SVKM's NMIMS
Mukesh Patel School of Technology Management & Engineering
Computer Engineering Department
Program: B. Tech. AI Semester: VI

Course: Software Engineering
List of Experiments

Faculty: Prof. Payal .Mishra

Exp No.	Title	Prerequisite*	CO#
1	Finalize the problem statement and conduct the requirement gathering	C, C++, Software Development Life Cycle /Literature survey	CO1
2.	Estimation of Project Metrics	Software Development Life Cycle /Literature survey	CO4
3.	Modeling UML Use Case diagrams and capturing Use Case scenarios	Process Model, DFD, Requirement Engineering	CO2
4	To draw the behavioral view diagram: Sequence diagram, Collaboration diagram	Process Model, DFD, Requirement Engineering	CO2
5	Design State chart diagram using case study's Control specifications. Make use of Star UML software for design	Process Model, DFD, Requirement Engineering	CO2
6	Design level-1, Level-2 and Level-3 Data flow diagram	Knowledge of ER diagram, Process model, Requirements	CO2
7	Design the appropriate user interface diagram for your project using three golden rules. Coding for your project according to the designs analyzed in EXP.5,6	DFD, CFD and UID design	CO2
8	Coding for your project according to the designs analyzed in EXP.5,6	DFD, CFD and UID design	CO2
9	Cost Estimation of the proposed system, Schedule & Milestone using Function point estimation	Requirement Engineering, DFD, CFD and UID design	CO4
10			

* Students are expected to be ready with the prerequisite before attending the lab

Experiment No.09

PART A

(PART A: TO BE REFFERED BY STUDENTS)

A.1 Aim: Cost Estimation of the proposed system, Schedule & Milestone using Function point estimation

A.2 Prerequisite:

Feasibility study of the system

A.3 Outcome:

Cost Estimation, Time estimation

A.4 Theory:

Project Estimation

- Project scope must be understood
- Elaboration (decomposition) is necessary
- Historical metrics are very helpful
- At least two different techniques should be used
- Uncertainty is inherent in the process

Function Point(FP) estimation:

The function point metric (FP), first proposed by Albrecht , can be used effectively as a means for measuring the functionality delivered by a system.

- Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and assessments of software complexity

Information domain values are defined in the following manner:

- number of external inputs (EIs)
- number of external outputs (EOs)
- number of external inquiries (EQs)

- number of internal logical files (ILFs)
- Number of external interface files (EIFs)

Information		**Weighting Factor**			
Domain Value	**Count**		**Simple**	**Average**	**Complex**
External Inputs	_____	x	3	4	6 = _____
External Outputs	_____	x	4	5	7 = _____
External Inquiries	_____	x	3	4	6 = _____
Internal Logical Files	_____	x	7	10	15= _____
External Interface Files	_____	x	5	7	10 = _____
Count total				_____	

FP=count total*[0.65+(0.01*sigma(Fi)]

External Inputs (EI) - is an elementary process in which data crosses the boundary from outside to inside. This data may come from a data input screen or another application. The data may be used to maintain one or more internal logical files. The data can be either control information or business information. If the data is control information it does not have to update an internal logical file.

External Outputs (EO) - an elementary process in which derived data passes across the boundary from inside to outside. Additionally, an EO may update an ILF. The data creates reports or output files sent to other applications. These reports and files are created from one or more internal logical files and external interface file. The following graphic represents on EO with 2 FTR's there is derived information (green) that has been derived from the ILF's

External Inquiry (EQ) - an elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files. The input process does not update any Internal Logical Files, and the output side does not contain derived data. The graphic below represents an EQ with two ILF's and no derived data.

Internal Logical Files (ILF's) - a user identifiable group of logically related data that resides entirely within the applications boundary and is maintained through external inputs.

External Interface Files (EIF's) - a user identifiable group of logically related data that is used for reference purposes only. The data resides entirely outside the application and is maintained by another application. The external interface file is an internal logical file for another application.

Value adjustment factors:

1) Does the system require reliable backup and recovery? - 3
2) Are specialized data communications required to transfer information to or from the application? - 4
3) Are there distributed processing functions? - 0
4) Is performance critical? – 4
5) Will the system run in an existing, heavily utilized operational environment? - 4
6) Does the system require on-line data entry? - 5
7) Does the on-line data entry require the input transaction to be built over multiple screens or operations? -2
8) Are the internal logical files updated on-line?- 5
9) Are the inputs, outputs, files, or inquiries complex? - 3
10) Is the internal processing complex? - 4
11) Is the code designed to be reusable? - 5
12) Are conversion and installation included in the design? - 3
13) Is the system designed for multiple installations in different organizations? - 0
14) Is the application designed to facilitate change and for ease of use by the user? – 5

Solved example:

The estimated number of FP is derived:

FPestimated = count-total 3 [0.65 + 0.01 3 S (Fi)]

FPestimated = 375

organizational average productivity = 6.5 FP/pm.

burdened labor rate = $8000 per month, the cost per FP is approximately $1230.

Based on the FP estimate and the historical productivity data, the total estimated project cost is $461,000 and the estimated effort is 58 person-months.

Task to be completed:

1) Complete the Function point estimation for your project

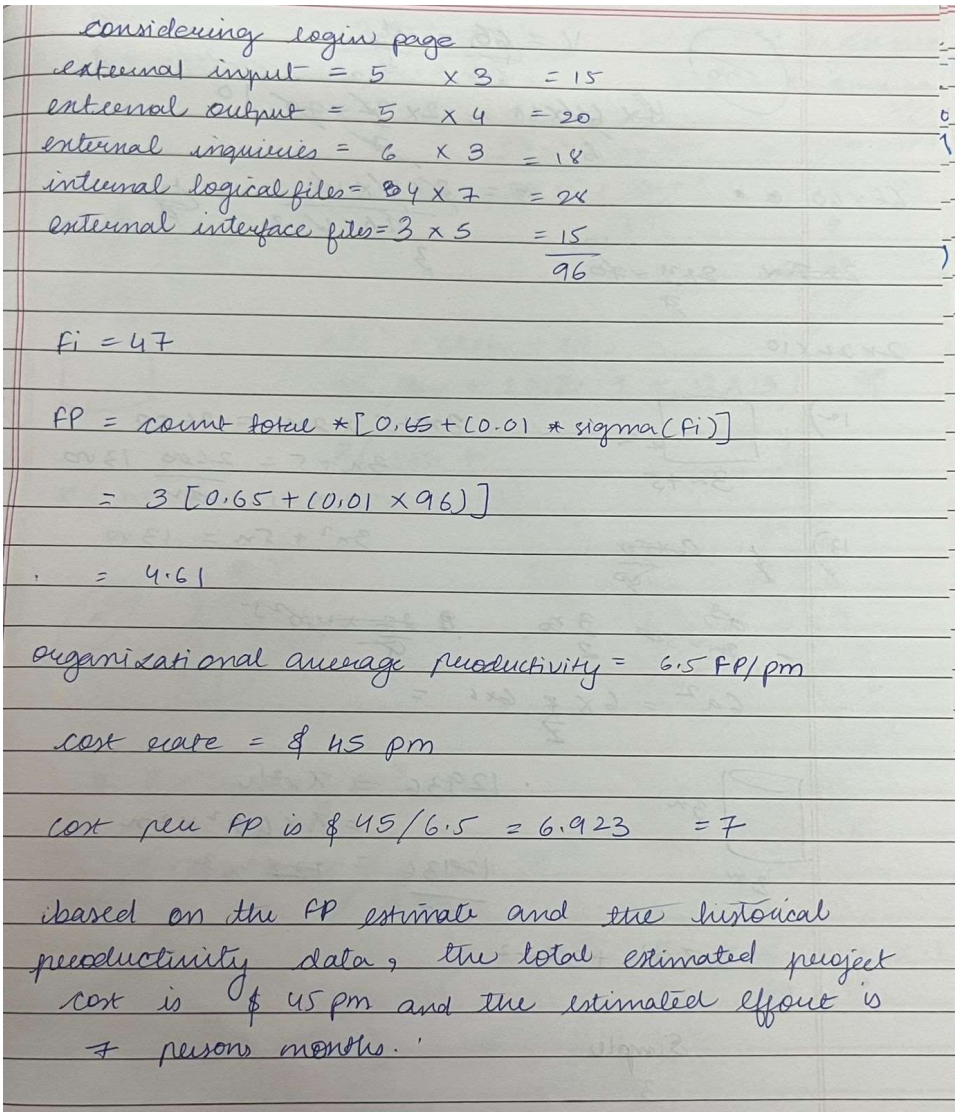
PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

Roll No. I070	Name: Ekta Mistry
Program : AI	Division: I
Batch:	Date of Experiment:
Date of Submission:	Grade :

B.1 Tasks given in PART A to be completed here


considering login page
external input = 5 x 3 = 15
external output = 5 x 4 = 20
external inquiries = 6 x 3 = 18
internal logical files = 4 x 7 = 28
external interface files = 3 x 5 = 15
96

Fi = 47

FP = count total * [0.65 + (0.01 * sigma(Fi))]

= 3 [0.65 + (0.01 x 96)]

= 4.61

organizational average productivity = 6.5 FP/pm

cost rate = $ 45 pm

cost per FP is $ 45/6.5 = 6.923 = 7

'based on the FP estimate and the historical productivity data, the total estimated project cost is $ 45 pm and the estimated effort is 7 persons months.'

B.4 Question of curiosity:

1. What is Function Point Analysis? What is A Function Point?

Function Point Analysis (FPA) is a method used to measure the size and complexity of software projects. It is a technique that was first proposed by Allan Albrecht in 1979 as a way to measure the functionality provided to the user by software.

A function point is a unit of measurement used in FPA to quantify the functionality provided by a software system. It is a measure of the software's capabilities in terms of the business requirements it meets, rather than a measure of the lines of code or other technical factors.

Function points are calculated by analyzing the software system's input and output interfaces, user interactions, data storage and processing requirements, and other factors. The resulting function point count provides a measure of the software's functionality, which can be used to estimate project effort, cost, and schedule.

FPA is a widely used technique in software development, particularly in the areas of software estimation, project planning, and quality assurance. It provides a standardized and objective way to measure software functionality, which can be used to compare different software systems and to track changes in functionality over time.

2. Who created Function Points Analysis? Why it was created?

The motivation behind FPA was to develop a standardized and objective way to measure the functional size of software systems. By focusing on the functionality provided to the user, rather than the technical aspects of the software, FPA provides a more accurate and meaningful measure of the software's value and complexity.

FPA was developed as a response to the limitations of other measurement techniques, such as lines of code, which were found to be unreliable and inconsistent measures of software development productivity. FPA provides a more holistic view of software functionality and is used to estimate project effort, cost, and schedule. It has become a widely used technique in software development, particularly in the areas of software estimation, project planning, and quality assurance.

3. Is the Function Point Analysis technique owned by some company?

No, Function Point Analysis (FPA) is not owned by any company. FPA is a standardized technique that is recognized and supported by several professional organizations, including the International Function Point Users Group (IFPUG) and the Consortium for IT Software Quality (CISQ).

These organizations provide guidance and standards for the application of FPA, and they offer certifications and training to professionals who want to become proficient in FPA. However, the FPA technique itself is not owned by any company or individual, and it can be used by anyone who wants to measure the functional size of software systems.

It's worth noting that there are commercial tools available that can assist with FPA analysis, but these tools are not necessary to perform FPA. FPA can be performed using manual methods or with the help of spreadsheets or other tools that do not require payment of licensing fees or ownership by any company.

4. What are Function Point Analysis benefits?

Function Point Analysis (FPA) is a method used to measure the functionality provided by a software system. It is often used in software development projects to estimate the size of the

software system, assess the complexity of the system, and provide a basis for estimating project effort and cost. The benefits of FPA include:

1. Standardized measurement: FPA provides a standardized method for measuring the size and complexity of a software system. This makes it easier to compare different software systems, and to estimate project effort and cost.
2. Accurate estimation: FPA can provide accurate estimates of project effort and cost, based on the size and complexity of the software system. This can help to ensure that projects are properly scoped and resourced, and that project budgets are realistic.
3. Improved productivity: FPA can help to identify opportunities to improve productivity by identifying areas of the software system that are particularly complex or time-consuming.
4. Better communication: FPA can help to improve communication between stakeholders by providing a common language for discussing the size and complexity of the software system.
5. Quality assurance: FPA can be used as a tool for quality assurance, by ensuring that the software system meets the functional requirements specified in the project scope.
6. Risk management: FPA can help to identify potential risks associated with the software system, such as complexity, size, and the number of interfaces.

Overall, Function Point Analysis is a useful tool for software development projects, as it provides a standardized method for measuring the size and complexity of a software system, and can help to improve project estimation, productivity, communication, quality assurance, and risk management.

5. Is it necessary to be a software developer to do Function Point Analysis?

No, it is not necessary to be a software developer to do Function Point Analysis (FPA). FPA is a technique used to measure the functional size of a software system, and it involves analyzing the functional requirements of the system to determine the number of function points.

While some knowledge of software development can be helpful, FPA can be performed by anyone who has a good understanding of the functional requirements of the system being analyzed. This could include business analysts, project managers, quality assurance professionals, or anyone else with a strong understanding of the functional requirements of the software system.

That being said, it is important to note that FPA requires a detailed understanding of the functional requirements of the software system being analyzed, as well as a good understanding of the FPA methodology and rules. It can be helpful to receive training or certification in FPA in order to perform it effectively.

6. Who uses Function Point Analysis in the world?

Function Point Analysis (FPA) is used by a wide range of organizations and professionals around the world, including:

1. Software development companies: FPA is commonly used by software development companies to estimate project effort and cost, and to measure the productivity and quality of their software development process.
2. Government agencies: FPA is often used by government agencies to assess the size and complexity of software systems, and to ensure that they meet functional requirements and standards.
3. Consulting firms: FPA is used by consulting firms to provide clients with accurate estimates of project effort and cost, and to help identify opportunities to improve productivity and quality.
4. IT departments: FPA is used by IT departments to measure the size and complexity of their software systems, and to identify opportunities for improvement.
5. Project managers: FPA is used by project managers to estimate project effort and cost, and to ensure that projects are properly scoped and resourced.
6. Business analysts: FPA is used by business analysts to assess the functional requirements of software systems, and to ensure that they meet business needs.

Overall, FPA is a widely used technique that is used by a variety of professionals and organizations around the world to measure the size and complexity of software systems, and to estimate project effort and cost.

7. What tools are suitable for support and/or to automate the use of FPA?

There are several tools available that can support and automate the use of Function Point Analysis (FPA). These tools can help to improve the accuracy and efficiency of the FPA process, and can also provide additional analysis and reporting capabilities. Some examples of FPA tools include:

1. Automated FPA tools: These tools use artificial intelligence and machine learning algorithms to analyze software requirements and automatically generate function point counts. Examples of these tools include CodeSight, Fuse and Automated Function Point Analysis.
2. Manual FPA tools: These tools are designed to support the manual FPA process, by providing templates and guidelines for documenting software requirements and calculating function points. Examples of these tools include FP-Tracker, QSM Function Point Workbench and MetriQ.
3. Project management tools: Many project management tools, such as Jira, Trello, and Asana, include FPA capabilities or integrations that can support the FPA process by helping to organize and document software requirements and estimate project effort.
4. Reporting and analysis tools: These tools are designed to provide additional analysis and reporting capabilities for FPA data, including metrics such as productivity, quality, and risk. Examples of these tools include CAST Application Intelligence Platform and QSM Metrics Dashboard.

Overall, the choice of FPA tool will depend on the specific needs and requirements of the organization or individual using it. Some organizations may prefer to use automated FPA tools for their speed and accuracy, while others may prefer to use manual tools for their flexibility and control.

8. Why automatic tools cannot correctly count function points?

Automatic tools can correctly count function points, but there are some limitations to their accuracy. These limitations are primarily due to the fact that function point analysis (FPA) requires a deep understanding of the software's functionality and the context in which it is being used.

One of the challenges with using automated FPA tools is that they may not be able to accurately interpret the nuances of software requirements or understand the broader context in which the software is being used. This can result in incorrect function point counts if the tool is not able to accurately interpret the requirements and context.

Another challenge with automated FPA tools is that they may not be able to identify non-functional requirements, such as performance, security, or usability. These requirements can have a significant impact on the size and complexity of the software system, and if they are not included in the function point count, the resulting estimate may be inaccurate.

Additionally, automated FPA tools may not be able to identify or handle certain types of software functionality, such as complex algorithms or novel user interfaces. This can lead to inaccuracies in the function point count and an inaccurate estimate of the software system's size and complexity.

While automated FPA tools can be useful for streamlining the function point counting process, it is important to have a deep understanding of FPA methodology and rules, and to carefully review the results to ensure accuracy. Ultimately, the best approach may be a combination of automated and manual FPA methods, depending on the specific needs of the software development project.

9. What kind of software can be measured by Function Points?

Function Point Analysis (FPA) can be used to measure the functional size of any software system that has well-defined functional requirements. This includes a wide range of software types and applications, such as:

1. Desktop applications: FPA can be used to measure the functional size of desktop applications, such as word processors, spreadsheet software, and graphics editors.
2. Web applications: FPA can be used to measure the functional size of web applications, such as e-commerce platforms, social media sites, and content management systems.
3. Mobile applications: FPA can be used to measure the functional size of mobile applications, such as games, productivity tools, and social media apps.

4. Embedded systems: FPA can be used to measure the functional size of embedded systems, such as automotive systems, medical devices, and consumer electronics.
5. Enterprise systems: FPA can be used to measure the functional size of enterprise systems, such as customer relationship management (CRM) systems, enterprise resource planning (ERP) software, and supply chain management systems.
6. Custom software: FPA can be used to measure the functional size of custom software systems that are designed to meet specific business needs.

Overall, any software system that has well-defined functional requirements can be measured using FPA, regardless of the specific technology or platform that is used.

10. Is it possible to use FPA in a project using agile methodology?

Yes, it is possible to use Function Point Analysis (FPA) in a project using agile methodology. In fact, FPA can be a valuable tool for estimating and measuring the size and complexity of software systems in agile projects, which typically prioritize flexibility and responsiveness to changing requirements.

Agile methodology typically involves iterative development cycles and continuous delivery of working software, which can make it challenging to accurately estimate the size and effort required for each iteration or release. FPA can help address this challenge by providing a consistent and standardized method for estimating the functional size of the software system, which can be used to inform release planning and resource allocation.

To use FPA in an agile project, it is important to incorporate the FPA process into the overall project management framework. This may involve using FPA tools to document and track software requirements, as well as regularly reviewing and updating function point counts to ensure they remain accurate as requirements evolve.

It is also important to recognize that agile projects may require more frequent and flexible use of FPA, as requirements and functionality may change rapidly over the course of the project. This may require a more collaborative approach to FPA, with regular engagement from project stakeholders and frequent updates to the function point count to reflect changing requirements.

Overall, FPA can be a valuable tool for estimating and measuring software size and complexity in agile projects, but it is important to tailor the FPA process to the specific needs and requirements of the project, and to incorporate it into the overall project management framework.

****