Presented at the 2019 ICEAA Professional Development & Training Workshop - www.iceaaonline.com

cobec CONSULTING

QUANTIFYING THE FUTURE

From Point A to Point Estimate: How Requirements Become Function Points

Dan French, CFPS, PMP, CSM
Principal Consultant
571-225-0380
dfrench@cobec.com

Carol Dekkers, CFPS (Fellow), PMP, CSM
President, Quality Plus Technologies, Inc
813-816-1329
Dekkers@qualiytplustech.com

FPA Historical Background

- Late 1960's → IBM identified an issue with the accuracy of SLOC based estimates for software development projects.
- IBM'er Allan Albrecht assigned to develop an alternative method to measure software size
- Function Points (FP) drafted as an alternative measure to SLOC
- Presented methodology for the first time in 1979 in his paper "Measuring Application Development Productivity"
- First formal counting guidelines published in 1984
- Counting Rules established by the International Function Point Users Group (IFPUG), founded in 1986
- Current version is 4.3.1, Released in January 2010
- The first International Standards Organization (ISO) Standard for software functional sizing (ISO/IEC 20926 Software Engineering - Function Point Counting Practices Manual)

What are Function Points?

- Function Points are a unit of software size measure
- Measure the work product of software development
- Work product is measured in terms of functionality from the user's perspective
- Functions points do not measure internal architecture, effort, or technological complexity of an application
- They are identified and calculated based on ***Functional*** User Requirements (FUR)

Benefits of Using Function Points

- Technology and language independent
- Consistent, repeatable, and verifiable
- Measures functionality the customer requests and receives
- Can use to derive metrics for cost, productivity, and quality
- Enables better management of project scope
- Allows for “apples to apples” comparisons between application types, platforms, organizations, and teams

Misconceptions about Function Points

- …can only be done late in the software development lifecycle (i.e. detail design) - FALSE
- Can't be used on some types of applications and platforms - FALSE
- FP are difficult – FALSE
- FP cannot be used once an application has been put into production - FALSE
- FP are good to measure individual productivity - FALSE
- Agile is not good for FP usage - FALSE

What Are Functional Requirements?

- a **functional requirement** defines a function of a system or its component. A function is described as a set of inputs, the behavior, and outputs.[1]
- They include:
  - Calculations
  - Data Processing
  - Reports
  - Screens
  - Interfaces

WHAT

How is Function Point Analysis Performed?

- Standard methodology: Counting Practices Manual (CPM) → owned by the International Function Point Users Group (IFPUG)
- Current version → CPM 4.1.3 (2010)
- FPA based on analyzing Function USER requirements and applying the rules in the CPM
- IFPUG certifications for FP:
  - Certified Function Point Specialist (CFPS)
  - Certified Function Point Practitioner (CFFP)

Function Point Counting: Components

- Five Functional Components: 3 Transactional and 2 Data
  - "Transaction" Functions
    1. External Inputs (EI) – e.g., Batch transaction file, input screen, control information
    2. External Outputs (EO) – e.g., Reports with calculations, output files with derived data
    3. External Inquiries (EQ) – e.g., On-line query screen, interface file with no calculations or derived data
  - Data Functions
    1. Internal Logical Files (ILF) – e.g., Application file, internal database
    2. External Interface File (EIF) – Reference

Steps in FP Counting Process

- Determine Type of Count (based on business need): New Development, Enhancement, Application
- Identify Counting Scope and Application Boundary
- Count Data Functions
- Count Transactional Functions
- Determine Unadjusted Function Point Count
- Determine Value Adjustment Factor*
- Calculate Adjusted Function Point Count*
  - *not part of the ISO standard definition of “Functional Size”

How Do Requirements Become Function Points?

- The analyst reviews the requirements and other supporting documentation to identify which are functional requirements that can be counted
- Each functional requirement (or set of functional requirements) is/are assessed to determine function type(s) and complexity, then entered into the counting tool
- Once all the functional requirements are counted, they are totaled to determine the function point count for the project

Determining the Transaction Complexity and Function Points

- Each transaction type has a Low, Average, or High complexity function point value
- Each transaction type's complexity is determined based on the type of transaction:
  - For EI, EO, & EQ → complexity based on the number of Data Fields and number of Logical files needed to complete the action
  - For ILF and EIF → complexity based on number of data fields and number of Record Element Types (RET)

The Complexity Matrices

- Each transaction and data type has their own complexity matrix

- Once the transaction's complexity is determined, The FP value for it is assigned

ILF and EIF Functional Complexity

		DETs		
		1 – 19	20 – 50	> 50
RETs	1	Low	Low	Average
	2 – 5	Low	Average	High
	> 5	Average	High	High

EI Functional Complexity

		DETs		
		1 – 4	5 – 15	> 15
FTRs	0 – 1	Low	Low	Average
	2	Low	Average	High
	> 2	Average	High	High

EO and EQ Functional Complexity

		DETs		
		1 – 5	6 – 19	> 19
FTRs	0 – 1	Low	Low	Average
	2 – 3	Low	Average	High
	> 3	Average	High	High
NOTE	An EQ has a minimum of 1 FTR.			

WEIGHTED COMPLEXITY OF FUNCTIONS

	Low	Average	High
ILF	7	10	15
EIF	5	7	10
EI	3	4	6
EO	4	5	7
EQ	3	4	6

cobec CONSULTING

What Is a Good Requirement?

- A well written requirement must be:
  - Testable/Verifiable
  - Clear and concise:
    - Single requirement
    - 30-50 words
    - Easy to read and understand
    - Unambiguous
    - No extraneous information (definitions, reasons for need, et al)
    - Not open ended or subjective

What Is a Good Requirement?

- A well written requirement must be:
  - Complete, containing all required information including fields, units, measures, duration, timing
  - Consistent and not conflict with other requirements
  - Unique
  - Use the same terminology used for all requirements
  - Traceable, include unique id that does not change or get reused
  - Viable
  - Necessary
  - Implementation agnostic

What Makes a Poorly Written Requirement?

- One that is vague and cannot be tested or evaluated
- Overly long and complicated
- Negative requirement
- Specifies how the developer should deliver the functionality
- Duplicate
- Missing

Examples of Poorly Written Requirements

- The system must be “easy to use”
- The GUI must be “visually appealing”
- The System must display analyzed safety data in dashboard form to include all relevant metrics associated with the critical safety items that are identified in Order XXXXX and are consistent with the current organizational standards, policies and requirements. The dashboard should contain all the data in a graphical format and be easy for users to read
- The System “must not allow” the deletion of flight data
- The System must produce Status Reports in an Microsoft Excel format

FP Counting Example

- System Req 167: The system must provide capability to manually enter data in cases of external sensor outages.
- This requirement implies* the manual maintenance of sensor data. Typically maintaining data requires add, update, delete and view capabilities
- So how is this requirement counted?

*Note: Once a count is completed. The FP analyst must review it with system experts (business analysts, developers, systems engineers, project managers, sponsors, end users et al to verify the accuracy of the count and validate/correct any assumptions

Counting Example (Transaction ID)

- Since the system must maintain sensor data, it must have an Internal Logical File (ILF) to store the data:

  Sensor Data ILF

- Add, Update and Delete are External Inputs (EI) who's primary intent is to maintain data on an ILF:

  Sensor Data-Add EI

  Sensor Data-Update EI

  Sensor Data-Delete EI

- As there is Update and Delete functionality, there must be the capability to view the data so there is a Sensor Data-View EQ

Counting Example (complexity)

- We determined in our discussion with the SMEs that the only FTR for these transactional functions is the Sensor Data ILF, so there is one FTR for each transaction.
- As the requirements are initial requirements and are at a high level, the specific fields have not been identified. However, the SMEs confirmed that the they expect the DETS to be greater than 20 with the exception of the Delete EI

Counting Example (complexity)

- Therefore, the View EQ, Add EI and Update EI are 21DETS/1FTR and the Delete-EI is 3 DETS/1FTR. The ILF is 21DETS/2 RET (weather and aircraft)

- Add & Update EIs are Ave complexity = 4FP
- View EQ is also Ave=4FP
- Delete EI is Low=3FP
- ILF is Ave = 10 FP

ILF and EIF Functional Complexity

		DETs		
		1 – 19	20 – 50	> 50
RETs	1	Low	Low	Average
	2 – 5	Low	Average	High
	> 5	Average	High	High

EI Functional Complexity

		DETs		
		1 – 4	5 – 15	> 15
FTRs	0 – 1	Low	Low	Average
	2	Low	Average	High
	> 2	Average	High	High

EO and EQ Functional Complexity

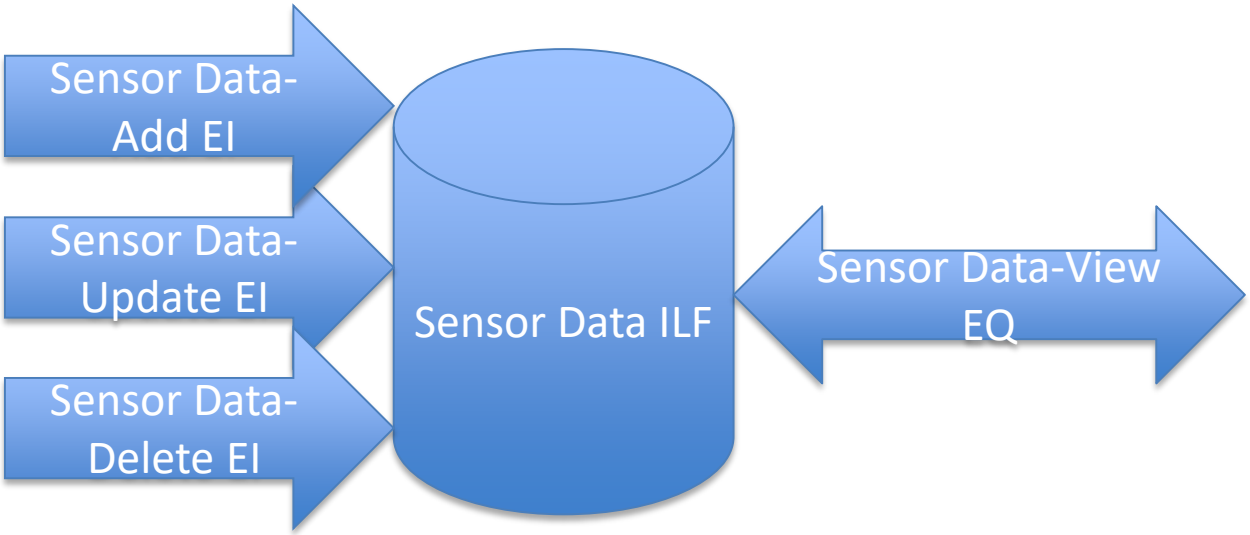
		DETs		
		1 – 5	6 – 19	> 19
FTRs	0 – 1	Low	Low	Average
	2 – 3	Low	Average	High
	> 3	Average	High	High
NOTE	An EQ has a minimum of 1 FTR.			

WEIGHTED COMPLEXITY OF FUNCTIONS

	Low	Average	High
ILF	7	10	15
EIF	5	7	10
EI	3	4	6
EO	4	5	7
EQ	3	4	6

cobec CONSULTING

Counting Example Diagram



Function Point Use at the FAA

- The FP size estimates were then used as input into the SEER-SEM parametric software estimation tool
- Results are then incorporated into the LCCE cost models under WBS 3.1.x and 4.1.x for maintenance
- Adoption of FP as a software sizing metric has been slow but is increasing at the agency

The Future of Function Points

- Software estimation is only just one of the many uses of function points and organization should want to consider expanding usage to include:
  - Application Baselining
  - FP based software metrics
  - Analogous estimating

Cautionary Advice

- Make sure that all stakeholders understand the key concepts of function points and why they are preferable to SLOC-based estimates
- Use experienced function point counters, preferably CFPS or those ready for the exam during baselining phase
- When training function point analysts, be sure to follow the recommended IFPUG training guidance
  - Have them take a IFPUG certified training class
  - Observer experienced counter
  - Count under guidance of experienced counter
  - Perform counts validated by experienced counter
  - Prepare for exam

In Conclusion

- Function Points are not a silver bullet, but will increase the accuracy and reduce the risk of software estimates
- Proper training of Function Point analysts is critical for the successful use of function points
- So long as the organizations continue to use SLOC for software estimation, there significantly greater programmatic risk and increased chance of project cost overruns, schedule delays and potential for cancellation
- Good communication, requirements elucidation and management are critical not only to the development of a defensible software estimate, but greatly increases the likelihood of delivering a successful project on time and on budget

Additional Information

Why are Some Software Estimates "Wrong"

- Expert Judgment estimates tend to be overly optimistic
- Estimates by Analogy don't use appropriate comparative project(s) and are not calibrated properly
- SLOC based estimates are essentially just guesstimates until code is actually written
- Sized based estimation using function points can help avoid these issues

Poor estimates result in misallocations of resources, over-stressed team members, unrealistic stakeholder expectations and potential contractual issues with vendors

So How Can We Address These Challenges Effectively?

- Many of these challenges can be addressed by developing or improving the processes for:
  - Requirements
  - Change Control/Scope Management
  - Quality Assurance
  - Estimation