# A Case Study Approach to Measure the Function Points from the Points of Relationships of UML

Dr.GSVP Raju
CS&ST Dept,Andhra University
Vizag, Andhra Pradesh

K.Koteswara Rao
CSE Dept,GMRIT
Rajam,Andhra Pradesh

M Sumender Roy
CSE Dept,GIET
Rajahmunndry, Andhra Pradesh

## ABSTRACT

The project manager plays the crucial role in the success or failure of the project. The primary success factor of the project is accurate forecast of the effort estimations. Unfortunately accurate forecast of effort estimations stems from the matured organizations,, others owing to lack of history databases. Estimations can be done in two ways, one is by estimating the size of the project and the other is by total functionality of the project. The primary one is suitable only when the software experts can able to claim the size of the software project in the efficient way, but it is not achieved and only 10 percent of the projects are delivered in on time. Most of the software experts suggesting that instead of estimating the size of the project it is better to go for estimating the total functionality of the project. As a result Function point concept is introduced. This paper explains about how the Function points can be measured from the points of relationships of UML with a case study.

**Keywords:** Function Point,relationship,UML

## 1. FUNCTION POINTS

Most of software project management experts claim that length is misleading when trying to size the software product. A better way to generate effort estimations and duration estimations from the product is to estimate the functionality rather than the physical size. Function Points are the one of the major technique to the major functionality of the system. This paper explains how the function points can be measured from the relationships of UML.

Function Points were originally introduced by Allan Albrecht over 20 years ago. At the early stages function points can be calculated from UAFP and TCF which includes the five components which are listed below.

. External Inputs (EI)
. External Outputs (EO)
. External Inquiries (EQ)
. Internal Files (IF). External Files (EF)

Here file means a user identifiable group of data, thus not necessary a traditional physical file implemented in computer system. After EI, EO, EF, IF, EQ is weighted then unadjusted function point count can be calculated by summing the all individual counts. After that Total complexity factor can be calculated from the 15 cost drivers based on technical and quality requirements. The multi placation of the UAFP and TCF is called FP.

## 2. CONCEPTUAL MODEL OF UML

To understand the UML it is necessary to form the conceptual model of the UML which includes the learning of 3 concepts

. Building Blocks
. Rules
. Common Mechanisms

Building blocks are the necessary elements which includes

. Things
. Relationships
. Diagrams

Things are nothing but abstractions which are first class citizens can be classified

Structural (Class, Use Case, Interface, Active Class, Component, Simple Collaboration and Deployment)

Behavioral Things (Interaction and State Machine)
. Grouping Things (Package)
. Annotational Things (Note)

Relationship is a semantic link between any two elements of the above things, which can be classified into Association, Dependency, Generalization and Realization. Diagrams are the graphical representation of elements, which are classified into 2 types.

1. Structural Diagrams (Static Aspects)
Class Diagram
Object Diagram
Component Diagram
Deployment Diagram
2. Behavioral Diagrams (Dynamic Aspects)
• Use Case Diagram
• Activity Diagram
• Interaction Diagram
Sequence Diagram
Collaborations Diagram
• State Chart Diagram

### 2.1Unified Library Application System

In ULAS member can login into system that means actor is providing the data to system so it can be counted as External Input. Actor can visualize the reports in Search and Browse phase it can be counted as External Output. The ULAS is maintaining the catalog to authenticate and to respond on operation, it can be treated as Internal File. The actor can have the availability to see the reservation details and the availability details so it can be treated as External File. When member asked for issue the book then the librarian can checks reservation, dues etc. it can be treated as External Inquiries. It can be enough for a rough estimation of effort to only count the number of transaction types and get an early estimate of the functional size of the designed system. To get more accurate figures, a base use case diagram is not enough. Explanations on different use case on natural language provide more information on attributes of each process. If these are available, the number of external and internal files can be calculated more easily and also the complexity factor of each use case can be extracted from the texts. The extra information can also become available as the design proceeds .so the results of function point calculations can be updated and adjusted when possible. It is a safe to say that the quality of function point calculation based on use cases is highly dependent on the quality of the use cases. The more detailed description of the use cases are the more accurate function point analysis. Frequently confronted problems with the use case include also the fact that one use case can contain several transactions or one transaction can consist of several use cases

## 2.2 Use case diagram:-

It is a brief functionality of the system. It shows the relationship between actors and use cases. Use case diagram explains about how the system is going to interact with the out side environment. The components are actor, use case, relationship and package. Actor is some one or something that is interacting with the system. Use cases are nothing but scenarios of the system. If the abstraction level of the diagram is more enough an estimation of the function point can be derived from it. Different use cases must be consistent and describe the behavior of the system. Here first defining the boundary of system of the application which is essential to find whether the file is internal or external. Here boundary exists between actor and the system. Transaction type can also be identified easily if the use cases are defined properly.

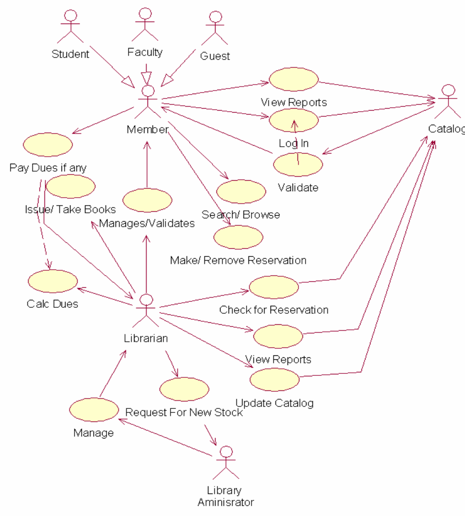


**Fig:2.1** Unified Library Application System

## 2.3 CLASS DIAGRAM

Class diagram shows structure of the software system. The class diagram shows a set of classes, interfaces and relationships. The components are:

a) Class, interface, package, simple collaboration

b) Relationship: those are Association, Aggregation ,Generalization ,Composition Dependency

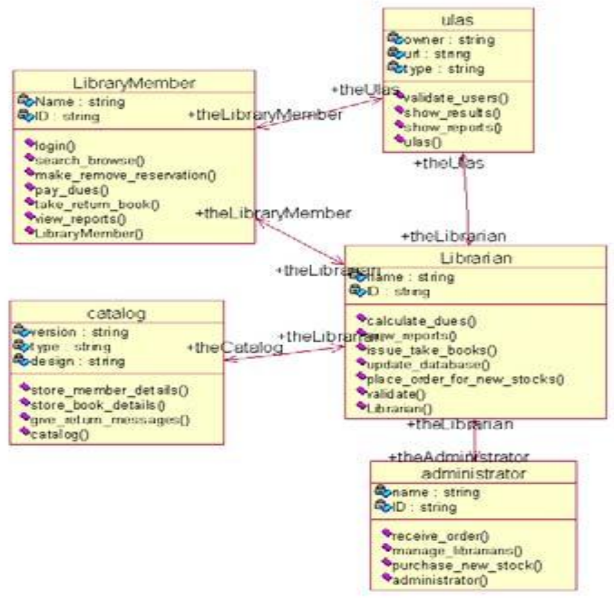


Fig:2.2Unified Library Application System

## 3. FROM THE POINTS OF RELATIONSHIPS OF UML:

The relationship can be defined as semantic link between any two things. In general the UML relationships can be classified into four types

1.Generalization

2. Association

3.Dependency

4.Realization

This section mainly emphasize how the function points can be measured from the points of relationships.

**3.1 Generalization :** Generalization is a specialization and it is nothing but Inheritance which can be represented by using solid line with Triangle head arrow. This relationship can be applied among.

a) actors b) Use cases c) Classes according to the context.

a) Among the Actors: Actor is some one or some thing who is interacting with system.

| Component | Category | Weight |
|---|---|---|
| Actor | simple | 1 |
| | Average | 2 |
| | Complex | 3 |

According to the number of actors involved, relationship may be simple /average/complex and weighting also can be given.

b) Use cases: Use cases are nothing but scenarios of the system.

Generalization may applied between Use cases also.

| Component | Category | Weight |
|---|---|---|
| Use case | Simple | 5 |
| | Average | 10 |
| | Complex | 15 |

Here by considering the number of use cases the relationship may be considered simple/ average/complex and corresponding weighting factor can be assigned.

c) Among the Classes: Class is a collection of similar objects which share the common name , behavior, structure and properties.

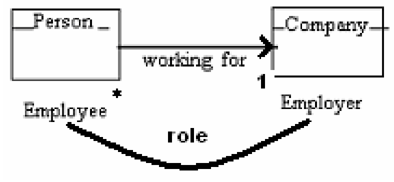
| Component | category | weight |
|---|---|---|
| class | Simple | 4 |
| | Average | 8 |
| | Complex | 12 |

By considering the context the relationship is considered as above categories which are listed in the table and weights are assigned. Now it is the time to calculate the number of generalization relationships

$$NO\backslash GR= \sum_{i=1}^{n} (NOGAA+NOGAU+NOGAC)$$
$$= X \text{ Function Points. (Assume)}$$

**3.2 Association:** Association is a structural relationship , In UML association is used to represent interaction among the things. Association may be existed between the use case and actor, between Classes also.

Association has four properties a) Name b) Direction c) Roles d) Multiplicity



a) Between actor and Use case:

| Component | Category | Weight |
|---|---|---|
| Actor &use case | Simple | 2 |
| | Complex | 4 |

By considering the context the relationship can be categorized and weighted as shown in table

b) Among the classes :

| Component | Category | Weight |
|---|---|---|
| Classes | simple(unidirectional) | 2 |
| | complex(Bi directional) | 4 |

Here also relationship category and weightage can be identified.

Now the number of association relationships can be calculated.

$$NOAR = \sum_{i=1}^{n} (NOARBAU+(NOARBC)$$
$$= Y \text{ Function Points. (Assume)}$$

**3.3 Aggregation and Composition (The forms of association):**

Aggregation and composition both are Stems from the association. These two are having the some specialized features in addition to association. Aggregation is a part of relationship .If we want to express the relationship between the part and its whole the use of aggregation is best suited . It can be represented by a solid line with hallow diamond.

Composition is consist of relationship, where it is best suited to relate whole to its part. It can be represented by a solid line with filled diamond.

| Component | category | weight |
|---|---|---|
| Class | Simple | 5/8 |
| | Complex | 10/16 |

$$NOAR = \sum_{i=1}^{n} (NOARC) = W \text{ Function Points}$$

$$NOCR = \sum_{i=1}^{n} (NOCRC) = Q \text{ Function Points (Assume)}$$

**3.4 Dependency Relationship :** Dependency is a using relationship which can be represented by using dotted line with arrow. If there is dependency relationship between A and B that means if any thing changes in A that will automatically reflect in B. Dependency is existed between use cases and classes only.

| Component | category | weight |
|---|---|---|
| Use cases | Simple | 5 |
| | Average | 10 |
| | Complex | 15 |
| Classes | Simple | 5 |
| | Average | 10 |
| | Complex | 15 |

Now the number of dependency relationship can be calculated. n

$$NODR = \sum_{i=1}^{n} ((NODRAC)+(NODRAC))$$
$$= U \text{ Function Points(Assume)}$$

**3.5 Realization:** Realization is a specialization where it can be existed between two classifiers, where one will act as contract and the other will give the guarantee to carry out the contract. It can be represented by using dotted line with triangle head arrow.

| Component | category | weight |
|---|---|---|
| Actor and collaboration | complex | 5 |

Identify the relationship category and weightage can be done.

Number of realization relationship can be calculated

$$NORR = \sum^{n} ( ( NORRAAC)$$

i =1
= V Function Points(Assume)

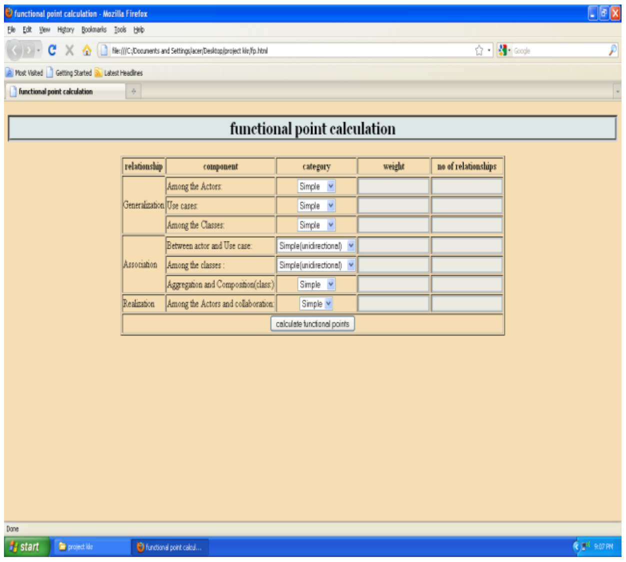
Now the relationship function points can be calculated

RFP= ∑ (X+Y+Z+W+U+V+Q ) Function Points, and the Total complexity factor is the multiplication of cost drives

FP= RFP*TCF

This is about how can function point can be calculated without considering EI, EO, EQ, IF, EF.

This results screen explains about how we can give weightage to the components in the relationships so that the above procedure can be carried out.



functional point calculation

| relationship | component | category | weight | no of relationships |
|---|---|---|---|---|
| Generalization | Among the Actors: | Simple | | |
| | Use cases: | Simple | | |
| | Among the Classes: | Simple | | |
| Association | Between actor and Use case: | Simple(unidirectional) | | |
| | Among the classes : | Simple(unidirectional) | | |
| | Aggregation and Composition(class:) | Simple | | |
| Realization | Among the Actors and collaboration: | Simple | | |

functional point calculation - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Documents and Settings/acer/Desktop/project kkr/fp.html
Most Visited
Getting Started
Latest Headlines
functional point calculation
calculate functional points
Done
start
project kkr
functional point calcul...

## 4. CONCLUSIONS

UML Relationships and Diagrams can be useful basis to calculate function points in the early phase of the life cycle of a software system. This paper introduced the idea of using UML Relationships as the starting point and gave the brief introduction on how to get the building blocks of the function point analysis out of the diagrams, Relationships. The original Albrecht's method used in our course relies heavily on assignment of the individual weighting factor, is restricted by the purpose of its development and lacks the systematic approach to be used directly to build automated tools for calculating function point. This is why several more suitable methods have been developed, using these methods with carefully constructed UML specifications, function points can be calculated effectively already in the early phases of a design process.

**Note:**

NOGR=number of generalization relationships

NOGAA= number of generalization among the actors

NOGAU= number of generalization among the use cases

NOGAC= number of generalization among the classes

NOAR= number of association relationships

NOARBAU= number of association relationships between actors and use cases

NOARBC= number of association relationships between classes

NODR= number of dependency relationships

NODRAU= number of dependency relationships among actors and use cases

NODRAC= number of dependency relationships among actors and classes

NORR= number of realization relationships

NORRAAC= number of realization relationships among actors and simple collaboration

NOCR = Number of composition relationships .

NOCRC = Number of composition relationships among the classes.

TCF=total complexity factor

## 5. REFERENCES

[1] A.J Albrecht. Function point analysis. Encyclopedia of software engineering, 1:John Wiley & Sons, 1994.

[2] Rational. UML1.1 Notation guide. Rational Software,1997

[3] S.Zamir. Handbook of Object Technology. CRC Press, 1999.

[4] Grady Booch, James Rumbaugh, Jacobson "Unified Modeling Language User Guide". PE,ISBN 81-7758-372-7

[5]. Pankaj Jalote ," An Integrated Approach to Soft ware Engineering" 2 nd Edition , Narosa Publishing House,2004,Chapter-4 (Planning a Software Project),Pg no. 166-170. ISBN – 81-7319-271-5

[6]. Roger S Pressmen, "Software Engineering - a Practitioner's Approch" 6th Eddition Mc Graw Hill international Edition, Pearson education, ISBN 007 - 124083 - 7

[7]. Waman S Jawdekar, "Software Engineering Principles and Practices" Tata Mc graw hill ISBN 0 -07 - 058371 – 4

[8]. Walker Royce "Software Project Management - A unified frame work" 2nd Edition, low price Edition, ISBN 81 - 7758 - 378 - 6, pearson education

[9]. Ian somarville, "Software Engineering" 5th Edition low price Edition, International Computer Science Series.

[10]. Shari, Laurance, Pfleeger, "Software Engineering theory and practies" 2nd Edition, ISBN 81 - 7808 - 4589 - 7, low price edition.

[11]. Carlo ghezzi, Mehdi Jazayeri Dino Mandrioli, "Fundamentals of Software Engineering", PHI, ISBN 81 - 203 - 0865 .