DOI: http://dx.doi.org/10.26483/ijarcs.v8i9.4909

ISSN No. 0976-5697

Volume 8, No. 9, November-December 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

SIZE ESTIMATION AT DESIGN STAGES OF SYSTEM DEVELOPMENT

Dr.Madhu Bhan
Department of Computer Applications
Ramaiah Institute of Technology
Bangalore,India

Dr.Rajanikanth K
Department of Information Science and Engineering
Ramaiah Institute of Technology
Bangalore, India

Dr. T.V. Suresh Kumar
Department of Computer Applications
Ramaiah Institute of Technology
Bangalore,India

Abstract: Class Point approach has been proposed to estimate software size of Object-Oriented Systems, systems that involve classes, encapsulation, inheritance and message passing. Another similar approach called Class Method Points has also been proposed in the literature to estimate the size of Object-Oriented software systems. We demonstrate estimation of the effort and size of OLAP system using Class Point approach and Class Method Points approach in the early stages of their development life cycle.

Keywords: class point; class method points; function point analysis; OLAP; software effort; software size

I. INTRODUTION

The Class Point approach[1,2] is based on the quantification of classes in a program which is analogous to the function counting performed in the Function Point Analysis FPA[3]. FPA is applicable to procedural paradigm where the basic programming units are functions or procedures. In the Object-Oriented systems, a framework called Class-point approach which considers classes as the basic building blocks is used. There are three phases in Class Point approach [1,2]. In the first phase, the classes identified from the design specifications are grouped into four types of system components. These are the problem domain type (PDT), the human interaction type (HIT), the data management type (DMT) and the task management type (TMT). In the second phase, each class identified from the design document is allocated a complexity level, depending on the number of external methods (NEM) and the number of services requested (NSR). The complexity measure so determined is used to derive a class point measure called CP1 which is meant to be used at the beginning of the development process. This CP1 gives a preliminary size estimate. This is later refined, when more information is available, by computing another measure called CP2 which takes into account an additional parameter called the number of attributes(NOA). Then the Total Unadjusted Class Point value (TUCP) is computed as a weighted total of the four components of the application as given by the formula

$$TUCP = \sum_{i=1}^{4} \sum_{j=1}^{3} w_{ij} * x_{ij} \qquad \text{Equation 1}$$

where x_{ij} is the number of classes of component type i (PDT, HIT etc.) with complexity level j (low, average, or high), and w_{ij} is the weighting value for type i and complexity level j. In the last phase, an adjustment value called Technical Complexity Factor (TCF) is determined by assigning the degree of influence that 18 general system characteristics such as User adaptivity, Rapid Prototyping, Multiuser Interactivity, Multiple Interfaces have on the application from designer's point of view. The formula for TCF is given below

$$TCF = 0.55 + 0.01 * \sum_{i=1}^{18} f_i \qquad \text{Equation 2}$$

Finally the Class Point value is determined by multiplying the TUCP by TCF as given by the formula

$$CP = TUCP * TCF \qquad \text{Equation 3}$$

The Class Method Points approach is another popular approach for estimating the size of Object Oriented software systems. There are five scope inputs in this approach [4]. These are:

1. The number of Control classes
2. The number of Interface classes
3. The number of other classes
4. The total number of Methods (member functions) within all classes combined
5. The total number of Database Tables

Each of these may be converted to Function Points(FP) or equivalent Source Lines of Code (eSLOC) using organizational specific data or typical industry-standard numbers [4] as given in the Table 1 below.

Table 1. Typical Conversion Factors

Metric	Convert to FP	Convert to eSLOC
Control Class	2	94
Interface Class	1.25	59
Other Class	1	47
Method	0.5	24
Database Table	6	282

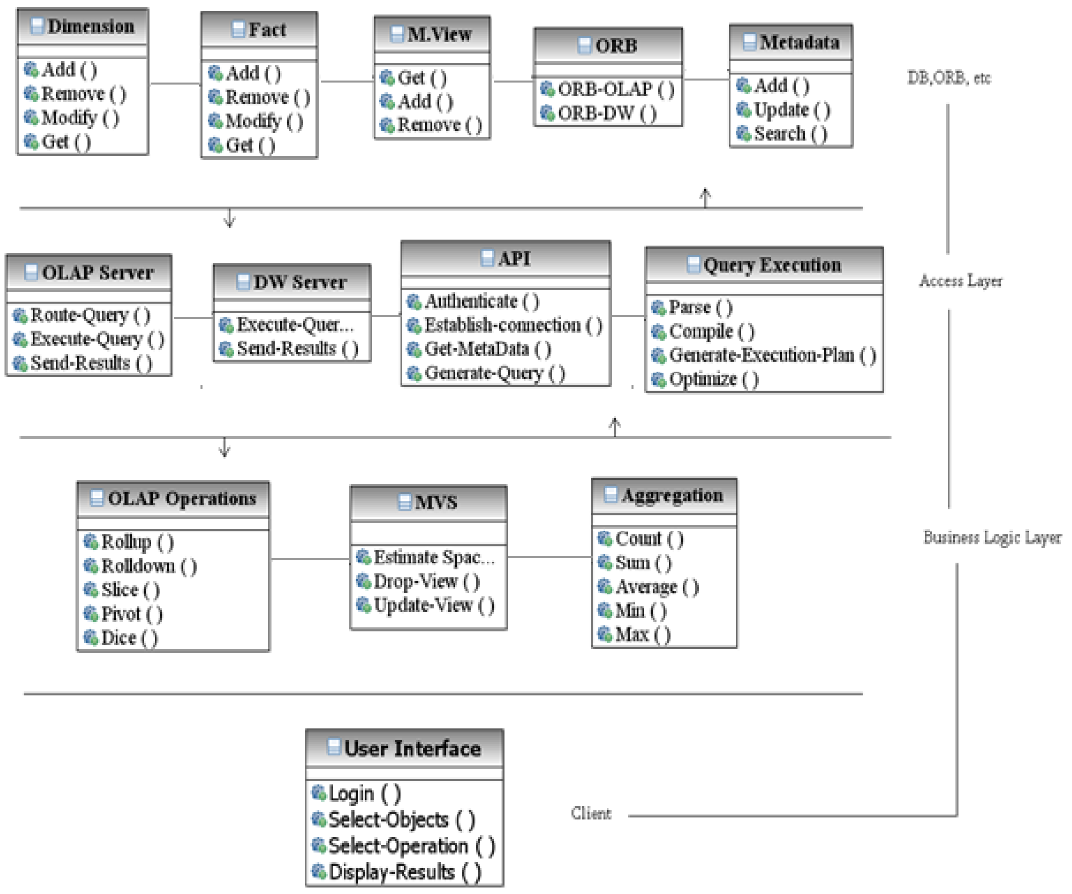

Figure 1. Class Diagram for OLAP System

II. CASE STUDY-A TYPICAL OLAP SYSTEM

A demonstration of the use of Class Point approach and Class Method Points approach for software size estimation of Data Warehouse/OLAP Systems is presented using a case study based on the TPC-H benchmark [5].

OLAP Systems considerably ease the process of analyzing large amounts of data, stored in Data Warehouses [6,7]. The UML class diagram in Figure 1 shows the static structure of a typical OLAP System. The system has three categories of classes namely Dimension, Fact, M.View and 10 other distinct classes namely ORB, Metadata, OLAP Server, DW Server, API, Query Execution, OLAP Operation, MVS, Aggregation and User Interface. The category dimension includes classes to which are mapped all the tables corresponding to all the dimensions. The category Fact includes classes to which are mapped all the tables corresponding to all the Facts. The category M.View (Materialized View) includes classes to which are mapped all the views. In this case study we are assuming that the total number of classes belonging to dimension category together with fact category are n and the number of classes belonging to M.View category are m. There are 2 other classes belonging to the data store group, 4 belonging to the access layer, 3 belonging to the Business logic layer and 1 class representing the user interface. The responsibilities of these classes are described below:

1. A class belonging to Dimension Category: Each table corresponding to a dimension is mapped to a class. A dimension has associated methods, such as an Add ( ) to add a record to the dimension, Remove() to delete a record, Modify() to update a record and Get() to read a record from the dimension table.

2. A class belonging to Fact Category: Facts are mapped to classes. A Fact class has a Add ( ) method to add a record to the Fact, Remove ( ) to delete a record from the Fact, Modify() to update a record and Get ( ) to read a record from the Fact table.

3. A class belonging to M.View Category: Materialized views are mapped to this class. Add ( ), Remove ( ), Update ( ) and Get ( ) are methods of this class to add a record to the view, to delete a record from a view, to update a record of a view and to read a record of view.

4. ORB (Object Request Broker): This class is responsible for sending and receiving messages to/from resources and other services distributed across several servers. It has methods like ORB-OLAP() and ORB-DW() to implement a communication channel through which applications can access and request data and other services. CORBA requires an Object Request Broker both on the OLAP API client computer and on the OLAP Services computer [8].

5. Metadata: The Metadata Class informs applications about the data that is available within the database. It has methods add( ), Update( ) and search( ).

6. OLAP Server Class: Materialized views are managed by an OLAP Server Class which is responsible for controlling access and retrieval of data. The server dispenses information to client applications. Methods like Route Query( ) route the query to Data Warehouse server when the results are not

available with the OLAP Server. Execute Query( ) is a method which executes the query sent by the user. Send-Results( ) to send results to the client.

7. DW(Data Warehouse) Server Class: This class is responsible for controlling access and retrieval of data between client and the Data Warehouse. The server releases information to client applications.

8. API (Application Program Interface) Class: The client and the Server Classes communicate through a well defined set of standard application program interfaces (API's) [8] This class has methods like authenticate( ), establish-conn( ) to authenticate and establish connection with the client.

9. Query Execution Class: This class is responsible for query execution which includes parsing of the input query, optimizing the submitted SQL statements followed by compilation and generation of the query execution plan.

10. OLAP Operations Class: This class has methods like slice(), dice() roll-up ( ), drill-down ( ) and pivot ( ). A rollup ( ) query summarizes the data along one or more dimensions, a drill-down ( ) allows the user to step from the current data cube to a more detailed data cube. Slice ( ) method picks a rectangular subset of a cube by choosing a single value for one of the dimensions and the dice ( ) method produces a sub-cube based on specific values of multiple dimensions. Pivot ( ) shows a rotated sub-cube.

11. MVS (Materialized View Selection): MVS Class determines the views to be materialized and the existing materialized views that should be dropped based on the estimated space requirements. It also ensures that all materialized views are refreshed each time the database is updated.

12. Aggregation: Aggregate values of the measures are computed through aggregate methods like Count (), Sum ( ), Avg ( ), Min ( ) and Max ( ).

13. User Interface: The end-user Interface class has methods like login ( ), Select-objects (), Select- operations ( ) and Display ( ).

A. Application of the Class Point Approach

Following the Class Point approach, the phases for calculating the class Point for OLAP System are as follows:

Phase 1:

The data model incorporated into a database system defines a framework of concepts that can be used to express the problem domain. Thus the PDT classes of a typical OLAP System are the Dimensions and Facts of the Data Warehouse. The classes of HIT type are designed to satisfy the need for information visualization and human-computer interaction. Thus the User Interface class belongs to the type HIT. The DMT component encompasses the classes that offer functionality for data storage and retrieval. Thus Data Management in a typical OLAP System is performed by classes OLAP Server, DW Server, Materialized views, MVS and Metadata. Task Management Classes are responsible for managing tasks. This type of Class include classes responsible for communication between subsystems and external systems. Task Management in a typical OLAP system is performed by classes ORB, Query Execution, API, OLAP operations and Aggregation[8,9]. Table 2 shows this categorization of classes.

Table 2. Categorization of Classes

Class Type	**Corresponding classes in the OLAP System**
Problem Domain Type (PDT)	1. Classes belonging to Dimension category 2. Classes belonging to Facts category (A total of n classes)
Human Interface Type (HIT)	13.User-Interface
Data Management Type (DMT)	3. Classes belonging to M.View category (A total of m classes) 5. Metadata 6. OLAP Server 7. DW Server 11. MVS
Task Management Type (TMT)	4. ORB 8. API 9. Query Execution 10. OLAP operations 12. Aggregation

Phase 2:

Here we will restrict our study to only CP1 inorder to find the complexity and weights associated with various classes of the OLAP system. The complexity level of a class is defined by the range of values of NSR and NEM, as given in Table 3 [1].

Table 3. Typical Conversion Factors

	0-4 NEM	5-8 NEM	≥9 NEM
0-1 NSR	Low	Low	Average
2-3 NSR	Low	Average	High
≥4 NSR	Average	High	High

We identify the values of NSR and NEM for each class and use this data as indicated by Table 3 to determine the complexity of each class of the OLAP System. For example, the Aggregation class as shown in Figure 1 has 5 public methods namely Count( ), Average( ), Max( ), Min( ) and Sum( ). The Aggregation class requests Get( ) of Fact, Get( ) of Dimension, Add ( ) and Update ( ) of Metadata. Thus the number of external method is 5 and that of services requested is 4. Hence it is assigned a 'High' complexity level as per Table 3. The Query Execution Class has 4 public methods namely the Parse ( ), compile ( ), Generate-Execution Plan ( ) and Optimize ( ). It requests the services Update ( ), Add ( ), Search() of Metadata and Estimate-space( ) of MVS [8,9]. Thus The number of services requested is 4. It is assigned a 'Average' complexity level. A class belonging to dimension category has 4 public methods, Add ( ), Remove ( ), Modify( ) and Get ( ). Thus it is assigned a 'Low' complexity level [9]. The complexities of all the classes have been assigned in this manner and are shown in Table 4.

Table 4. Complexity of Classes

Class	**NEM**	**Type**	**NSR**	**Complexity**
A Class belonging to Dimension category	4	PDT	0	Low
A Class belonging to Fact category	4	PDT	0	Low
MVS	0	DMT	9	Average
OLAP Server	2	DMT	7	Average
DW Server	2	DMT	9	Average
Metadata	3	DMT	2	Low
Aggregation	5	DMT	4	High
A Class belonging to M.Views category	1	DMT	1	Low
ORB	2	TMT	0	Low
API	3	TMT	5	Average
Query Execution	4	TMT	4	Average
OLAP operation	5	TMT	5	High
User Interface	0	HIT	5	Average

We now count the number of classes under each type that belong to each of the three complexity levels. These counts are shown in Table 5 below.

Table 5. Count of Classes based on Complexities

System Component Type.	Low	Average	High
PDT	n	0	0
HIT	0	1	0
DMT	1+m	3	1
TMT	1	2	1

Table 6. Computation of CP1

System Component Type	**Complexity**			
	Low	Average	High	Total
PDT	…*3=	…*6=	…*10=	…
HIT	…*4=	…*7=	…*12=	…
DMT	…*5=	…*8=	…*13=	…
TMT	…*4=	…*6=	…*9=	…
CP1=Total Unadjusted Class Point (TUCP)				

After assigning a complexity level to each class, each class is weighted in accordance with its type and complexity level.

The weighted sum is computed giving the Total Unadjusted Class Point measure CP1. The template for these calculations is shown in Table 6 [1].

The computation of CP1 for our case study of OLAP System, is shown in Table 7 below.

Table 7. CP1 for the OLAP Case Study

System Component Type	**Complexity**			
	Low	Average	High	Total
PDT	n*3=3n	0*6=0	0*10=0	3n
HIT	0*4=0	1*7=7	0*12=0	7
DMT	(m+1)*5 =5m +5	3*8=24	1*13=13	42+5m
TMT	1*4=4	2*6=12	1*9=9	25
CP1=Total Unadjusted ClassPoint (TUCP)				74+3n+5m

Thus TUCP = 74 + 3n + 5m Equation 4

Phase 3:

The Technical Complexity Factor (TCF) is computed in this phase. We assign DI (degree of influence ranging from 0 to 5) that these 18 general system characteristics have on the application from the designer's perspective[1]. Table 8 below shows these estimates The justification for assigning weights to each of these characteristics is given below:

Table 8. Computation of Total Degree of Influence for the OLAP Case Study

System Characteristic	DI	System Characteristic	DI
Data Communication	5	Reusability	3
Distributed Architecture	5	Installation ease	2
Performance	5	Operational ease	4
Cross-Platform Support	4	Multiple sites	2
Transaction rate	1	Facilitation of change	4
On line data entry	0	User Adaptivity	4
End-user efficiency	5	Rapid Prototyping	3
Online update	1	Multiuser Interactivity	3
Complex processing	4	Multiple Interfaces	2
Total-Degree-of-Influence(TDI) =57			

Data communication - Since Data communication between the system components of an OLAP System is high, we assign a value of 5 to this factor.
Distributed Architecture - The architecture of OLAP System is Distributed. Hence we assign value of 5 to this factor.
Performance - The quickness of response for users is an important (and non-trivial) factor. We assign a value of 5 to represent increasing importance of response time of OLAP queries.
Cross-Platform Support - Issues like multi-platform support need to be handled by an OLAP System. We assign a value of 4 to this factor.
Transaction rate - Data Warehouse is a large repository of information characterized by read-only queries. We assign a low value of 1 to transaction rate factor.
On line Data entry - There is no online data entry for a Data Warehouse system. We assign a value of 0 to On line Data entry factor.
End user efficiency - OLAP System relies heavily on the application to improve user efficiency. Hence we assign a value of 5 to this factor.
Online update - There are very less online update operations in a Data Warehouse system. We assign a value of 1 to Online update factor.
Complex processing - Processing of OLAP cubes is complex. Hence we assign a value of 4 to this factor.
Reusability - Code reuse reduces the amount of effort required to deploy a System. The higher the level of re-use, the lower the number. We assume a average level of code re-use and assign a low value of 3 to this factor.
Installation ease - The higher the level of competence of the users, the lower the number. We assign a value of 2 to this factor, assuming that the users have an average competence
Operational ease - Since the importance of operational use of an OLAP System is high, a high value of 4 is assigned to this factor.
Multiple sites- This is not an overriding factor. Hence we assign a low value of 2 to this factor.
Facilitation of change - The higher the level of competence of the users, the lower the number. Since we expect the users to be from top management, we assign a value of 4 to this factor.
User Adaptivity- An OLAP System should be able to adapt for any given input based on subsequent interactions. Thus a value of 4 has been assigned to user adaptivity factor.
Rapid Prototyping - A value of 3 has been assigned to this factor since a user interface prototype is required which presents an interface for various inputs to realize a usability test.
Multiuser Interactivity - The system should provide simple control of the order of interaction among users. Thus a value of 3 is assigned to this factor.
Multiple Interfaces - Different Interfaces for novice and experienced users are required. Thus a value of 2 is assigned to this factor.
These 18 degrees of Influence as shown in Table 8 are added together to arrive at total degree of influence.
The Technical Complexity Factor (TCF) is calculated as:

$$TCF = 0.55 + 0.01(57) = 0.55 + 0.57 = 1.12 \qquad \text{Equation 5}$$

The final value of the Adjusted Class Point (CP) is obtained by multiplying the Total Unadjusted Class Point value by TCF. For OLAP Systems the final class point value, CP1, is

$$CP1 = 1.12(74 + 3n + 5m) \qquad \text{Equation 6}$$

Validation: The above approach is validated using TPC-H Database [5]. The TPC-H consists of eight separate tables belonging to Dimension category and Fact category. Thus the value of n is 8. From our experimental study we found that the performance after materializing 10 views. remains almost constant and hence we do not materialize the remaining possible views[9]. Thus m is taken as 10. Substituting these values into Equation 6 given above we get CP1 as

$$CP1=1.12(74 + 3 * 8 + 5 * 10) = 165.56 \qquad \text{Equation 7}$$

We validate this value indirectly by estimating Effort using this value of CP1 and then comparing the resulting Effort estimate with Effort estimates reported in literature. Effort is estimated using CP1 based on the regression equation reported in [1]. Thus

$$\begin{aligned} Effort &= 0.843 * (165.76) + 241.85. \\ &= 381 \text{ person hours(approx.)} \qquad \text{Equation 8} \end{aligned}$$

Effort estimates for typical OLAP System are reported in literature [10]. These are as follows:
1) Simple OLAP of low complexity - 2 person-weeks
2) OLAP of medium complexity - 2 person-month
3) OLAP of high complexity - 2 person-years.
In our case study the OLAP System is of medium complexity as the Data Warehouse is assumed to be already existing. According to the calculation shown in Equation 8 above the Effort estimate based on Class Point Approach is 381 person hours. Assuming the typical 40 hours per week, this Effort translates to approximately to 9.5 person-weeks which is very close to the reported value of 2 person- months. This demonstrates that the Class Point approach can be successfully used for estimating the size of software for OLAP Systems.

B. Application of the Class Method Points Approach

We analyze the Class diagram of Figure 1 to identify Control classes and Interface classes in an OLAP System. A Control Class is the one which provides an interface between the application domain, user interface, and database subsystems [11]. We identify three Control classes which are API, ORB and OLAP operations as shown in Figure 2 below.

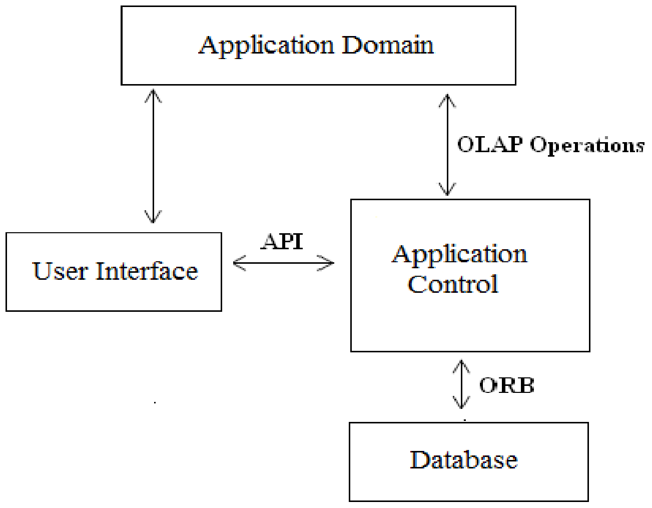

Figure 2. Control Classes

OLAP operations Class: The interface to the Application domain of an OLAP System is provided through OLAP operations.
ORB Class: This class provides an interface to the database which implements a communication path through which applications can access and request data from Database.
API Class: The class provides a well defined set of standard application program interfaces which allow client and the Application communicate. There is only one Interface Class, the User Interface Class [8]. The total number of methods throughout all the Classes is 46.
Thus applying the Class Method Points approach we get Function Point (FP) measure as given below

$FP = (3*2) + (1*1.25) + (9*1) + (46*0.5)$
$FP = 39.25$ Equation 9

In the above computation, we ignore Database tables as we consider FP of an OLAP System for which the Data Warehouse is already existing.

Validation:

We validate this value indirectly by estimating Effort using this value of FP and then comparing the resulting Effort estimate with Effort estimates computed earlier using CP1 .
Effort is calculated as

$Effort=FP/Productivity$

Assuming an average Productivity of 18 Function points per month [12], we get Effort by substituting the values of FP and Productivity in the formula

$Effort = 39.25/18$

Effort = 2.18 person months Equation 10

This value of Effort is very close to the estimated Effort of 2 person months obtained previously in sub-section *A*.
This demonstrates that the Class Method Points approach can also be successfully used for estimating the size of software for OLAP Systems.

III. CONCLUSION

Software Size estimation is one of the important inputs for performance assessment in the Software Performance Engineering(SPE) approach. The Class Point approach provides a size measure by suitably combining well known Object-Oriented measures. The Class Method Points approach estimates the size by considering distinct scope inputs. In this paper we demonstrated the use of Class Point approach and Class Method Points approach for effort and size estimation of OLAP Systems.

IV. REFERENCES

[1] Gennaro Costagliola and Genoveffa Tortora, “Class Point: An Approach for the Size Estimation of Object-Oriented Systems”, IEEE Transactions on Software Engineering, Vol. 31, No. 1, Jan.2005, pp 52-74.

[2] Wei Zhou and Q.Liu , ”Extended Class Point Approach of size estimation for OO product”, in the Proc. of International Conference on Computer Engineering and Technology, ICCET, Vo.l 4, 2010, pp 117-122.

[3] Bingchiang Jeng, Dowming Yeh, Deron Wang, Shulan Chu and Chia-Mei Chen, “A Specific Effort Estimation Method Using Function Point”, Journal of Information Science and Engineering 27, 2011, pp 1363-1376 .

[4] William Roetzheim, “Estimating Effort Using Use-Case and UML Class-Method Points”, in the Proc. of UML & Design World 2005, Austin, Texas, June 2005.

[5] www.tpc.org

[6] Veronika Stefanov, “Bridging the Gap between Data Warehouses and Organizations”, 13th Doctoral Consortium on Advanced Information Systems Engineering (CAiSE), Luxembourg, 2006, pp 1160-1167.

[7] Codd, E.F., S.B. Codd, and C.T. Salley, “Providing OLAP (On Line Analytical Processing) to User Analyst, An IT Mandate”, 1993, http://www.arborsoft.com/OLAP.html.

[8] Ali Bahrami, Object-Oriented System Development , International edition , 1999.

[9] Madhu Bhan,T.V.Suresh Kumar, K.Rajanikanth, “Size Estimation of OLAP Systems” in the proceedings of Computer Science and Information Technology(CS&IT) International Conference on Parallel, Distributed Computing technologies and Applications, DOI 10512/ CSIT. 2013.3649, pp(431-441)

[10] http://datawarehouse.ittoolbox.com,

[11] Bernd Bruegge, Kevin O’Toole and David Rothenberger, “Design Considerations for an Accident Management System”, in the Proc of the 2nd International Conference on Cooperative Information Systems, Toronto Press, May 1994.

[12] Gianfranco Lanza, “ Function Point: how to transform them in effort? This is the problem”, in the Proc. of the 5th Software Measurement European Forum, Milan, Itlay, 2008, pp 127-136.