2001ICT / 7621ICT - Project Management

Function Point Counting Basics

Function point counting is a technique used to estimate a level of complexity for a product, based on the functionality that it will deliver. It does this by counting the individual elements involved in each function. The technique uses a specific language to define the elements that are being counted. The key terms are:

|  |  |  |
| --- | --- | --- |
| **Term** | **Acronym** | **Definitions** |
| Function Types (functions that the product performs): | | |
| Internal Logical Files | ILF | Files of information that are stored and modified within the system (ie, a personnel file for a payroll system) |
| External Interface Files | EIF | External files of information that are used by the system (ie, tax information from the ATO that the payroll system might use) |
| External Inputs | EI | Information that is provided to the system (eg add, edit and delete a person) |
| External Output | EO | Complex information from the system (the function involves some manipulation of stored data) |
| External Inquiry | EQ | Simple information from the system (the function simply displays stored information) |
| Element Types (individual files or pieces of data that the functions use) | | |
| Data Element Types | DET | Non-repeated fields or attributes |
| Record Element Types | RET | Groups of data within an ILF or EIF |
| File Types Referenced | FTR | The ILF or EIF that is used or maintained by the function |

To estimate you need to classify each type of function that you are developing, and then count the appropriate data elements according to function type. Function Point tables are used to convert these counts into complexity, and the complexity into a function point number.

## Applying the Technique

1. First, look at all your functions and classify each one as either ILF, EIF, EI, EO, or EQ.
2. For each ILF or EIF, count RETs and DETS.
3. For each EI, EO, or EQ, count FTRs and DETs
4. Match these counts to the appropriate table (following) to determine complexity:

**EIF or ILF**

|  |  |  |  |
| --- | --- | --- | --- |
| RET | 1-19 DET | 20-50 DET | 51+ DET |
| 1 | Low | Low | Avg |
| 2-5 | Low | Avg | High |
| 6+ | Avg | High | High |

**EI**

|  |  |  |  |
| --- | --- | --- | --- |
| FTR | 1-4 DET | 5-15 DET | 16+ DET |
| 0-1 | Low | Low | Avg |
| 2 | Low | Avg | High |
| 3+ | Avg | High | High |

**EO/EQ (These share the same values).**

|  |  |  |  |
| --- | --- | --- | --- |
| FTR | 1-5 DET | 6-19 DET | 20+ DET |
| 0-1 | Low | Low | Avg |
| 2-3 | Low | Avg | High |
| 4+ | Avg | High | High |

1. Calculate the final FP count for the system using the following table to give a numerical value (number of function points) to each complexity level

.

|  |  |  |  |
| --- | --- | --- | --- |
| FP Type | Low | Avg | High |
| EI | 3 | 4 | 6 |
| EO | 4 | 5 | 7 |
| EQ | 3 | 4 | 6 |
| ILF | 7 | 10 | 15 |
| EIF | 5 | 7 | 10 |

As an example:

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Type** | **Complexity** | **FP** |
| The system stores information (35 pieces) on employees and their dependents |  | \_\_ RETs (Employees & Dependents)  \_\_ DETs (information)  = Complexity |  |

## System Details

Detailed Requirements Table (including columns for FP counting)

|  |  |  |  |
| --- | --- | --- | --- |
| Functionality | Type | Complexity | FP |
| The system will store information relating to client accounts, including ‘s’ number, parking permit status, payment status, home campus, and ‘valid until’ date |  | RET/FTRs - \_\_\_  DETs - \_\_\_  =\_\_\_\_\_\_\_\_\_\_\_complexity |  |
| The system uses client information (16 attributes) from the staff/student system |  | \_\_\_\_s - \_\_\_  DETs - \_\_\_  =\_\_\_\_\_\_\_\_\_\_\_complexity |  |
| The system will provide an Administrator Interface that will allow creation, editing, and deletion of client accounts |  | \_\_\_\_s - \_\_\_  DETs - \_\_\_  =\_\_\_\_\_\_\_\_\_\_\_complexity |  |
| The system will provide a user interface that will allow the user to view their account information, including usage data, personal information, and permit information |  | \_\_\_\_s - \_\_\_  DETs - \_\_\_  =\_\_\_\_\_\_\_\_\_\_\_complexity |  |
| The system will store real time information on occupied and unoccupied car parks |  | \_\_\_\_s - \_\_\_  DETs - \_\_\_  =\_\_\_\_\_\_\_\_\_\_\_complexity |  |
| The system will display real time information on occupied and unoccupied car park locations and numbers |  | \_\_\_\_s - \_\_\_  DETs - \_\_\_  =\_\_\_\_\_\_\_\_\_\_\_complexity |  |
| The system will provide instructions to the user for finding an available car park |  | \_\_\_\_s - \_\_\_  DETs - \_\_\_  =\_\_\_\_\_\_\_\_\_\_\_complexity |  |
|  |  | **TOTAL** |  |

# Agile Story Point Counting

There are a range of techniques for counting story points for Agile.

The first technique is calculating complexity based on aspects including time, effort, risk, and resources required for each user story. This is done by allocating a pre-defined value for each aspect – for example:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **User Story** | **Time** | **Effort** | **Risk** | **Resources** | **Total** |
| US07 | As a consultant I can store information about firm employees and their dependents so that the company has an accurate HR record | 2 | 3 | 3 | 2 | 10 |

For this story the team defined a ‘2’ for time as between 12 and 35 hours work, a ‘3’ for effort as the whole team required for the story, a ‘3’ for risk as is a high risk story, and a ‘2’ for resources as a medium level of resource requirements.

The second technique is looking at the effort required for a user story specifically as it applies to your team and developing metrics (or sizes) for this effort. For this you must understand your team’s capability (work level, productivity, skills) as well as the metric you choose (how do you define ‘size’? Examples include t-shirt sizing, simple numerical rankings, and so on). Each user story is then assigned a ‘size’, resulting in an overall size for the project.

## Applying the Technique

1. First choose a user story that you consider to be the simplest and that everyone can understand.
2. Decide on a metric that your team will use for sizing.
3. Individually, estimate effort for the story by applying your metric to each complexity element listed above and determining an overall value for the user story.
4. Compare each individual value and form a consensus.
5. Based on this common understanding, estimate the remainder of the user stories as a group.
6. What is your total value for all of the stories?

## Optional Exercise

You can use this to complete a practice story point count. These are the same requirements used in the function point count, converted into user stories. How does your total here compare to the function point total?

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **User Story** | **Time** | **Effort** | **Risk** | **Resources** | **Total** |
| U1 | As a system, I want to store information relating to client accounts, including ‘s’ number, parking permit status, payment status, home campus, and ‘valid until’ date, so that I have a record of all client parking details |  |  |  |  |  |
| U2 | As a system I want to use client information from the staff/student system, so that I can access all relevant client details |  |  |  |  |  |
| U3 | As an Administrator, I want to be able to create, edit, and delete client accounts, so that the system contains current and correct information |  |  |  |  |  |
| U4 | As a client, I want to view my account information, including usage data, personal information, and permit information, so that I can monitor my account |  |  |  |  |  |
| U5 | As a system, I want to store real time information on occupied and unoccupied car parks, so that I have accurate parking situation at any time |  |  |  |  |  |
| U6 | As a client, I want to see real time information on occupied and unoccupied car park locations and numbers, so that I know the best location for me to park |  |  |  |  |  |
| **Total** | | | | | |  |