Group Project REPORT DOCUMENT

Software Metrics Dashboard Development

Team Members: Nhan Tran Phuc Thi

Truong Thi Mai Tuan

Huynh Anh

Vu Tran Thanh

Da Nang, 2024

PROJECT INFORMATION

I. Information:

Project acronym	LOC, FP, UCP, EVM		
Project Title	Software Metrics Dashboard Development (LOC, FP, UCP, EVM)		
Start Date	Mar 7, 2023	**End Date**	May 20, 2023
Lead Institution	International School, Duy Tan University		
Project Mentor	MSc Man, Nguyen Duc		
Project Leader & Contact Details	Vu, Tran Thanh Email: vutv579@gmail.com Tel: 0702374029		
Partner Organization	Duy Tan University		
Project Web URL			
Team members	Vu, Tran Thanh	vutv579@gmail.com	0702374029
	Tuan, Huynh Anh	anhtuan2003147@gmail.com	0369705323
	Thi, Truong Thi Mai	maithivip611@gmail.com	0396696223
	Nhan, Tran Phuc	phucnhant@gmail.com	0865003881

Name	Task
Vu, Tran Thanh	- EVM
Tuan, Huynh Anh	-LOC
Thi, Truong Thi Mai	-UCP
Nhan, Tran Phuc	-FP

II. Topic description:

Topic 4- Software Metrics Dashboard Development: This project could focus on designing and implementing a software metrics dashboard that provides real time visibility into key metrics for a software project or organization.
Guidelines might include identifying relevant metrics, creating visualization components, and integrating data sources for automatic updates. (LOC, FP, UCP, EVM, DRE)

III. Goal:

The project " Software Metrics Dashboard Development" aims to provide a practical and efficient tool consisting of LOC, FP, UCP, EVM calculation functions.
Goals can be divided into the following categories:

-Create intuitive graphical dashboards: Dashboards should be designed to be easy to use and understand

-Clearly present necessary metrics: Dashboards need to display important metrics such as LOC, FP, UCP, EVM in a clear and transparent manner

IV. The theory of measurement:

LINE OF CODE (LOC)

1. Definition:

- Line of code (LOC) refers to the number of individual lines of programming code in a software application or program.

- It is a common metric used by software developers and managers to measure the size, complexity, and productivity of software development projects.

- NCLOC and CLOC are measurement metrics used in software development to count the lines of code in a program.

  + NCLOC: non-commented source line of code or effective lines of code (ELOC).

  + CLOC: commented source line of code

- By measuring NCLOC and CLOC separately we can define:

$$\text{Total length (LOC)} = \text{NCLOC} + \text{CLOC}$$

- The units of LOC are:

+ KLOC- Thousand lines of code

+ NLOC- Non comment lines of code

+ KDSI- Thousands of delivered source instruction

- LOC can be used to estimate project timelines, allocate resources, and assess the quality of the code. It is also used in compliance with industry standards and regulations.

- While it is not a perfect measure of software development progress or quality, it remains a widely used metric in the industry.

2. Measurement methods:

- There are two main types of LOC metrics: Physical LOC and Logical LOC. Physical LOC counts all lines of code, including comments and blank lines, while Logical LOC counts only the lines of code that contribute to the functionality of the software. Other variations of LOC metrics include Executable LOC, which counts only the lines of code that are compiled and executed by the computer, and Source Lines of Code per Function Point (SLOC/FP), which measures the amount of code required to implement a single function point in a software system.

a. Physical LOC (Lines of Code)

- A measure of the number of lines of code in a software system.

- It counts all lines of code, including comments and blank lines, and is often used as a metric to estimate the size and complexity of a software project.

- However, it is important to note that SLOC alone is not a comprehensive measure of software quality or productivity, and should be used in conjunction with other metrics and analysis techniques.

b. Logical SLOC (also known as LLOC)

- A measure of SLOC. It counts the number of statements or logical lines of code in a program, which includes all executable statements, declarations, and comments.

- Logical SLOC is used to measure the size and complexity of a software system and is often used for estimating development effort, testing effort, and maintenance effort.

- However, it does not take into account the physical structure of the code or the number of lines actually written, which can vary depending on coding style and formatting.

3. Advantages of LOC Metrics:

"Measuring software productivity by lines of code is like measuring progress on an airplane by how much it weighs."- Bill Gates

a. Scope for automation of counting:

- LOC is a physical entity, manual counting effort can be easily eliminated by automating the counting process. Small utilities may be developed for counting the LOC in a program.

b. Estimating project size and complexity:

- LOC can be used as a measure of the size and complexity of a software project. This information is useful for project planning, scheduling, and resource allocation.

c. Measuring productivity:

- By tracking the number of lines of code produced by developers over time, managers can assess their productivity and identify areas for improvement.

d. Evaluating software quality:

- The number of lines of code can be an indicator of software quality. For example, if a program has a large number of lines of code but produces very little output, it may be an indication that the code is inefficient or poorly written.

e. Identifying potential bugs:

- Large programs with many lines of code are more likely to have bugs than smaller programs. By counting the number of lines of code, developers can identify areas that may need more testing or debugging.

f. Compliance with industry standards:

- Some industries have standards that require software projects to meet certain size or complexity criteria. Counting the number of lines of code can help ensure compliance with these standards.

4. Disadvantages of LOC Metrics:

a. Limited scope:

- LOC only measures the size of the program's code and does not take into account other important aspects of the software development process, such as requirements, design, testing, or documentation.

b. Difference in languages:

- Counting lines of code is not always an accurate way to measure the complexity or quality of software. Different programming languages and coding styles can result in vastly different line counts

for similar functionality. Consider two applications that provide the same functionality one is written in Java and the other is written in C#:

+ The number of function points would be exactly the same, but aspects of the application would be different.

+ The lines of code needed to develop the application would certainly not be the same. As a consequence, the amount of effort required to develop the application would be different (hours per function point).

+ Unlike Lines of Code, the number of Function Points will remain constant

c. Narrow perspective:

- LOC characterizes only one aspect of size, namely length, and does not account for functionality or complexity. Therefore, a program with a smaller LOC may be more complex and harder to maintain than a program with a larger LOC.

d. Quality concerns:

- Using LOC as a sole metric for evaluating the productivity of programmers may encourage bad programming practices, such as code duplication or excessive comments, which can result in bloated and less maintainable code.

e. Lack of user-friendliness:

- LOC is a technical metric that may not be easily understandable for non-technical users, making it challenging to communicate the program's size and complexity to stakeholders outside of the development team.

FUNTION POINT (FP):

1. Definition:

- Function Point is a method of measuring the size of software based on the functionality and value it provides, rather than the number of lines of code. This makes FP independent of the programming language and technology used.

- FP is used to:
  * Estimate the effort required for software development and maintenance.
  * Compare productivity and efficiency between projects or development teams.
  * Manage projects and monitor the progress of software development.

2. Method measurement:

- FP is calculated based on five main types of functions:

Measurements Parameters	Examples
1.Number of External Inputs(EI)	Input screen and tables
2. Number of External Output (EO)	Output screens and reports
3. Number of external inquiries (EQ)	Prompts and interrupts.
4. Number of internal files (ILF)	Databases and directories
5. Number of external interfaces (EIF)	Shared databases and sharec routines.

-Each type of function is assessed based on complexity (low, medium, high) and assigned a corresponding weight. The total number of FPs is calculated by multiplying the total number of unadjusted functions (UFP) by an adjustment factor (CAF) based on factors that affect the project

Measurement Parameter	Low	Average	High
1. Number of external inputs (EI)	7	10	15
2. Number of external outputs (EO)	5	7	10
3. Number of external inquiries (EQ)	3	4	6
4. Number of internal files (ILF)	4	5	7
5. Number of external interfaces (EIF)	3	4	6

F.P = UFP x VAF
UFP : Unadjusted Fucntional Point
VAF : Value Adjustment Factor

Weighting: EI, EO, EQ, ILF, EIF
UFP = Weighting*N(EI) + Weighting*N(EO) + Weighting*N(EQ) + Weighting*N(ILF) + Weighting*N(EIF)
VAF = 0.65 + (0.01* ∑Fi)

3. Advantages of Functional Point Analysis:

- Technological Independence: It calculates a software system's functional size independent of the underlying technology or programming language used to implement it. As a result, it is a technology-neutral metric that makes it easier to compare projects created with various technologies.

- Better Accurate Project Estimation: It helps to improve project estimation accuracy by measuring user interactions and functional needs. Project managers can improve planning and budgeting by using the results of the FPA to estimate the time, effort and resources required for development.

- Improved Interaction: It provides a common language for business analysts, developers, and project managers to communicate with one another and with other stakeholders. By communicating the size and complexity of software in a way that both technical and non-technical audiences can easily understand this helps close the communication gap.

- Making Well-Informed Decisions: FPA assists in making well-informed decisions at every stage of the software development life cycle. Based on the functional requirements, organizations can use the results of the FPA to make decisions about resource allocation, project prioritization, and technology selection.

- Early Recognition of Changes in Scope: Early detection of changes in project scope is made easier with the help of FPA. Better scope change management is made possible by the measurement of functional requirements, which makes it possible to evaluate additions or changes for their effect on the project's overall size.

4. Disadvantages of Functional Point Analysis:

- Subjective Judgement: One of the main disadvantages of Functional Point Analysis is it's dependency on subjective judgement i.e. relying on personal opinions and interpretations instead of just using clear, measurable standards.

- Low Accuracy: It has low evaluation accuracy as it's dependency on subjective judgement. Time Consuming: Functional Point Analysis is a time consuming process, particularly during the initial stages of implementation.

-Steep Learning Curve: Learning FPA can be challenging due to its complexity and the length of time required to gain proficiency.

- Less Research Data: Compared to LOC-based metrics, there is relatively less research data available on function points.

- Costly: The need for thorough analysis and evaluation can result in increased project timelines and associated costs.

USE CASE POINT (UCP):

1. Defintion:

- Use case points (UCP or UCPs) is a software estimation technique used to forecast the software size for software development projects. UCP is used when the Unified Modeling Language (UML) and Rational Unified Process (RUP) methodologies are being used for the software design and development. The concept of UCP is based on the requirements for the system being written using use cases, which is part of the UML set of modeling techniques. The software size (UCP) is calculated based on elements of the system use cases with factoring to account for technical and environmental considerations. The UCP for a project can then be used to calculate the estimated effort for a project.

2. Method measurement:

The method for determining the size estimate to develop a system is based on a calculation with the following elements:

- Unadjusted Use Case Weight (UUCW) – the point size of the software that accounts for the number and complexity of use cases.
- Unadjusted Actor Weight (UAW) – the point size of the software that accounts for the number and complexity of actors.
- Technical Complexity Factor (TCF) – factor that is used to adjust the size based on technical considerations.
- Environmental Complexity Factor (ECF) – factor that is used to adjust the size based on environmental considerations.

Unadjusted Use Case Weight (UUCW)

Use Case Classification	No. of Transactions	Weight
Simple	1 to 3 transactions	5
Average	4 to 7 transactions	10
Complex	8 or more transactions	15

UUCW = (Total No. of Simple Use Cases x 5) + (Total No. Average Use Cases x 10) + (Total No. Complex Use Cases x 15)

Unadjusted Actor Weight (UAW)

Actor Classification	Type of Actor	Weight
Simple	External system that must interact with the system using a well-defined API	1
Average	External system that must interact with the system using standard communication protocols (e.g. TCP/IP, FTP, HTTP, database)	2
Complex	Human actor using a GUI application interface	3

UAW = (Total No. of Simple actors x 1) + (Total No. Average actors x 2) + (Total No. Complex actors x 3)

Use Case Points (UCP)

UCP = (UUCW + UAW)

3. Advantages of Functional Point Analysis:

- UCPs are based on use cases and can be measured very early in the project life cycle.

- UCP (size estimate) will be independent of the size, skill, and experience of the team that implements the project.

-UCP based estimates are found to be close to actuals when estimation is performed by experienced people.

-UCP is easy to use and does not call for additional analysis.

- Use cases are being used vastly as a method of choice to describe requirements. In such cases, UCP is the best suitable estimation technique.

4. Disadvantages of Functional Point Analysis:

- UCP can be used only when requirements are written in the form of use cases.

- Dependent on goal-oriented, well-written use cases. If the use cases are not well or uniformly structured, the resulting UCP may not be accurate.

- Technical and environmental factors have a high impact on UCP. Care needs to be taken while assigning values to the technical and environmental factors.

- UCP is useful for initial estimate of overall project size but they are much less useful in driving the iteration-to-iteration work of a team.

Earned Value Management (EVM):

1. Defintion:

- EVM is possibly one of the most productive techniques to measure the performance of a project. A project manager always follows a plan-do-check-act management cycle to ensure all the measures of the project. That's where earned value management helps the project managers to conclude the plan-do-check-act management cycle. EVM allows the project manager to find that all the tasks of the project are going on according to the schedule or not, whether the problems occurring are critical or not. EVM not only helps in recognizing the problems but also provides the approach that is needed to take the project back on track.

2. Method measurement:

1. PV = (BAC/expected duration) * actual completion time

   PV: Planned Value

   BAC: Budget at completion

2. SV = EV - PV

   SV: Schedule Variance

   EV: Earned Value

   AC: Actual Cost

3. SPI = EV/PV

   SPI: Schedule Performance Index

4. CV = EV - AC

   CV: Cost Variance

5. CPI = EV / AC

   CV: Cost Performance Index

6. EAC = (BAC - EV)/CPI - AC

   EAC: Estimate at completion

7. ETC = EAC - AC

   ETC: Estimate to completion

8. VAC = EAC - BAC

   VAC: Variance at completion

3. Advantages of Functional Point Analysis:

- Performance Measurement: EVM provides an objective and quantitative method of measuring project performance. It allows project managers to track project progress against planned performance and identify any deviations from the plan.

- Early Warning: EVM provides early warning signs of potential problems that may arise in the project. It enables project managers to take corrective action before the problems become too severe.

- Improved Communication: EVM provides a common language for project managers, team members, and stakeholders to discuss project performance. It promotes better communication and collaboration among all parties involved in the project.

- Cost Control: EVM enables project managers to track and control project costs more effectively. It provides a mechanism for monitoring cost performance and identifying cost variances, allowing project managers to take corrective action.

- Predictability: EVM provides a reliable method for predicting project outcomes. By analyzing cost and schedule performance data, project managers can make informed decisions about future project performance and adjust project plans accordingly.

4. Disadvantages of Functional Point Analysis:

- Complexity: EVM can be a complex technique to implement and understand. It requires a significant amount of training and experience to use effectively.

- Data Requirements: EVM requires accurate and reliable data to be effective. If the data is inaccurate or incomplete, the EVM analysis will be flawed.

- Focus on Cost: EVM tends to focus on cost performance and may not provide a complete picture of project performance. It does not consider other factors such as quality, customer satisfaction, and stakeholder engagement.

- Subjectivity: EVM involves subjective judgments, such as estimating the percentage of work completed. This can introduce bias and affect the accuracy of the EVM analysis.

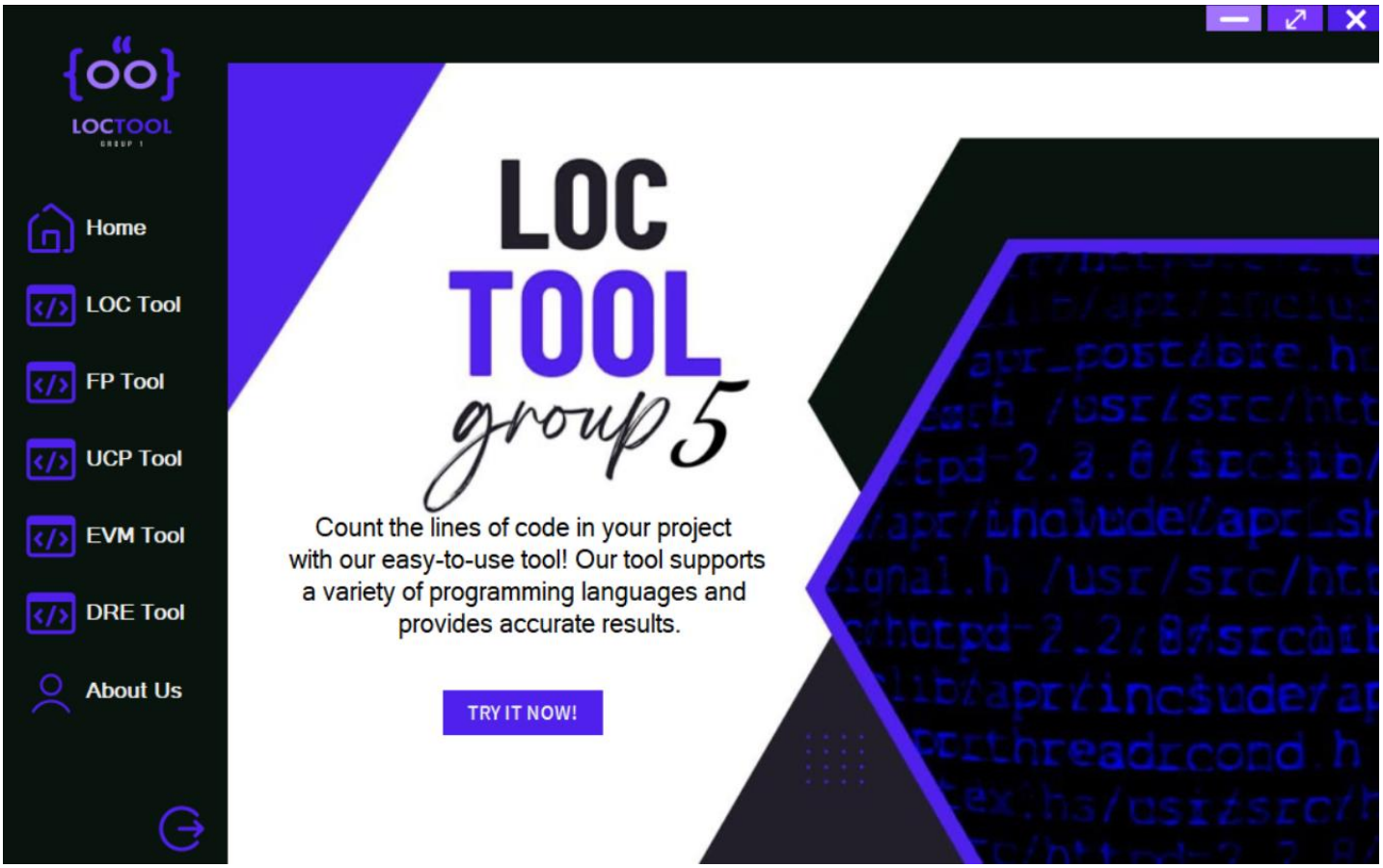
- Lack of Flexibility: EVM is based on a predetermined project plan, and any changes to the plan can make the EVM analysis less useful. It may not be suitable for projects that require a high degree of flexibility and adaptability.

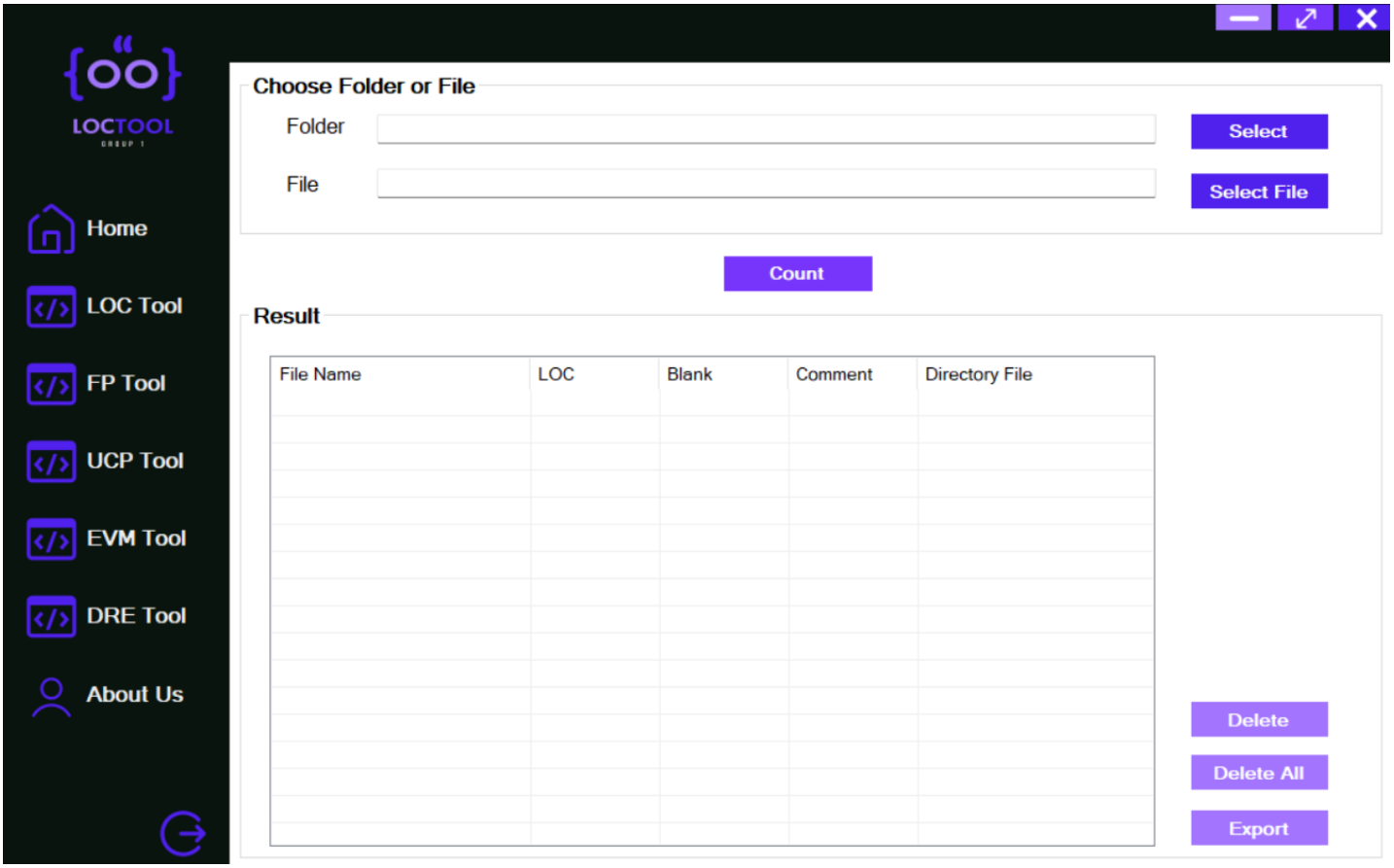
V. Plan:

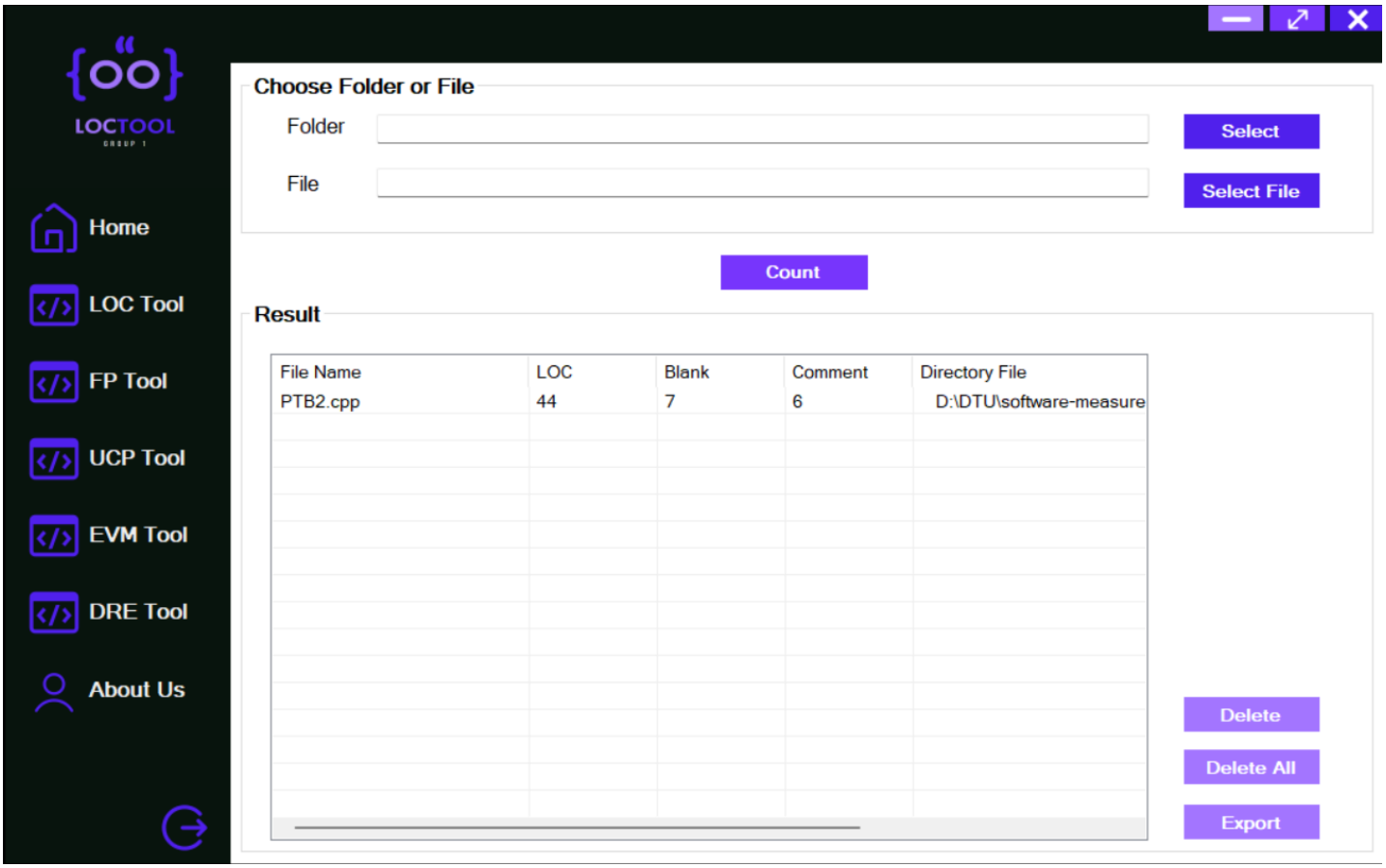
1. Learn about LOC, FP, UCP, EVM how to calculate it...
2. Design the user interface (UI):
   - Use the UI designer to create controls such as TextBox, Button, Label to display information and allow users to input data.
   - Design the interface so that users can input the path to the file or directory to count LOC, FP, UCP, EVM.
3. Build the LOC, FP, UCP, EVM counting logic:
   - Write code to count the number of lines in a file or an entire directory and its subfiles.
   - Utilize methods and classes from the .NET library to traverse files, read each line, and count the lines containing code.
4. Connect the interface and logic:
   - Define events for controls on the interface such as Button Click event. In this event, call the LOC, FP, UCP, EVM counting logic and display the result on the interface.
   - Handle errors and exceptions
   - Handle potential exceptions such as file not found or errors while reading files.

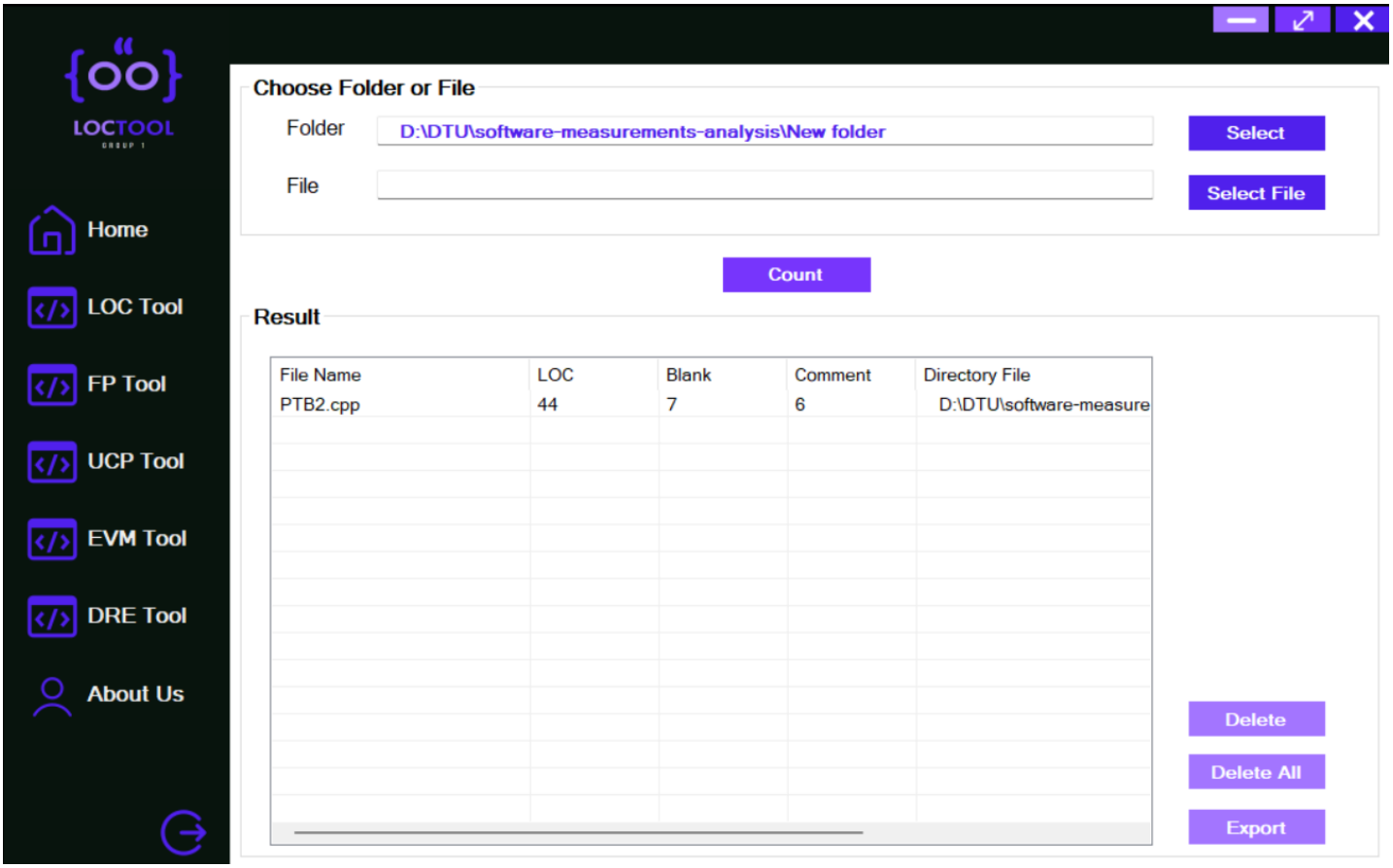
5. Test the application:
   - Verify the correctness of the application by inputting paths to files or directories containing code and checking the calculated result.
6. Optimize and maintain:
   - Review and optimize the source code to ensure smooth and efficient operation of the application.
   - Maintain and update the application based on user requirements and feedback.
7. Deploy the application:
   - Build and deploy the application so that users can use it.

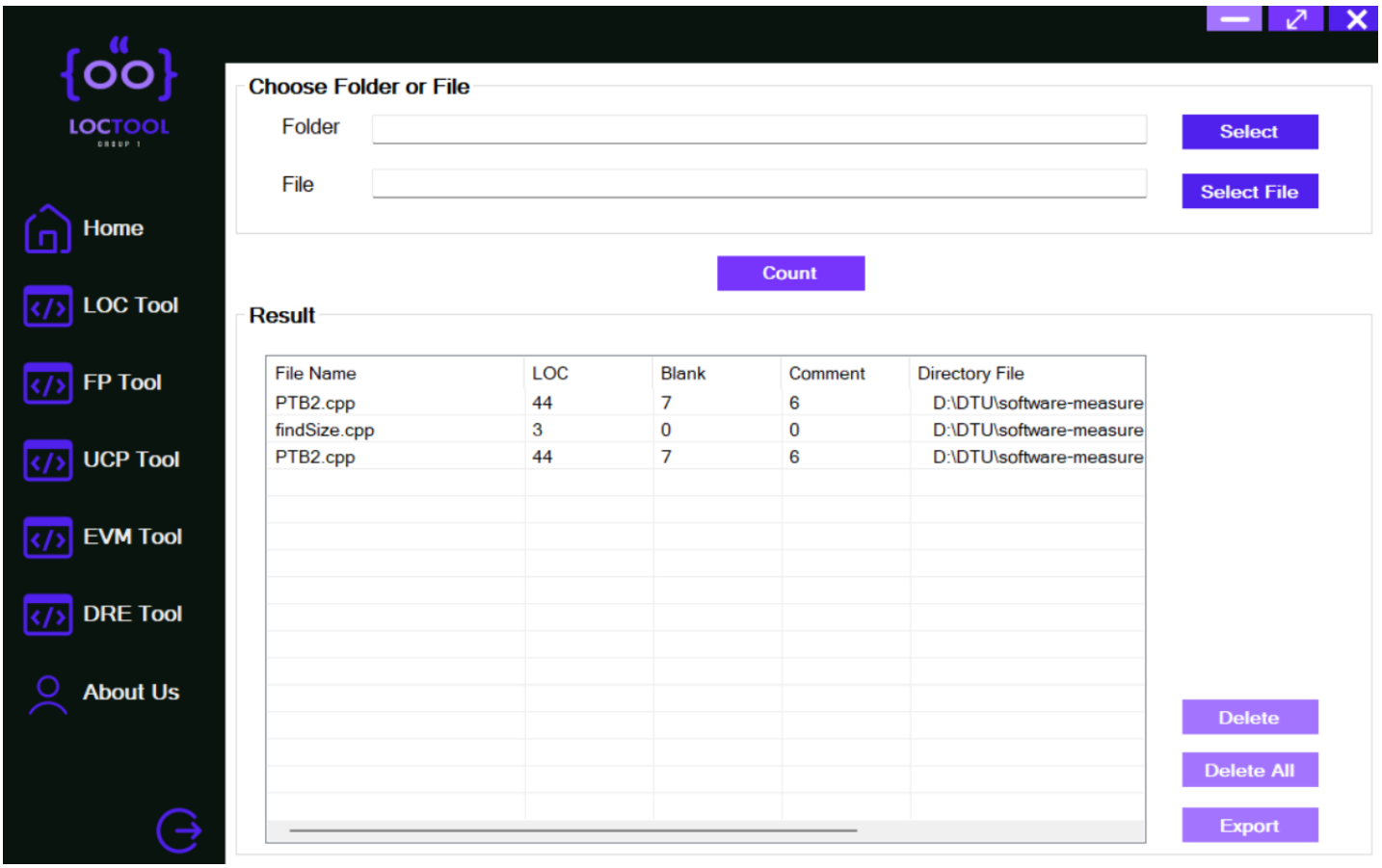
VI. Demo:

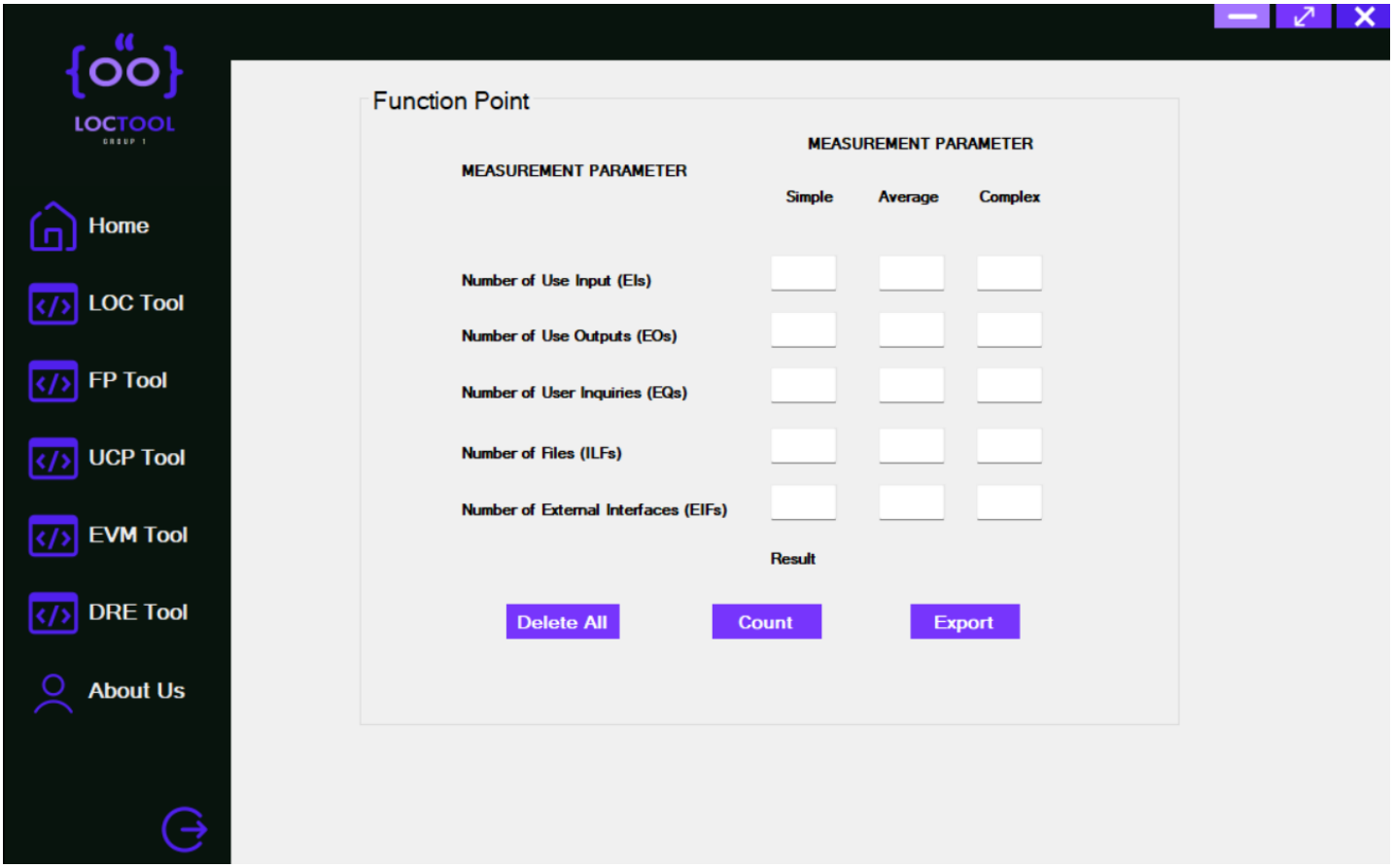

LOCTOOL
Home
LOC Tool
FP Tool
UCP Tool
EVM Tool
DRE Tool
About Us
LOC
TOOL
group 5
Count the lines of code in your project with our easy-to-use tool! Our tool supports a variety of programming languages and provides accurate results.
TRY IT NOW!

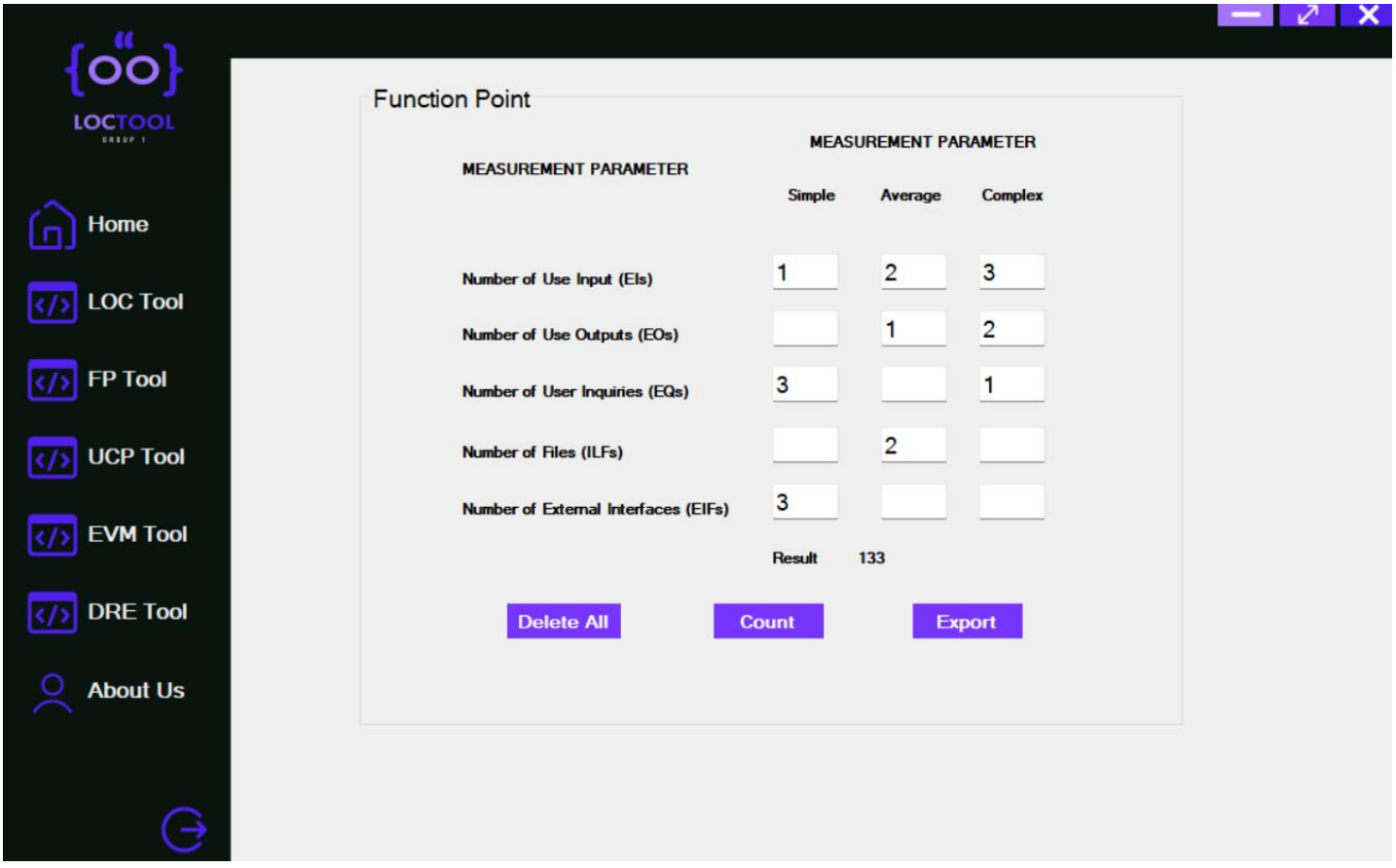

LOCTOOL
Home
LOC Tool
FP Tool
UCP Tool
EVM Tool
DRE Tool
About Us
Choose Folder or File
Folder
File
Select
Select File
Count
Result
File Name
LOC
Blank
Comment
Directory File
Delete
Delete All
Export

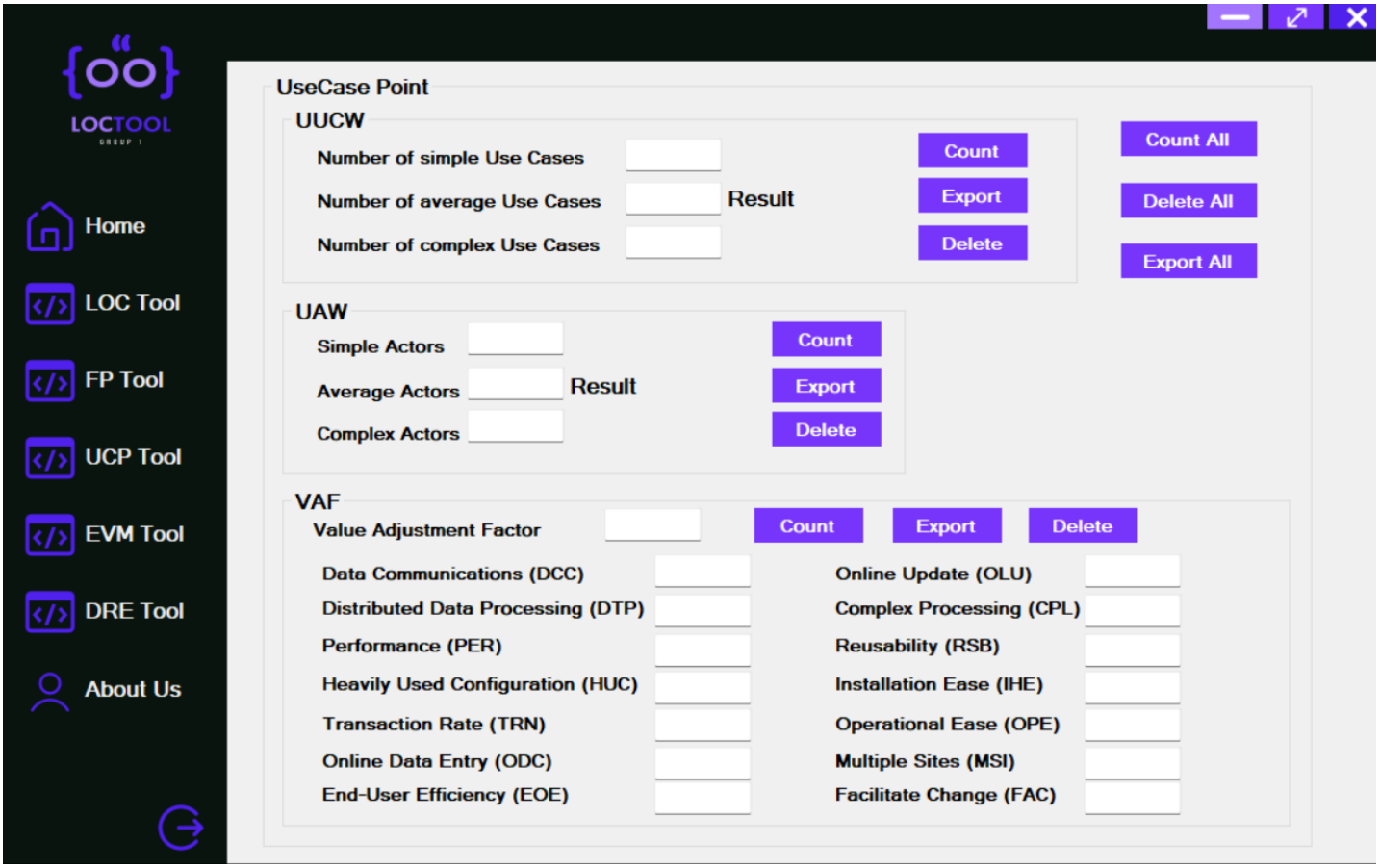

LOCTOOL
Home
LOC Tool
FP Tool
UCP Tool
EVM Tool
DRE Tool
About Us
Choose Folder or File
Folder
File
Select
Select File
Count
Result
File Name
LOC
Blank
Comment
Directory File
PTB2.cpp
44
7
6
D:\DTU\software-measure
Delete
Delete All
Export

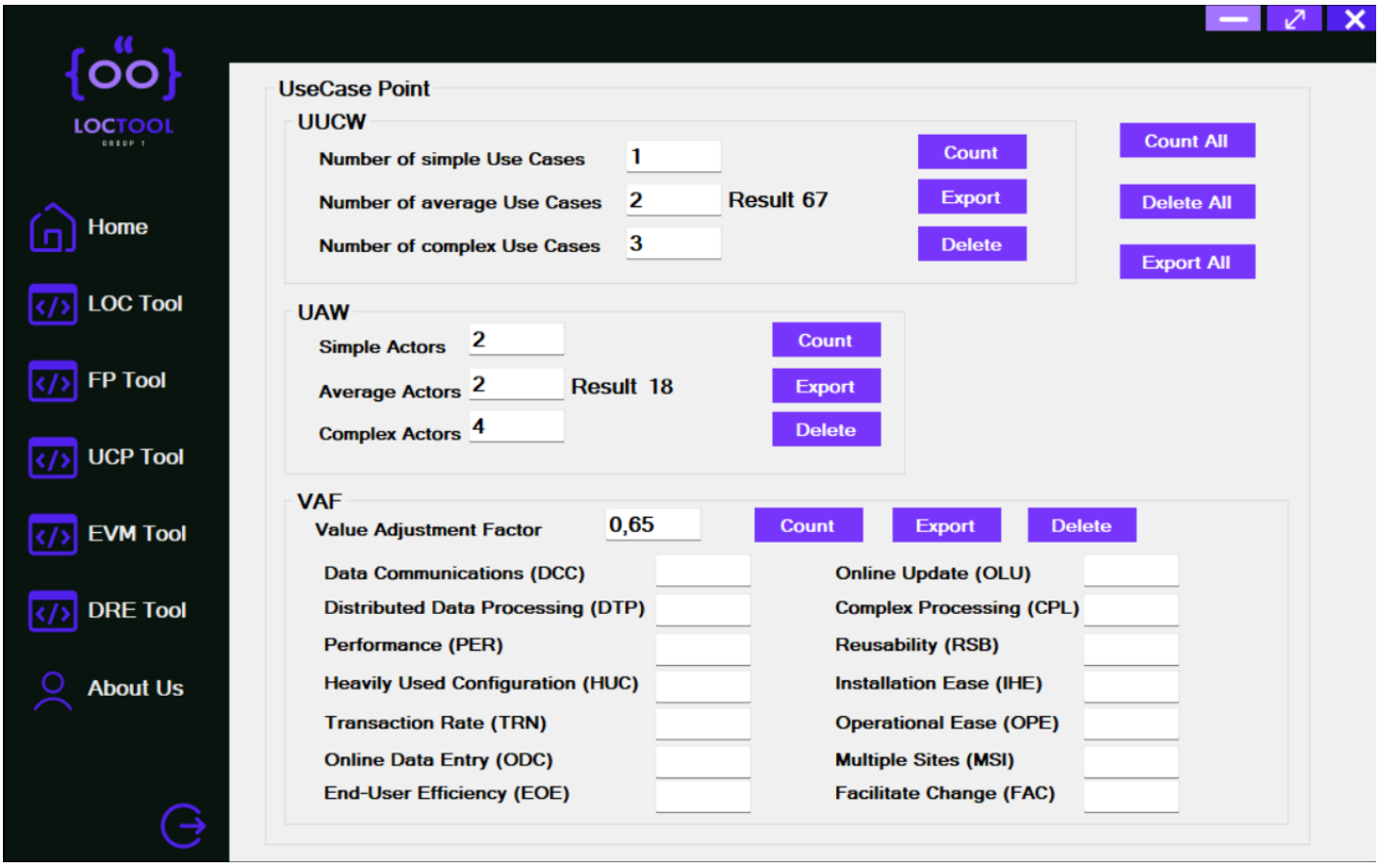

LOCTOOL
Home
LOC Tool
FP Tool
UCP Tool
EVM Tool
DRE Tool
About Us
Choose Folder or File
Folder
D:\DTU\software-measurements-analysis\New folder
Select
File
Select File
Count
Result
File Name
LOC
Blank
Comment
Directory File
PTB2.cpp
44
7
6
D:\DTU\software-measure
Delete
Delete All
Export

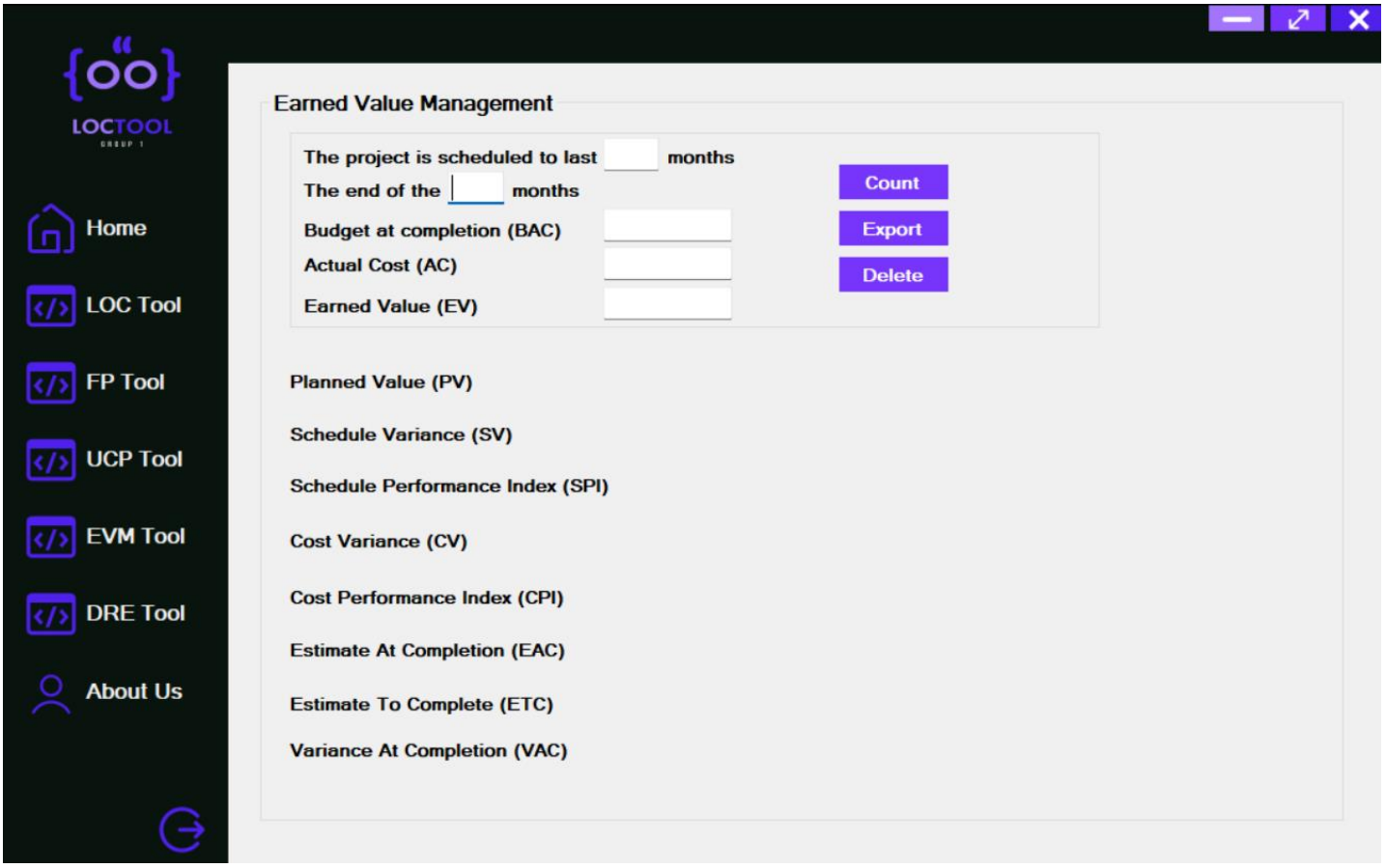

LOCTOOL
Home
LOC Tool
FP Tool
UCP Tool
EVM Tool
DRE Tool
About Us
Choose Folder or File
Folder
Select
File
Select File
Count
Result
File Name
LOC
Blank
Comment
Directory File
PTB2.cpp
44
7
6
D:\DTU\software-measure
findSize.cpp
3
0
0
D:\DTU\software-measure
PTB2.cpp
44
7
6
D:\DTU\software-measure
Delete
Delete All
Export

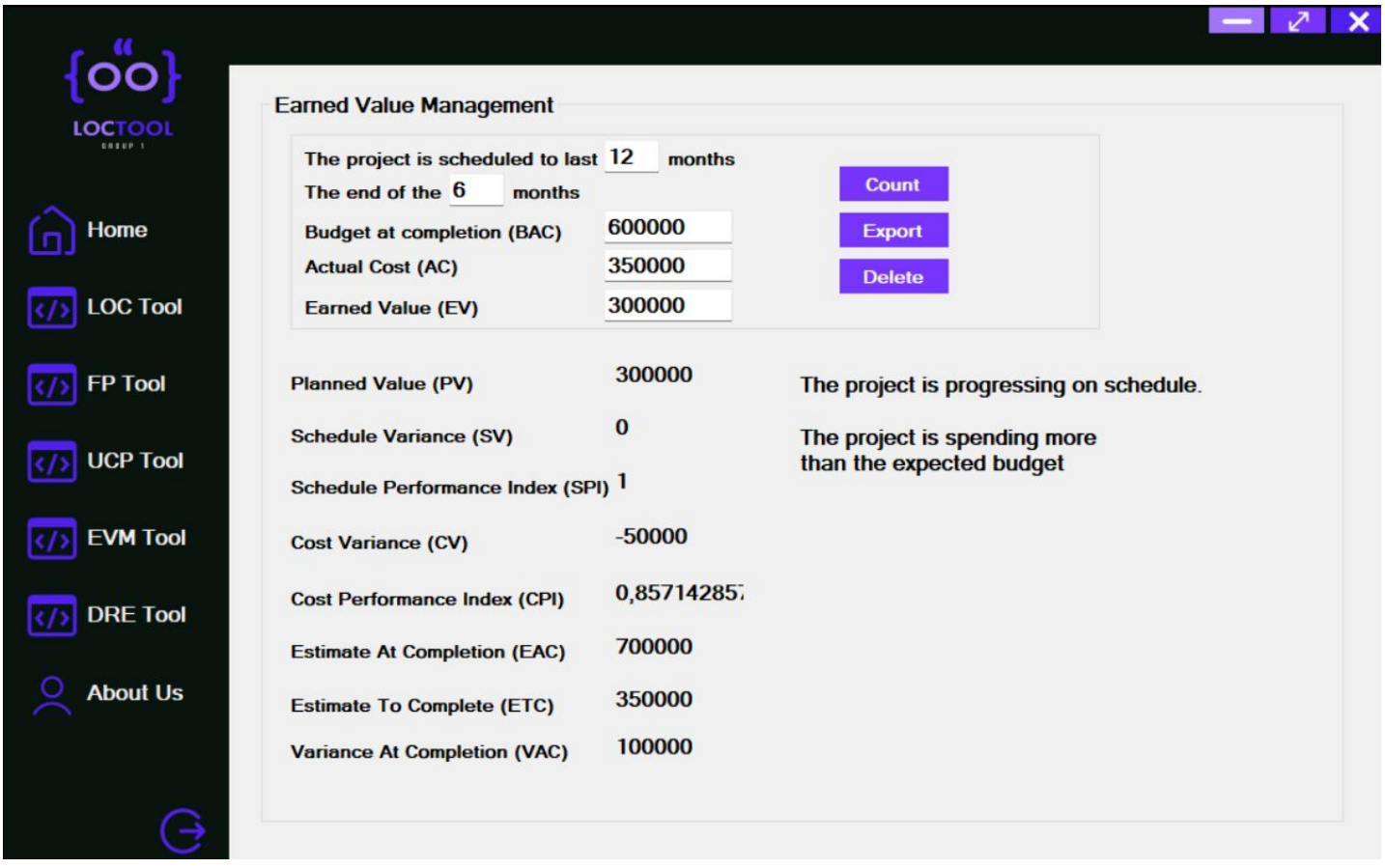

LOCTOOL
Home
LOC Tool
FP Tool
UCP Tool
EVM Tool
DRE Tool
About Us
Function Point
MEASUREMENT PARAMETER
MEASUREMENT PARAMETER
Simple
Average
Complex
Number of Use Input (EIs)
Number of Use Outputs (EOs)
Number of User Inquiries (EQs)
Number of Files (ILFs)
Number of External Interfaces (EIFs)
Result
Delete All
Count
Export


LOCTOOL
Home
LOC Tool
FP Tool
UCP Tool
EVM Tool
DRE Tool
About Us
Function Point
MEASUREMENT PARAMETER
MEASUREMENT PARAMETER
Simple
Average
Complex
Number of Use Input (EIs)
1
2
3
Number of Use Outputs (EOs)
1
2
Number of User Inquiries (EQs)
3
1
Number of Files (ILFs)
2
Number of External Interfaces (EIFs)
3
Result
133
Delete All
Count
Export


LOCTOOL
Home
LOC Tool
FP Tool
UCP Tool
EVM Tool
DRE Tool
About Us
UseCase Point
UUCW
Number of simple Use Cases
Number of average Use Cases
Number of complex Use Cases
Result
Count
Export
Delete
Count All
Delete All
Export All
UAW
Simple Actors
Average Actors
Complex Actors
Result
Count
Export
Delete
VAF
Value Adjustment Factor
Count
Export
Delete
Data Communications (DCC)
Distributed Data Processing (DTP)
Performance (PER)
Heavily Used Configuration (HUC)
Transaction Rate (TRN)
Online Data Entry (ODC)
End-User Efficiency (EOE)
Online Update (OLU)
Complex Processing (CPL)
Reusability (RSB)
Installation Ease (IHE)
Operational Ease (OPE)
Multiple Sites (MSI)
Facilitate Change (FAC)


LOCTOOL
Home
LOC Tool
FP Tool
UCP Tool
EVM Tool
DRE Tool
About Us
UseCase Point
UUCW
Number of simple Use Cases 1
Number of average Use Cases 2
Number of complex Use Cases 3
Result 67
Count
Export
Delete
Count All
Delete All
Export All
UAW
Simple Actors 2
Average Actors 2
Complex Actors 4
Result 18
Count
Export
Delete
VAF
Value Adjustment Factor 0,65
Count
Export
Delete
Data Communications (DCC)
Distributed Data Processing (DTP)
Performance (PER)
Heavily Used Configuration (HUC)
Transaction Rate (TRN)
Online Data Entry (ODC)
End-User Efficiency (EOE)
Online Update (OLU)
Complex Processing (CPL)
Reusability (RSB)
Installation Ease (IHE)
Operational Ease (OPE)
Multiple Sites (MSI)
Facilitate Change (FAC)


LOCTOOL
Home
LOC Tool
FP Tool
UCP Tool
EVM Tool
DRE Tool
About Us
Earned Value Management
The project is scheduled to last months
The end of the months
Budget at completion (BAC)
Actual Cost (AC)
Earned Value (EV)
Count
Export
Delete
Planned Value (PV)
Schedule Variance (SV)
Schedule Performance Index (SPI)
Cost Variance (CV)
Cost Performance Index (CPI)
Estimate At Completion (EAC)
Estimate To Complete (ETC)
Variance At Completion (VAC)


LOCTOOL
Home
LOC Tool
FP Tool
UCP Tool
EVM Tool
DRE Tool
About Us
Earned Value Management
The project is scheduled to last 12 months
The end of the 6 months
Budget at completion (BAC) 600000
Actual Cost (AC) 350000
Earned Value (EV) 300000
Count
Export
Delete
Planned Value (PV) 300000
Schedule Variance (SV) 0
Schedule Performance Index (SPI) 1
Cost Variance (CV) -50000
Cost Performance Index (CPI) 0,85714285
Estimate At Completion (EAC) 700000
Estimate To Complete (ETC) 350000
Variance At Completion (VAC) 100000
The project is progressing on schedule.
The project is spending more than the expected budget

VII. Demo:

https://www.geeksforgeeks.org/lines-of-code-loc-in-software-engineering/

https://www.geeksforgeeks.org/earned-value-management-evm/

https://www.tutorialspoint.com/estimation_techniques/estimation_techniques_use_case_points.htm

https://www.geeksforgeeks.org/software-engineering-functional-point-fp-analysis/