world-experience.html [Accessed 12 May 2022]

Whelan, C. (2022). How NIST CSF Risk Assessments and the FAIR Risk Model Are Complementary. Available from https://www.risklens.com/resource-center/blog/how-nist-csf-and-the-fair-risk-model-are-complementary [Accessed 12 May 2022]