SIZE ESTIMATION OF OLAP SYSTEMS

Madhu Bhan[1], T V Suresh Kumar[2] and K.Rajanikanth[3]

[1,2] M.S.Ramaiah Institute of Technology-Bangalore,India
[3] Visvesvaraya Technological University, India
madhoobhan@yahoo.co.in

ABSTRACT

Software size estimation at early stages of project development holds great significance to meet the competitive demands of software industry. Software size represents one of the most interesting internal attributes which has been used in several effort/cost models as a predictor of effort and cost needed to design and implement the software. The whole world is focusing towards object oriented paradigm thus it is essential to use an accurate methodology for measuring the size of object oriented projects. The class point approach is used to quantify classes which are the logical building blocks in object oriented paradigm. In this paper, we propose a class point based approach for software size estimation of On-Line Analytical Processing (OLAP) systems. OLAP is an approach to swiftly answer decision support queries based on multidimensional view of data. Materialized views can significantly reduce the execution time for decision support queries. We perform a case study based on the TPC-H benchmark which is a representative of OLAP System. We have used a Greedy based approach to determine a good set of views to be materialized. After finding the number of views, the class point approach is used to estimate the size of an OLAP System The results of our approach are validated.

KEYWORDS

Class point, Materialized views, OLAP Systems, Software size, performance

1. INTRODUCTION

In the present day scenario, estimating the size of the software has become a tedious task. Size evaluation is one of the main tasks for planning software project development with reliable cost, effort and performance estimation [15]. The applicability of Function Point (FP) approach to estimate software size is limited to procedure oriented systems [11]. FP is not suitable for object oriented programming paradigm which involves classes, encapsulation, inheritance and message passing [3], [5]. The idea underlying the Class Point method is the quantification of classes in a program in analogy to the function counting performed by the Function Point measure. In the procedural paradigm the basic programming units are functions or procedures; whereas, in the object-oriented paradigm, the logical building blocks are classes, which correspond to real-world objects and are related to each other [1],[8]. In this paper, we explore the Class Point approach for estimating the size of OLAP systems based on design documentation. Fast analysis of data stored in databases or warehouses, is indispensable for businesses that wish to stay ahead in the present competitive market scene**.** A data warehouse is a relational database that usually contains historical data derived from multiple, heterogeneous and independent data sources [16]. Materialized views are views that improve query execution times by pre-calculating expensive

Jan Zizka (Eds) : CCSIT, SIPP, AISC, PDCTA - 2013
pp. 431–441, 2013. © CS & IT-CSCP 2013 DOI : 10.5121/csit.2013.3649

joins and aggregation operations prior to execution of queries and storing the results at the data warehouse [7],[8]. This dramatically improves the response time of decision support queries. However the number of possible views exponentially increases relative to the number of database dimensions. We need to find out what views should be materialized in order to improve query performance under resource constraints. A greedy algorithm is adopted to choose the most beneficial view per storage space (Benefit-Per-Unit-Space) up to the given storage limit. The algorithm considers that there is a linear relationship between the cost of answering a user query and the size of the view that is used to answer that query. This cost, which is the number of rows in the view, is then used to select the most beneficial view for materialization. In this paper we study the TPCH Benchmark [6] to find the number of tables and views present in an OLAP system.

2. CLASS POINT APPROACH

The Class Point approach provides a system-level estimation of the size of OO products. The class point approach was introduced in 1998 [17]. In object-oriented development, the class diagram has a great deal of quantification information based on the design document. It contains the structural functionality of the target system and its class hierarchy, which are the logical blocks of the developed system. This approach to size estimation focuses on 1) Local methods 2)Interaction of the class 3) The attributes. The Class Point size estimation process is structured into three main phases, corresponding to analogous phases in the FP approach. During the first step the design specifications are analyzed in order to identify and classify the classes into four types of system components, namely the problem domain type, the Human interaction type, the data management type, and the task management type. During the second step, each identified class is assigned a complexity level, which is determined on the basis of the local methods in the class and of the interaction of the class with the rest of the system. The measures and the way they are used to carry out this step represent the substantial difference between CP1 and CP2. Indeed, in CP1 the complexity level of each class is determined on the basis of the Number of External Methods (NEM) and the Number of Services Requested (NSR). The NEM measure of a class in an object-oriented system is given by the number of public methods in a class. NSR is used to measure the interconnection of system components. It is again applicable to a single class and is determined by the number of different services requested to other classes. In CP2, besides the above measures, the Number Of Attributes (NOA) measure is also taken into account in order to calculate the complexity level of each class. Once a complexity level of each class has been assigned, such information and class type are used to assign a weight to the class. Then, the Total Unadjusted Class Point value (TUCP) is computed as a weighted sum. The Technical Complexity Factor (TCF) of the application is determined by assigning the degree of influence that 18 general system characteristics have on the application. The sum of the influence degrees related to such general system characteristics forms the Total Degree of Influence (TDI). Finally, the Class Point value is determined by adjusting the TUCP with a value obtained by considering global system characteristics as in FPA and some additional characteristics especially conceived for object-oriented systems, namely:

- User Adaptivity
- Rapid Prototyping
- Multiuser Interactivity
- Multiple Interfaces.

3. RELATED WORK

Many metrics for size estimation have been proposed for procedure oriented systems among which the Function points have achieved a wide acceptance in the estimation of size of business

systems [5]. The method provides estimation of size by measuring the functionality of the system to be developed.. This allows Function Point Analysis (FPA) to be applied in the early phases of the lifecycle, which is the main reason for the success of the method. Function point depends on the information available at the time of specifications [11].Several measures have been defined so far in order to estimate the size of software systems. Chidamber and Kemerer defined six measures for assessing OO systems.[12] Among them the Weighted Method per Class(WMC) , the Number of Children and the Response for a Class have been used as size measures. All of these measures are useful for productivity analysis as all of them provide class level measurement. The Use Case Points approach was introduced by Karner [4] as a software project effort estimation model. Use Case Point effort estimation is an extension of existing estimation methods, such as function point analysis. This approach, however, has weak points when applied to general software projects. In recent past other size measures have been suggested as adaptations of the FP method to Object oriented systems [2]. Whitmire proposes the application of his 3D Function Points to object-oriented software systems, by considering each class as an internal file and messages sent across the system boundary as transactions [14]. However, 3D Function Points require a greater degree of detail in order to determine size and consequently make early counting more difficult. Object point measure is another adaption of function point, used in the improved COCOMO2.0 effort estimation technique[13]. Object Point count is very similar to Function Point, but objects are taken as the basis of the counting process. However, such objects are not directly related to objects in the OO methodology, but rather refer to screens, reports, or 3GL modules. The class point approach is conceived by recasting the ideas underlying the FP analysis within the OO paradigm and by combining well known OO measures [1],[3],[9]. The class point size estimation process is structured in three main phases and two levels of complexity and classifying the system into component types.

4. OLAP SYSTEM

OLAP systems have become increasingly popular in many application areas as they considerably ease the process of analysing large amounts of data, stored in data warehouses [16]. Data warehouse must have efficient OLAP tools to explore the data and to provide users with real insight of the data in data warehouse. Due to large size of the data warehouse and the complexity of queries, quick response time plays an important role as timely access to information is the basic requirement of an OLAP system. Unified Modelling Language (UML) diagrams represents the static and dynamic aspects of a OLAP system. The UML class diagram in Figure.1 shows the static structural behaviour of the OLAP system, in which operations are designed for the complete system. The class diagram has persistent classes, like Dimensions ,Facts and Views and Control classes like ORB, Query Execution, OLAP API, OLAP operations, and Aggregation. These classes are related to each other by through associations. The access layer must be able to translate the data related requests from the user or business layer i.e it must be able to create the correct SQL statement and execute it. Server programs generally receive requests from the client from the client programs and execute database retrieval and updates. Each portion of the database is managed by a server, a process which is responsible for controlling access and retrieval of data from database portion. The server dispenses information to client applications. The client and the server processes communicate through a well defined set of standard application program interfaces(API's).The data model incorporated into a database system defines a framework of concepts that can be used to express the problem domain. Materialized views within the data warehouse are transparent to the end user or to the database application. Materialized views are usually accessed through the query rewrite mechanism. If a materialized view is to be used by query rewrite, it must be stored in the same database as the fact or detail tables on which it relies. The motivation for using materialized views is to improve performance. Materialized view management activities considers measuring the space to be used by materialized views ,determining which existing materialized views should be dropped, ensuring that all materialized

views are refreshed properly each time the database is updated. In such a system, aggregates play a very important role, because an OLAP query is usually an aggregated view on existing (relational) data. In SQL, you are probably familiar with aggregate functions like: COUNT, SUM, AVG, MIN and MAX. Execution of query by the relational engine involves parsing of the submitted statements, optimization of the SQL statements, compilation of the code, and generation of the query execution plan. During execution, programs call the storage engine to retrieve or manipulate the data stored in the database. A database administrator adds OLAP metadata to a data warehouse. The end result is the creation of one or more measure folders that contain one or more measures. The measures have dimensions, and the dimensions have hierarchies, levels, and attributes. An OLAP API gives access only to the measures that are contained in measure folders. Conceiving data as a cube with hierarchical dimensions leads to conceptually straight forward operations to facilitate analysis. Common OLAP operations include roll up, slice and dice, drill down and pivot. A roll-up involves summarizing the data along a dimension. Drill Down allows the user to move from the current data cube to a more detailed data cube. Slice is the act of picking a rectangular subset of a cube by choosing a single value for one of its dimensions, creating a new cube with one fewer dimension. The dice operation produces a subcube by allowing the analyst to pick specific values of multiple dimensions. The Object Request Broker (ORB) is a process which sends and receives messages to resources and other services distributed across multiple application servers. CORBA object request brokers(ORB's) implement a communication channel through which applications can access object interfaces and request data and services. CORBA requires an Object Request Broker both on the OLAP API client computer and on the OLAP Services computer. When an application calls a method that requires an interaction with an OLAP server, the client ORB intercepts the call, interacts with the OLAP servers ORB to find the object on the server side that can implement the request, passes the parameters, invokes the object's method, and returns the results. The client application forms the front end of the system which the user sees and interacts with. The end-user query model identifies all the conceptual query objects with which the application user interface will deal, thus taking advantage of the strengths of object-oriented design. It allows for a clear correspondence between user interface objects and OLAP API objects. The UML class diagram in Figure.1 shows the static structural behaviour of the OLAP system, in which operations are designed for the complete system. The class diagram has persistent classes, like Dimensions ,Facts and Views and Control classes like ORB, Query Execution, OLAP API, OLAP operations, and Aggregation. These classes are related to each other by through associations. The steps to calculate the class point for OLAP Systems are as follows:

1.Identification of classes
2.Determination of complexity of classes
3.Calculation of unadjusted class point
4.Calculation of technical complexity factor
5.Calculation of class point

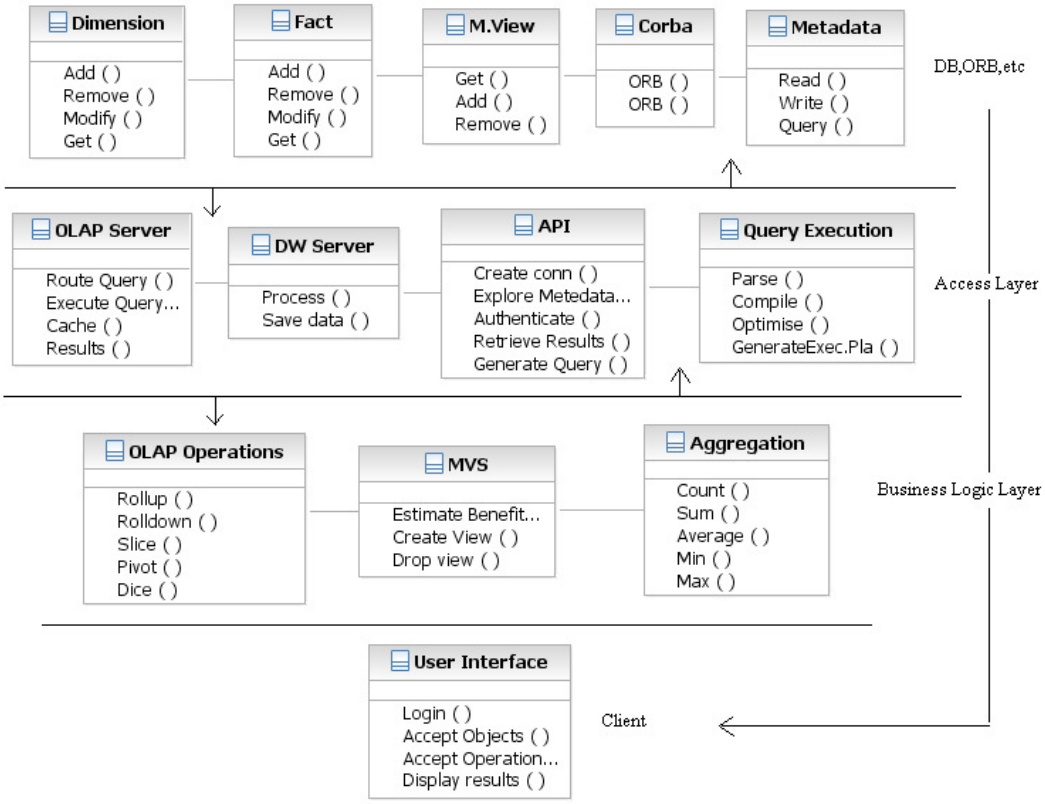


Figure 1.Class diagram

Step 1.

From the class diagram of OLAP system, the classes are classified into typical PDT, HIT, DMT and TMT classes as given in Table 1.

Step 2.

The class point method uses two complexity level measures CP1 and CP2. In CP1 the complexity level of each class is determined based on Number of external methods and Number of services requested. Both the measures are available in design documentation. The CP1 measure can be used in early phases of software development and the CP2 measure can be used only when the number of attributes is available [1]. Thus considering only CP1 the complexity and the weights associated with various classes forming an OLAP system is given in Table 2.

Table 1

Class type	Description	OLAP Systems
Prolem Domain Type(PDT)	Represents real world entities in the application domain	Dimensions and Facts
Human Interface Type(HIT)	Satisfies the need for visualizing information and human computer interactions.	User-Interface
Data Management Type(DMT)	Includes classes which Incorporate data storage and retrieval	OLAP server, DW server, MVS, Metadata, Mat. views
Task Management Type(TMT)	Includes classes which are responsible for tasks	ORB, Query Exec, OLAP API, OLAP operations, Aggregation

Table 2

Class	Type	NEM	NSR	Complexity	Weight.
Fact, Dimension	PDT	3	0	low	3
MVS	DMT	0	2	Avg.	8
OLAP server	DMT	2	10	Avg.	8
DW Server	DMT	1	7	High	13
Metadata	DMT	3	2	Low	5
Aggregate	DMT	5	2	Avg.	8
Mat. Views	DMT	3	0	Low	5
ORB	TMT	3	0	Low	4
OLAP API	TMT	4	8	Avg.	6
Query Exec	TMT	4	4	Avg.	6
OLAP Operation	TMT	5	5	High	9
User Interface	HIT	0	5	Avg.	7

Step 3.

Assuming that there are n dimensions and facts and m number of materialized views we can compute total unadjusted class point value. Thus, the TUCP is computed as the weighted total of the four components of the application:

$$TUCP = \sum_{i=1}^{4} \sum_{j=1}^{3} w_{ij} \times x_{ij};$$

where xij is the number of classes of component type i (problem domain, human interaction, etc.) with complexity level j (low, average, or high), and wij is the weighting value for type i and complexity level j.

Step 4.

The Technical Complexity Factor (TCF) is determined by assigning the degree of influence (ranging from 0 to 5) that 18 general system characteristics have on the application from the designer's point of view [1]. The sum of the influence degrees related to such general system characteristics forms the Total Degree of Influence (TDI), which is used to determine the TCF according to the following formula:

$$TCF = 0.55 + 0.01 \times \sum_{i=1}^{18} f_i$$

For OLAP Systems characteristics like Data communication, Distributed functions, Multiple users, Ease of operation ,adaptability by user have strong or significant influence on development of the system while as characteristics like Transaction rate, Online data entry, Online update and multiuser interaction have no influence or least influence on processing the system. The other characteristics Performance, Reusability, Compiler processing and High end configuration have average influence on the system. Based on the influence of these characteristics the TCF factor is calculated as

$$TCF=0.55+0.01\{55\}=0.55+0.55=1.10$$

Step 5.

The final value of the Adjusted Class Point (CP) is obtained by multiplying the Total Unadjusted Class Point value by TCF .

$$CP = TUCP \times TCF$$

The CP count can vary with respect to the unadjusted count from -45 percent +45 percent due to adjustment factor. For OLAP systems the final class point value is given as

$$CP=1.10(74+3n+5m) \qquad \text{--- eq. 1.}$$

Where n is the number of facts and dimensions and m is the number of materialized views. Experimental Study

5. EXPERIMENTAL STUDY

In order to determine the values of m and n in the above Complexity formula we have used TPC–H Benchmark for experimentation and Illustration [6]. The TPC Benchmark H models a data warehouse for any organization which must sell, distribute or manage a product worldwide. The data base has data about each such transaction over a period of seven years. The TPC-H database is defined to consist of eight separate tables. The name of the tables in itself indicate their contents: part, supplier, partsupp, customer, nation, region, lineitem and orders The queries and the data populating the database have been widely used in research as it has industry-wide relevance. Using Table-Class mapping where a single table is mapped to a single class we have obtained the value of n as 8 [2]. In order to obtain the value of m i.e the number of materialized views we need to understand the Lattice structure of a cube and the concept behind Greedy algorithm for selecting the views. The challenge for the design of OLAP systems is the exponential explosion of possible views as the number of dimensions increases. If D is the

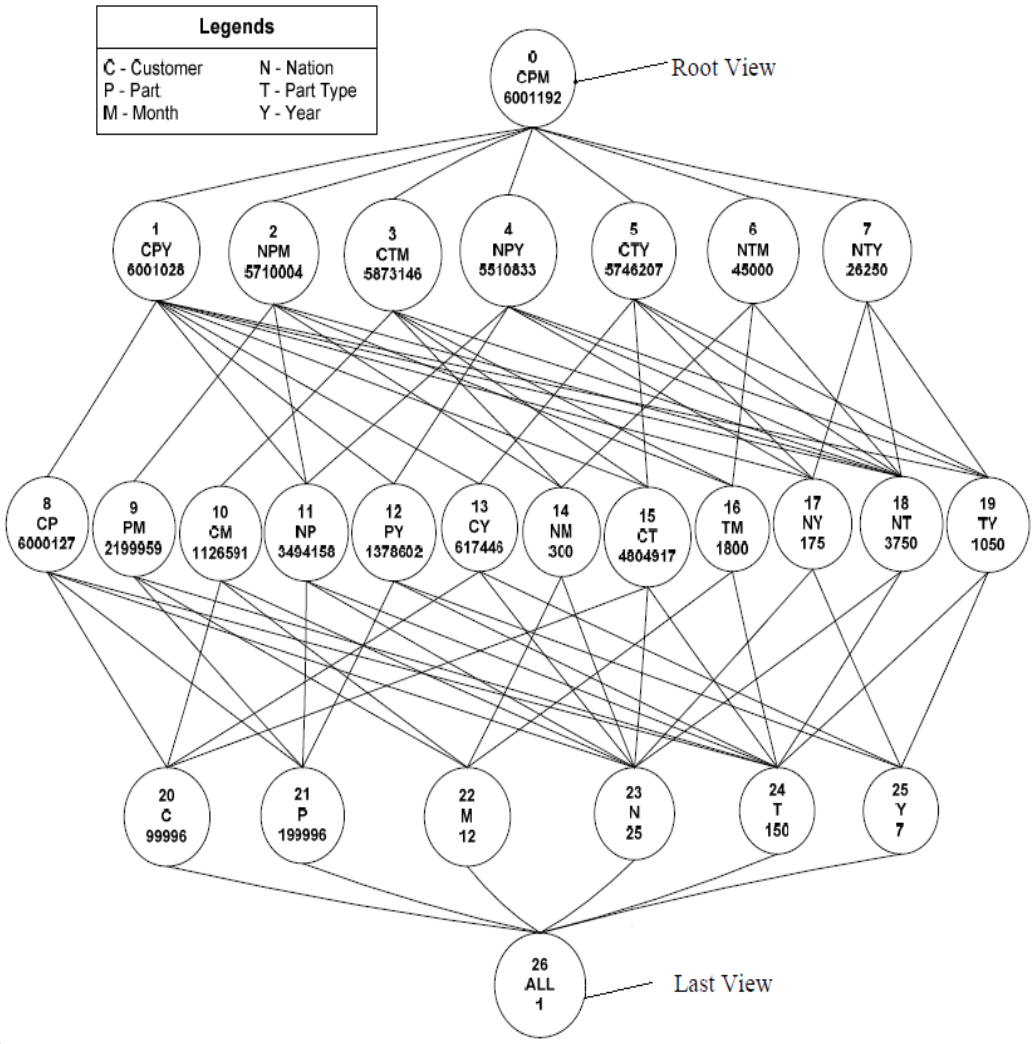
number of dimensions and hi the number of hierarchical levels in dimension i then the general equation for calculating the number of possible views is given by Equation

$$\text{Possible views} = \prod_{i=1}^{D} hi$$

As dimensionality increases linearly, the number of possible views explodes exponentially. OLAP system cannot materialize all the views in a given lattice structure because of constraints on storage space, computational time and view maintenance cost.. Typically, a strategic subset of views must be selected for materialization. There is a need to select an optimal set of views to be materialized. Using Greedy based approach for view selection we select a beneficial view at each step that fits within the space available for view materialization. Greedy algorithm considers Cost(vw), the cost associated with each view based on the number of rows in the view. k is the number of views to be materialized in addition to the root view. After selecting set S of views, the benefit of view vw relative to S is denoted by Ben(vw,S). For selecting a set of k views to be materialized ,the Greedy Algorithm is given below:

```
S = {root view};
for i = 1 to k do begin
select that view vw not in S such that Ben(vw,S) is maximized;
S = S union {vw};
end;
```

For our experimental study, we have populated a 1-GB Benchmark database. We populated the root node from this database using the Customer (C), Part (P), Month (M) dimensions, and the "Sale" measure. The customer dimension has 3 levels of hierarchy customer, nation and all. The part dimension has 3 levels of hierarchy part, part type, all. The sales are analyzed at three levels of time hierarchy-Month, Year, All. We ran the greedy algorithm on the lattice of Figure 2 using the TPC-H database as described above. When k=5, the greedy algorithm picks the root view (view 0) view then view 6 whose benefit is maximum as all the views below it are each improved from 6001192 to 45000,when we use view 6 in place of view 0. Similarly view 9,view 10,view 11 are picked according to the order of their respective benefits. Table 3 shows the results for the number of views to be materialized derived after applying Greedy Algorithm. In the first problem instance, we imposed a constraint of 5 views to be materialized; in the second problem instance, we set it to 10 views; in the third.



.

Figure 2:Lattice Structure with Actual Number of Rows
(Source:TPCH Benchmark Database)

Table 3

Lattice	Problem Instance	Optimal solution (Set of Materialized Views)
TPC-H	Materialize 5 views	0,6,9,10,11
	Materialize 10 views	0,2,6,9,10,11,12,15,20,21
	Materialize 20 views	0,2,3,4,5,6,7,8, 9,10,11,12,13,14,15,16,17,18,20,21,24

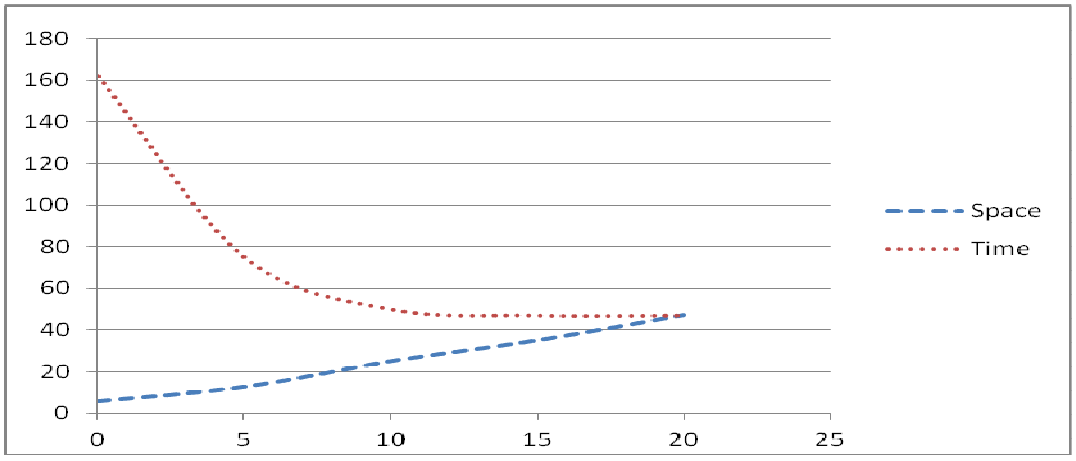


Figure 3

problem instance, we set it to 20 views. The views picked up by the greedy algorithm with maximum benefit in terms of storage space are shown in the table. The Y-axis in the graph of Figure 3 shows the total time taken as well as the space used. On X-axis it has the number of views picked. From the graph we can make a clear decision of when to stop materializing views. It is clear from the graph that for the first 10 views the query time in terms of number of rows is reduced substantially. However we have observed that performance after materializing 10 views remains constant and hence we do not materialize the remaining possible views. After knowing the number of tables and number of views we can calculate the value of CP1 as given in eq. 1 which equals to 158. The Effort is defined using regression analysis [1] as Effort=0.843*CP1+241.85. Thus the effort comes to be equal to 380 person hours.

6. VALIDATION AND CONCLUSION

Size estimation is one of the critical tasks in object oriented software project management. It is widely accepted that system size is strongly correlated with development effort [13], [18], [19], [20]. The Class Point approach provides a system-level size measure by suitably combining well known OO measures, which consider specific aspects of a single class. In particular, two measures are proposed, namely, CP1 and CP2. We have used CP1 at the beginning of the development process of OLAP System to carry out a preliminary size estimation. In this paper we have used UML class diagram document for size estimation of OLAP systems. The results of our approach are validated from websites like - http://datawarehouse.ittoolbox.com, which estimate effort of OLAP Systems based on their complexity to be as 1) Simple OLAP - 2 weeks 2) Medium OLAP - 2 months 3) Complex OLAP - 2 years. Assuming Medium case OLAP system where the data warehouse is already existing, our approach to size estimation is very close to industry estimated values. Future work may include refinement of size estimation by applying CP2.

REFERENCES

[1] Gennaro Costagliola and Genoveffa Tortora, "Class Point: An Approach for the Size Estimation of Object-Oriented Systems", IEEE transaction on Software Engineering, Vol. 31, No. 1,January 2005,page 52-74.

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

[3] Wei Zhou and Qiang Liu," Extended Class Point Approach of size Estimation for OO Product", IEEE sponsored 2nd International Conference on Computer Engineering and Technology,2010,Vol-Page:117-122.
[4] Parastoo Mohagheghi, Bente Anda, and Reidar Conradi, "Effort Estimation of Use Cases for Incremental large-Scale Software Development, " Proceedings of the International Conference on Software Engineering (ICSE'05), pp. 303-311, May 15-21. 2005.
[5] Roger S. Pressman, ―Software Engineering A practioner's Approach, published McGraw Hill, 2005.
[6] TPC Benchmark H (Decision Support) Revision 2.14.4. http://www.tpc.org.
[7] Yang J., Karlapalem K., Li Q., Algorithms For Materialized View Design in Data Warehousing Environment. VLDB 1997.
[8] Andreas Rauber, Philipp Tomsieh, "An Arichitecture for modular On- Line Analytical Processing systems:Supporting Distributed and Parallel Query Procesing using Co-operating CORBA objects". DEXA Workshop,1999.pp45-49.
[9] Aditi Kapoor, Parul Pandey ,"Fuzzy Class Point Approach".Journal, IJRTET,Volume 5, Issue 1, 2011.
[10] V. Harinarayan, A. Rajaraman, and J. D. IJllman. Implementing Data Cubes Efficiently. A f~dl version of this paper. At http://clb.stanford. edu/ pllb/harinarayan /199,5/cube.ps
[11] J.B. Dreger, Function Point Analysis. Prentice Hall, 1989.
[12] C.F. Kemerer and B.S. Porter, "Improving the Reliability of Function Point Measurement: An Empirical Study," IEEE Trans. Software Eng., vol. 18, no. 11, pp. 1011-1024, Nov. 1992.
[13] P. Nesi and T. Querci, "Effort Estimation and Prediction of Object- Oriented Systems," J. Systems and Software, vol. 42, pp. 89-102, 1998.
[14] S.A. Whitmire, "3D Function Points: Applications for Object- Oriented Software," Proc. ASM'96 Conf., 1996.
[15] L.C. Briand, S. Morasca, and V.R. Basili, ―Property Based Software Engineering Measurement‖, IEEE Transaction on Software Engineering, vol. 22, no. 1, pp. 68-86, 2009.
[16] Surajit Chaudhuri,Umeshwar Dayal,"An Overview of Data Warehousing and OLAP Technology" .SIGMOD Record, Vol.26,No.1,1997.pp 65-74 .
[17] Smith R. K, "Effort Estimation in Component-Based Software Development: Identifying Parameters", In the proceedings of the 36th ACM Southeast Regional Conference, Marietta, Georgia, USA, pp 323- 325,1998.
[18] V.B. Misic and D.N. Tesic, "Estimation of Effort and Complexity: an Object Oriented Case Study," J. Systems and Software, vol. 41, pp. 133-143, 1999.
[19] S. Moser, B. Henderson-Sellers, and V.B. Misic, "Cost Estimation Based on Business Models," J. Systems and Software, vol. 49, pp. 33- 42, 1999.
[20] B.W. Boehm, B. Clark, and E. Hiriwitz, "Cost Models for Future Life Cycle Processes: COCOMO 2. 0," Ann. Software Eng., vol. 1, no. 1, pp. 1-24, 1995.

Authors

Madhu Bhan. Faculty at M.S.Ramaiah Institute of Technology, Bangalore, India. Currently perusing P.hd. Area of Intrest is Databases and Data Warehousing

Dr.T.V. Suresh Kumar. Professor at M.S. Ramaiah Institute of Technology, Bangalore, India. Area of Interests include Software Performance Engineering,Object Technology.

Dr.K.Rajanikanth. Ph.d from Indian Institute of Science (IISc), Bangalore, India. Currently working as Chairman and Advisor for many bodies of Visvesvaraya Tech. University. Area of interests include Adhoc Networks, Image processing, Microprocessors and Micro-controllers, Data Warehouses.