**ASSIGNMENT 3**

**Q) Find product size estimation based on FP Method for project software.**

**Ans :**

**Use Case Points:**

The number of use case points in a project is a function of the following:

• the number and complexity of the use cases in the system

• the number and complexity of the actors on the system

• various non-functional requirements (such as portability, performance,

maintainability) that are not written as use cases

• the environment in which the project will be developed (such as the language, the

team’s motivation, and so on)

**Use Case Specifications :**

**Use Case Name :** Purchase Food Grains

**Actors :** Registered user(primary)

FPS shop seller (secondary)

**Other Stakeholders:** Godown (food grains distributor)

LINK (communication between other FPS)

**Summary Description :** Allows registered customer to purchase food grains from associated FPS.

**Priority** **:** Must Have

**Risk Level :** High

**Status :** Fully Detailed

**Pre Condition :**  Customer should registered.

Customer should have Ration Card

Cutomer should have smart card.

**Post Condition:** The customer has purchased the goods from FPS and paid the money to shopkeeper. Shopkeeper updates the records of goods in customer login ,customer receives message of transactions.

**Basic Path:**

1.The customer goes to Fair Price Shop.

2.The customer asks for ration goods to FPS seller.

3.The FPS seller takes out goods for customer according customer requirements.

4.FPS seller asks customer to authenticate using fingerprint.

5.After authentication seller gives goods to customer.

6.The customer enters smart card into RFID.

7.The RFID verifies that card is valid.

8.The RFID requests a pincode.

9.The customer enters their pincode.

10.The RFID validates customer against pin code.

11.FPS seller updates the customer good records.

12.Customer pays money to the FPS seller.

13.FPS seller gives receipt using Handheld electronic machine to customer.

13.Customer receives a transaction message.

14.Customer takes their goods.

**Alternative Paths: (identified only)**

**3a. Goods/materials are not available.**

1.It indicates that goods are not available in FPS now.

2.Shopkeeper asks for another material.

3.Rejoin the basic path at step3.

**4a. Fingerprint authentication fails.**

1.Wrong finger has pressed for authentication.

2. Unauthorized person is authenticating.

3.Rejoin the basic path at step 4.

**7a.invalid card**

1.The RFID indicates that it is wrong type of card

2.The RFID asks customer to insert another card.

3.Rejoin the basic path at step7

**7b.card upside down**

1.The RFID indicates that the card is upside down

2.RFID asks customer to insert the card again.

3.Rejoin the basic path at step 7

**10a.Stolen card**

1.RFID matches the card number to stolen card.

2.The RFID runs process Stolen card use case.

3.End use case.

**10b.invalid pin**

1.RFID indicates that wrong pin has entered.

2.It permitted no of tries is not exceeded, RFID asks customer to enter their pin again and rejoins the basic path at step 4.

**Business Rules:**

**B1: Limited goods**

Not more than permiited limit of ration goods can be taken by customer

**B2:service options**

Current service option are: Enquire details of goods,Change pin, pay money,purchase goods

**B3:Format of Pin**

It is a 4 digit number which is stored on a chip in the card.

**Non-Functional Requirements:**

**NF1: Use of any finger for fingerprint**

The fingerprint used for authentication should be the finger given during issuing the card.

**NF2:Security for Pin entry**

The pin entered by the customer should be encoded as \* on screen.

**NF3:Time for complete transaction of goods**

The transaction should typically take 5 minutes to complete.

**Project Estimation :**

|  |  |  |
| --- | --- | --- |
| Use case Complexity | No.of transactions | Weight |
| Simple | 3 or fewer | 5 |
| Average | 4 to 7 | 10 |
| Complex | More than 7 | 15 |

Table :1 Use case weights depending on number of transactions.

|  |  |  |  |
| --- | --- | --- | --- |
| Use case complexity | Weight | Number of use case | product |
| Simple | 5 | 21 | 105 |
| Average | 10 | 15 | 150 |
| Complex | 15 | 8 | 120 |
| Total |  |  | 375 |

Table 2: calculating Unadjusted use case weight (UUCW )for a project.

|  |  |  |
| --- | --- | --- |
| Actor Type | Example | Weight |
| Simple | Another system through an API 1 | 1 |
| Average | Another system through a protocol  A person through text-based interface | 2 |
| Complex | A person through a graphical user interface | 3 |

Table 3 :Actor complexity

|  |  |  |  |
| --- | --- | --- | --- |
| Actor type | Weight | No of actors | Product |
| Simple | 1 | 8 | 8 |
| Average | 2 | 6 | 12 |
| Complex | 3 | 5 | 15 |
| Total |  |  | 35 |

Table 4 : Calculating unadjusted actor weight

**Unadjusted Use Case Points:**

At this point we have the two values that represent the size of the system to be built.

Combining the Unadjusted Use Case Weight (UUCW) and the Unadjusted Actor Weight

(UAW) gives the unadjusted size of the overall system. This is referred to as Unadjusted

Use Case Points (UUCP) and is determined by this equation:

**UUCP =UUCW +UAW**

UUCF for E- Khadya Suraksha project:

UUCP= 375 + 35 = 410

3 D **Function Points:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Measurement parameter** | **count** |  | **Simple** | **Average** | **complex** | **Product** |
| No.of external inputs (EI) | 30 | \* | 3 | 4 | 6 | 120 |
| No of external outputs(EO) | 18 | \* | 4 | 5 | 7 | 90 |
| No of external inquiries(EQ) | 8 | \* | 3 | 4 | 6 | 48 |
| No.of internal files(ILF) | 6 | \* | 7 | 10 | 15 | 48 |
| No.of external interfaces(EIF) | 4 | \* | 5 | 7 | 10 | 28 |
| Total |  |  |  |  |  | 334 |

***Adjusting For Technical Complexity***

The total effort to develop a system is influenced by factors beyond the collection of use cases that describe the functionality of the intended system. A distributed system will take more effort to develop than a nondistributed system. Similarly, a system with difficult to meet performance objectives will take more effort than one with easily met performance objectives. The impact on use case points of the technical complexity of a project is captured by assessing the project on each of thirteen factors, as shown in Table5. Many of these factors represent the impact of a project’s nonfunctional requirements on the effort to complete the project. The project is assessed and rated from 0 (irrelevant) to 5 (very important) for each of these factors.

**Factor Description Weight**

|  |  |  |
| --- | --- | --- |
| T1 | Distributed system | 2 |
| T2 | Performance objective | 2 |
| T3 | End-user efficiency | 1 |
| T4 | Complex processing | 1 |
| T5 | Reusable code | 1 |
| T6 | Easy to install | 0.5 |
| T7 | Easy to use | 0.5 |
| T8 | portable | 2 |
| T9 | Easy to change | 1 |
| T10 | Concurrent use | 1 |
| T11 | Security | 1 |
| T12 | Access for third parties | 1 |
| T13 | Training needs | 1 |
| **Total** |  | **15** |

Table 5. The weight of each factor impacting technical complexity

|  |  |  |  |
| --- | --- | --- | --- |
| Factor | Weight | Assessment | impact |
| Distributed system | 2 | 3 | 6 |
| Performance objective | 2 | 3 | 6 |
| End-user efficiency | 1 | 3 | 3 |
| Complex processing | 1 | 2 | 2 |
| Reusable code | 1 | 0 | 0 |
| Easy to install | 0.5 | 0 | 0 |
| Easy to use | 0.5 | 4 | 2 |
| Portable | 2 | 2 | 4 |
| Easy to change | 1 | 5 | 5 |
| Concurrent use | 1 | 5 | 5 |
| Security | 1 | 5 | 5 |
| Access for third parties | 1 | 3 | 3 |
| Training needs | 1 | 2 | 2 |
| Total Factor |  |  | 38 |

The weighted assessments for these twelve individual factors are next summed into what is called the *TFactor*. The TFactor is then used to calculate the*Technical Complexity Factor*, TCF, as follows:

***TCF* = 0.65 + (0.01\**TFactor*)**

**For E-Khadya suraksha :**

***TCF* = 0.65 + (0.01\**38*)**

**= 0.65+0.38**

**=1.03**

**Function point = count total\*TCF**

**=334\*1.03**

**=344.02**

***Adjusting For Environmental Complexity***

Environmental factors also affect the size of a project. The motivation level of the team,their experience with the application, and other factors affect the calculation of use case points. Table 7 shows the eight environmental factors Karner’s formulas consider for each project.

|  |  |  |
| --- | --- | --- |
| **Factor** | **Description** | **Weight** |
| E1 | Familiar with the development process | 1.5 |
| E2 | Application experience | 0.5 |
| E3 | Object-Oriented experience | 1 |
| E4 | Lead analyst capability | 0.5 |
| E5 | Motivation | 1 |
| E6 | Stable requirements | 2 |
| E7 | Part time staff | -1 |
| E8 | Difficult programming Languages | -1 |

**Table 7 : Environmental factors and their weight**

The EFactor is then used to calculate the *Environment Factor*, EF, as

follows:

***EF* =1.4 + (-0.03\* *EFactor*)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Factor** | **weight** | **Assessment** | **Impact** |
| Familiar with the development process | 1.5 | 3 | 4.5 |
| Application experience | 0.5 | 4 | 2 |
| Object-Oriented experience | 1 | 4 | 4 |
| Lead analyst capability | 0.5 | 4 | 2 |
| Motivation | 1 | 5 | 5 |
| Stable requirements | 2 | 1 | 2 |
| Part time staff | -1 | 0 | 0 |
| Difficult programming Languages | -1 | 2 | -2 |
| Total Factor |  |  | 17.5 |

**Table 8 : Environmental Factor calculations**

**EF =1.4 + (-0.03\* EFactor)**

**=1.4 + (-0.03\*17.5)**

**=0.89**

**Putting It All Together**

To come up with our final *Use Case Point* (UCP) total, Karner’s formula takes the

Unadjusted Use Case Points (UUCP, the sum of the Unadjusted Use Case Weight and the Unadjusted Actor Weight) and adjusts it by the Technical Complexity Factor (TCF) and the Environmental Factor (EF). This is done with the following formula:

***UCP* *UUCW* *TCF* *EF***

**UCP=410\*0.98\*0.89**

**=357.602**

**Deviation :**

* The project has 375 use case points
* The team will average between 20 and 28 hours per use case point
* Iterations will be two weeks long
* A total of ten developers (programmers, testers, DBAs, designers, etc.) will work on this project

**Project will take 7500 -10500 hours**

We estimate that each developer will spend about 30 hours per week on project tasks. The rest of their time will be sucked up by corporate overhead—answering email, attending meetings, and so on. With ten developers, this means the team will make10 \* 30 = 300 hours per week or 600 hours of progress per iteration.

Dividing 7500 hours by 600 hours and rounding up indicates that the overall project might take 13 two-week iterations. Dividing 10500 by 600 hours and rounding up indicates that it might take 18 two-week iterations. **Our estimate is then that this project will take between 13 and 18 two-week iterations (26 to 36 weeks).**