

Measuring the functional size of a data warehouse application using COSMIC-FFP

Harold van Heeringen

Abstract

A data warehouse system is not the kind of traditional system that is easily sized with FPA, although there have been a number of attempts in literature to do so. In one of the proposed frameworks, the assumption has been made that it would probably make more sense to do a functional size measurement in COSMIC-FFP, because of the layer concept available to the measurer. COSMIC-FFP is a fairly new functional sizing method. One of the reasons this method is developed, is because practitioners found it hard to carry out function point analysis on modern types of functional documentation. This paper will explore the use of COSMIC-FFP to size data warehouse systems.

1. Introduction

Nowadays, more and more organizations hope to create a sustainable competitive advantage against their competitors by creating a system with which they can assess the current status of their operations at any moment and with which they can analyze trends and connections in truly up-to-date data. Therefore, one of the trends in the ICT market at the moment is the growing interest in the development of large data warehouses.

However, experience shows that these kinds of projects tend to fail even more often than 'normal' ICT projects. For some reason, it's very hard to estimate the amount of work that is required to build a data warehouse system. One of the methods that are quite common in estimating the amount of work in an IT project is to carry out some sort of functional size measurement (FSM). Doing this, one has to measure the functional size of the software to be developed, and then multiply this size by the projected project delivery rate (hours per size unit).

The most applied method of measuring functional size is Function Point Analysis (FPA). Another method, which is ISO certified as well, is COSMIC Full Function Points (COSMIC-FFP). In recent history, frameworks have been described of how to measure data warehouse applications with FPA. Santillo [1] for instance offers a very good way to estimate the size of data warehouses using FPA, but states in his conclusions that COSMIC-FFP will probably be a better way of measuring, because of the layer concept and because of the fact that the size of individual functions are not 'cut off' by the maximum size of a function, like in FPA.

In this paper, a way will be proposed how to measure the size of a data warehouse system using the COSMIC-FFP method.

2. What is a data warehouse?

Today, databases are the engines in almost every data-driven company. Through the years, databases have been optimized to support the operational business processes within these companies. However, as the number of different databases is rising within a single company, it's becoming more and more difficult to extract meaningful data from these databases. This is because the database entities and also the corresponding attributes do not always carry the

same name, or even the same data types, across the different databases. For instance, the entity Customer may be named ‘Client’ in the order processing system, but may be named ‘Debtor’ in the financial system. This makes it hard to generate a list with the total amount of outstanding orders in combination with a client’s current debt ratio. Standard database management systems are not able to produce these kinds of reports and they are generally not equipped to merge its own data with data from other databases.

In the mean time, the focus of more and more organizations shifts to decision support systems with which they can make sound decisions based on the data within the company. These kinds of systems have a direct influence on the strategic decision making capability of a company, which can lead to acquiring a sustainable competitive advantage over competitors. The implementation of a good decision support system starts with the clarification of the most important aspects of the operational business of a company. Information about these aspects must be gathered in a full and consistent way. Then it is possible to generate reports upon this data.

A data warehouse system does just the above. Bill Inmon, a pioneer in this field, defined a data warehouse to be: “a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management’s decision making process.”[10]. Another commonly used definition is the following: “A collection of decision support technologies, aimed at enabling the knowledge worker to make better and faster decisions.”[4].

A data warehouse system loads data from various operational databases, cleans this data, transforms the data attribute in a commonly defined attribute description and then stores the data into the data warehouse. In this process, a number of aggregations and calculations with the data are performed to speed up the performance of the tool. The rules that must be used to clean and transform the data and to perform the calculations are part of the metadata management system of the application. Users can extract the data from the data warehouse using for instance query tools. The components of a typical data warehouse system are listed in figure 1.

It’s obvious that a large number of differences exists between traditional transactional systems, which involve a lot of user interaction with the database (sometimes referred to as CRUD functionality), and data warehouses. The main differences between transactional processing systems and data warehouse systems are shown in table 1 [1].

Table 1: differences between transactional system and data warehouse system

	Transaction Processing	Data Warehouse
Purpose	Run day-to-day operations	Information retrieval and analysis
Structure	RDBMS optimized for Transaction processing	RDBMS optimized for Query Processing
Data Model	Normalized	Multi-dimensional
Access	SQL	SQL, plus Advanced Analytical tools.
Type of Data	Data that runs the business	Data to analyze the business
Nature of Data	Detailed	Summarized & Detailed
Data Indexes	Few	Many
Data Joins	Many	Some
Duplicated Data	Normalized DBMS	Denormalised DBMS
Derived data & aggregates	Rare	Common

Because of the fact that FSM methods are primarily developed to measure the size of transaction processing systems, these differences make it hard to apply existing functional size measurement methods to size the data warehouse type of software.

The architecture of a data warehouse system can be visualized as follows:

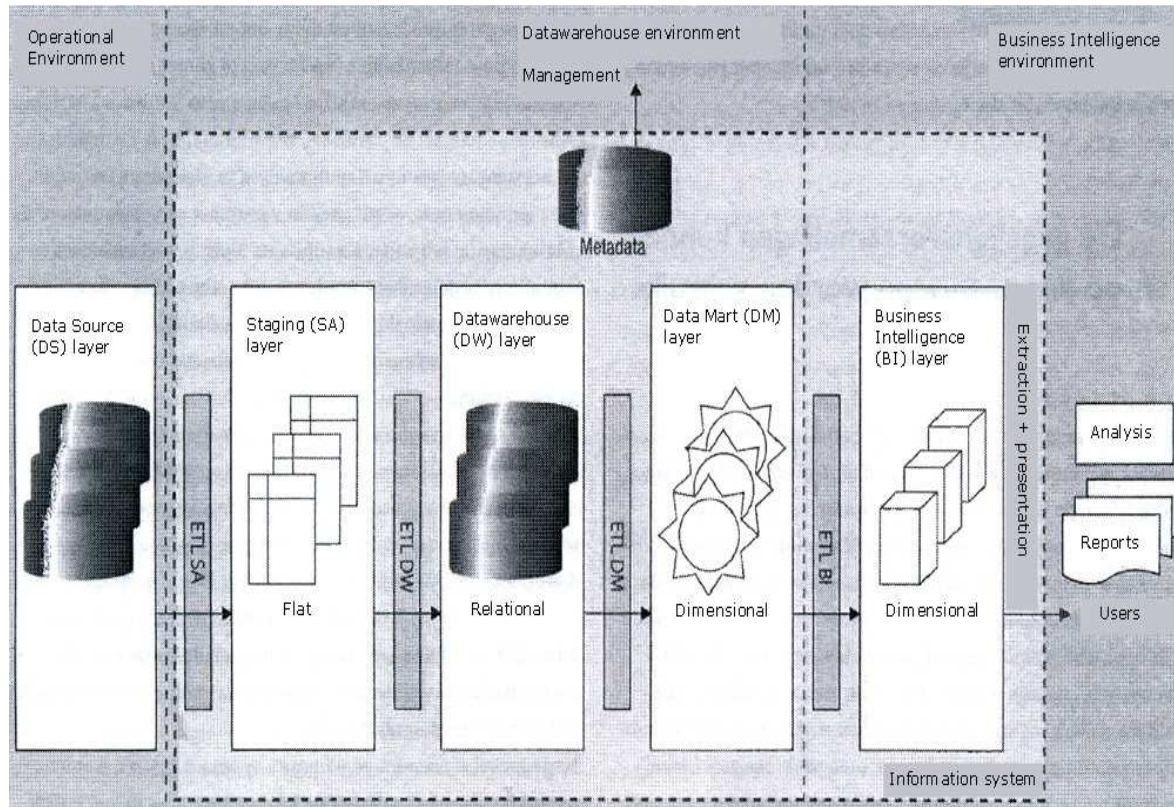


Figure 1: visualization of a data warehouse [2].

The main components of a data warehouse system are:

Operational data sources

An operational data source is the operational system whose function it is to capture the transactions of the business. The source systems should be thought of as being outside the data warehouse, since the data warehouse system has no control over the content and format of the data. The data in these systems can be in many formats from flat files to hierarchical and relational databases.

Data Staging Area

The data staging area is the portion of the data warehouse restricted to extracting, cleaning, matching and loading data from multiple source systems. The data staging area is explicitly off limits to the end users. The data staging area does not support query or presentation services. A data-cleansing tool may be used to process data in the staging area to resolve name and address misspellings and the like. It may as well resolve other data cleansing issues by use of fuzzy logic.

Extract Transform Load (ETL) Processes

Data Extraction-Transformation-Load (ETL) processes are used to extract data from data sources, cleanse the data, perform data transformations, and load the target data warehouse

and then again to load the data marts. The ETL processes are also used to generate and maintain a central metadata repository and support the data warehouse.

Data Warehouse Database

The warehouse is no special technology in itself. The data warehouse database is a relational data structure that is optimized for distribution. It collects and stores integrated sets of historical, non-volatile data from multiple operational systems and feeds them to one or more data marts. It becomes the one source of the truth for all shared data.

Data Marts

The easiest way to conceptually view a data mart is that a mart needs to be an extension of the data warehouse. Data is integrated as it enters the data warehouse from multiple sources. Data marts then derive their data from the central data warehouse source. The theory is that no matter how many data marts are created, all the data is drawn from the one and only one version of the truth, which is the data contained in the warehouse. Distribution of the data from the warehouse to the data marts provides the opportunity to build new summaries to fit a particular department's need. The data marts contain subject specific information supporting the requirements of the end users in individual business units. Data marts can provide rapid response to end-user requests if most queries are directed to pre-computed, aggregated data stored in the data mart.

Metadata management

Metadata literally means data about data. This is a very important part of any data warehouse system. Metadata is not the actual data; but rather information that addresses a number of data characteristics such as names and definitions of data elements, where the data comes from, how it is collected, and what transformations it goes through before being stored in the data warehouse. Additionally, meaningful metadata identifies important relationships among data elements that are critical to using and interpreting the data available through the end user query tools.

Business intelligence (End User functionality)

Before the end-users have access to the data, first the data is stored into the business intelligence layer. In this layer, the data can be visualized as cubes. There are numerous ways for users to extract the data from the data marts, or from the data warehouse. On-Line Analytical Processing (OLAP) tools provide the end users a way to access the data "multidimensionally" in a fast and easy manner. The main difference between OLAP and other generic query and reporting tools is that OLAP allows users to look at the data in terms of many dimensions. Furthermore, standard defined or ad hoc queries can be performed with other query tools. Another way to analyze the data is to employ some kind of data mining tool. These tools analyze the data and try to find correlations and meaningful patterns in a fully automated way.

Data transformation

Figure 1 shows how the data is transformed throughout the data warehouse system. In the operational data sources, it may exist in flat files, relational databases, or other forms of data storage. In the staging area, the data is transformed into flat files. When loaded into the data warehouse, it is quite common to store the data in relational tables, although sometimes data in data warehouses can be stored dimensionally as well. In data marts, data is stored in a dimensional way, as is also the fact in the business intelligence layer. Dimensional data

storage also means the storage of all kinds of aggregations and computations. in order to make the retrieval of the data faster. A common way to store the data in dimensional way is the use of the star schema. In a star schema, a fact table forms the center, and dimension tables form the surroundings. An example of this is presented in figure 2. The fact is computed by the chosen values of the dimensions.

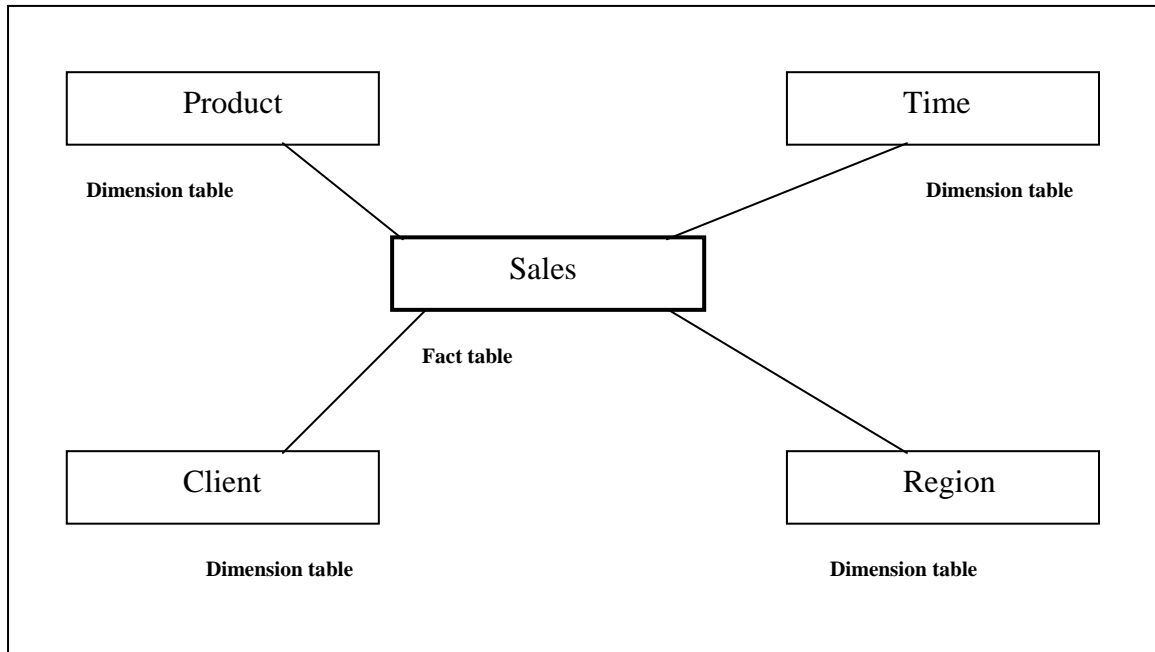
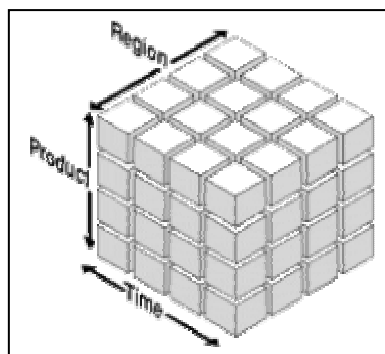


Figure 2: Star schema.

Dimension tables often consist of hierarchies. Typically, it is possible for end users to roll up a dimension, or drill down a dimension of the data. For instance, within the time dimension, days roll up to weeks, which roll up to months, which roll up to years.

In the business Intelligence layer, the data can be viewed as cubes. An example is shown in figure 3.



scheme

Figure 3: cubic

End users have tools at their disposal to slice and dice these cubes at will.

Now that it has been clarified what a data warehouse system is, it's useful to see which experiences are known with the estimation of these kinds of systems.

3. Experiences with estimating data warehouse systems

In recent years, there have been a number of attempts to propose ways of measuring data warehouse systems with FPA. However, the various authors don't seem to agree on how to apply the counting guidelines in a consistent way.

Santillo [1] presents an approach, which he doesn't consider to be a different sizing approach, but rather an 'instantiation' of FPA. He offers a few highly relevant counting guidelines, with which it is possible to make a good FPA count of the data warehouse system. He proposes for instance that each 'logical star' is an ILF and the fact table and dimensional tables involved should be counted as Record types (RET's), which contribute to the assessment of the complexity of the ILF. Schipper [2] also states that each star schema is an ILF, and the number of dimensions plus 1 (the fact table) equals the number of record types. Dekkers [5] proposes that each fact table should be counted as an ILF, but that the same should be true for all dimension tables (with the exclusion of coding tables) used in the application. The new Nesma guideline, written by Eveleens [6] agrees with Dekkers and proposes to count 1 ILF for every dimension table and for every fact table in the system.

Another example is the fact that Eveleens [6] states that metadata management should not be counted, because of the assumption that there will be a tool available to the user in which it is possible to manage the metadata. Santillo [1] however states that the data warehouse administrator is one of the figures which constitute the general system user, which means that some of the metadata tables can be considered ILF's, with the accompanying functions.

This means that there is some discussion in the field of functional size measurement as to how to map the FPA counting principles upon the different artifacts of a data warehouse system. This proves the assumption that it is quite hard to map the IFPUG or NESMA counting guidelines upon the database warehouse functional user requirements.

Santillo [1] states that the method COSMIC-FFP would probably be better to apply, because of the layer concept and because of the fact that there is no maximum number of points that a single function could receive. In the remainder of this paper this assumption will be examined.

4. COSMIC-FFP

Like FPA, COSMIC-FFP is a Functional Sizing Method. Both methods have been certified by ISO in December 2002, which means that both methods comply with the concepts and definitions of the ISO/IEC standard 14143 [7].

The reason that COSMIC-FFP has been developed in the nineties is because of the fact that it is difficult to apply the FPA counting principles on functional specifications of new development methods, like Object Oriented Development, Component Based Development and Rapid Application Development. FPA was created in the era of software development on the mainframe platform, based on specifications derived from functional decomposition. The use of new development methods and new types of architecture mean that functional specifications change as well. Furthermore, software systems seem to become increasingly complex, which means that the correlation between functional size and effort becomes less clear. COSMIC-FFP has been designed to be more easily applicable to modern types of software development methods and architectures. With this method it is possible to determine the size of particular pieces of software using a specific set of measurement guidelines, which

are described in the COSMIC-FFP measurement manual [3].

COSMIC-FFP principles

Like any ISO 14143 certified methods, from the complete set of requirements only the functional user requirements are measured. To do this, the Base Functional Components (BFC's) must be identified within the set of functional user requirements and these BFC's must be measured.

Base Functional Components in COSMIC-FFP are data movement types, which are identified per functional process type (figure 4). The underlying principles are:

1. Software is activated by input and produces output, or results, that is of use to the user.
2. Software processes chunks of data, which are materialized by data groups, which are a subset of an object of interest (OOI). A data group consists of one or more data attributes.

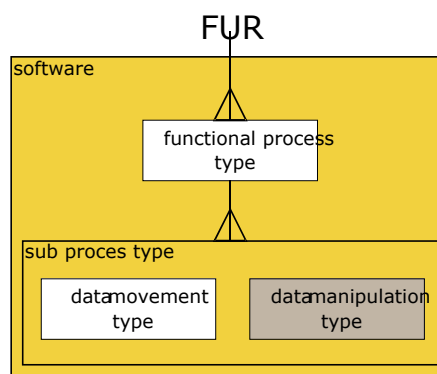


Figure 4: COSMIC BFC's

The FUR's are to be examined and the various functional processes are to be identified. Any of these functional processes consist of a number of sub processes (BFC's), which are called **data movements** (with included data manipulations). COSMIC-FFP recognizes four kinds of data movement sub processes, Entry, Exit, Read and Write types. The definitions of these types are the following [3]:

Entry: a data movement that moves a data group from a user across the boundary into the functional process where it is required.

Exit: a data movement that moves a data group from a functional process across the boundary to the user that requires it.

Read: a data movement that moves a data group from persistent storage within reach of the functional process which requires it.

Write: a data movement that moves a data group lying inside a functional process to persistent storage.

The different data movements are presented in figure 5.

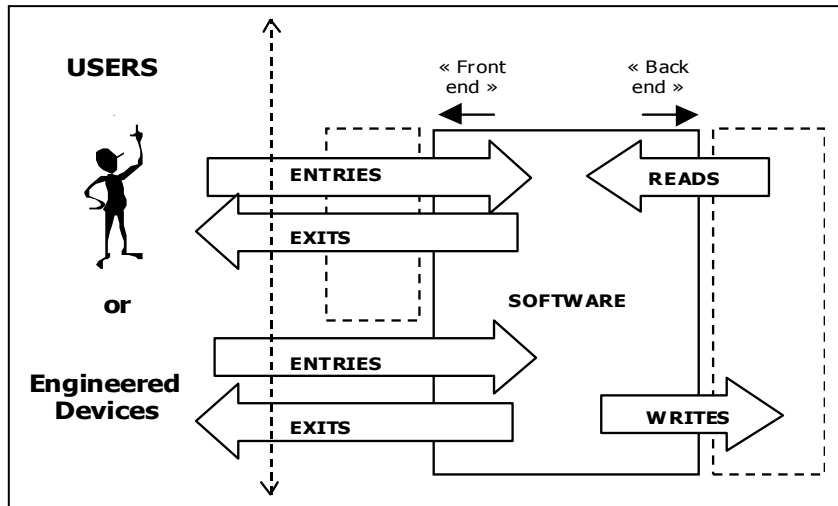


Figure 5: COSMIC-FFP data movements [3].

COSMIC-FFP provides the measurer with the layer concept, which allows the measurer to functionally partition the software into different layers, to make sure that all functional processes function on the same level of abstraction. There is always a form of hierarchical dependence of functional processes in a lower level upon functional processes in a higher level.

COSMIC-FFP also allows the software residing within one layer to be partitioned into 'peer components', if these components are developed with different technologies, or if they are implemented on different processors. A visualization of this is presented in figure 6.

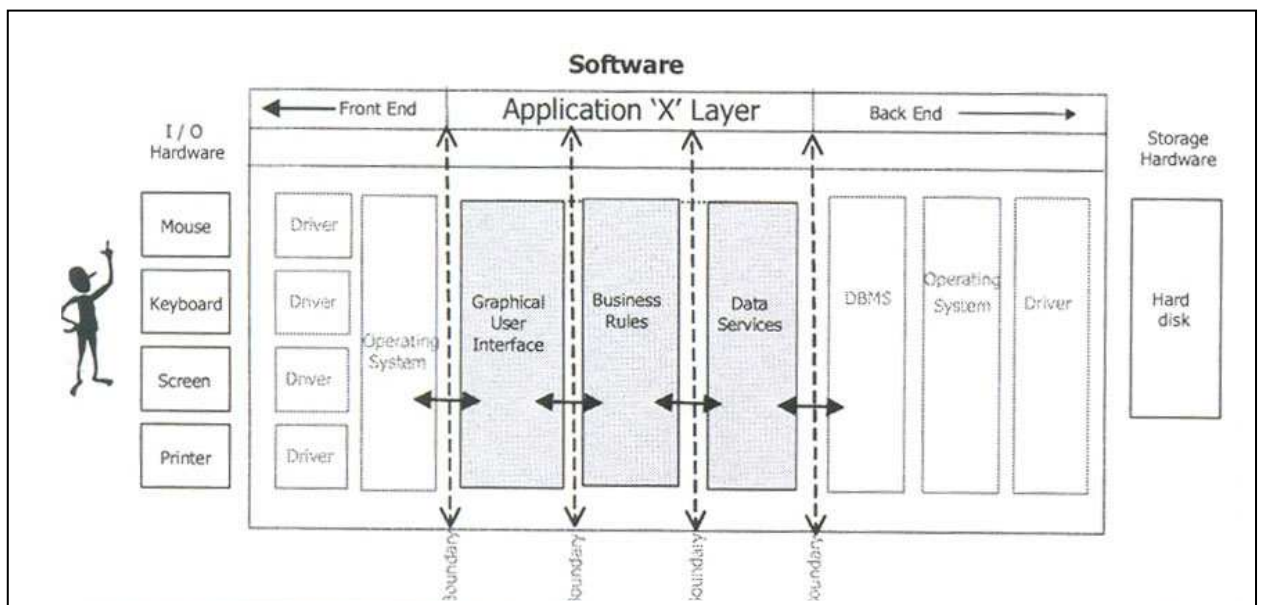


Figure 6: Dividing software within one layer into peer components

In this example, the architecture of the software is taken into consideration, and the GUI, the business rules and the data services are measured separately, although they still reside in the same layer. The amount of data movements between the components are measured, in addition to the standard interaction between the application layer and the users and between the application layer and persistent storage.

There are two main advantages that COSMIC-FFP provides on top of FPA:

1. The size of a functional process is not limited (In FPA the maximum number of points is limited to 7 per function).
2. It is possible to measure not only from the viewpoint of the end user (seeing only the input and output possibilities of the software), but it's also possible to measure different components of software.

5. Measuring Data warehouses with COSMIC-FFP

The COSMIC-FFP measurement manual [3] proposes a three-phase process to perform a functional size measurement with COSMIC-FFP:

1. Preparation
2. Mapping phase
3. Measurement phase

Phase 1: Preparation

In the preparation phase, the purpose, scope and measurement viewpoint of the analysis should be defined. In most cases, the purpose is to identify the functional size of the system. Within the scope of the measurement are all the data warehouse components mentioned above, with the exception of the operational data sources, which are considered to be on the outside of the boundary.

Possible users are: DB administrator, ETL procedures administrator, end-users, metadata management administrators and applications that provide the data to the warehouse.

Phase 2: Mapping phase

The mapping phase is the most important phase in the measurement process. In this phase, the FUR's are being examined and the various layers, functional processes, object of interests, data groups, and of course the data movements are identified.

Identifying layers

At first it may seem that the different components of a data warehouse system reside in different layers, but closer examination points out that this is not true. For a piece of software to reside in another layer, a number of conditions must be fulfilled. Most important, software that shares data with other software shall not be considered to be in different layers if they identically interpret the data attributes that they share [11]. Although the data attributes are transformed throughout the data warehouse from flat to relational to dimensional, the different components of the data warehouse all interpret the data attributes in the same way. This means that there is only one layer to be identified, the standard application layer.

However, almost any data warehouse system consists of the major components mentioned above and in figure 1. As already mentioned before, COSMIC-FFP allows the measurement of different peer components. This way of mapping is the way chosen in this paper.

The following components are identified:

- Data source components (outside of the scope of the measurement)
- Staging area component
- Data warehouse component
- Data mart component

- Business Intelligence component
- Metadata management component

Partitioning the software in this way means that boundaries are identified between peer components sharing data between them. The remainder of the mapping phase will be done per component.

Staging area component

Identifying objects of interests (OOI), data groups and functional processes (FP)

The first functional processes we are encountering are the ones that move data from operational data sources into the staging area component



Figure 7: Functional processes within Staging area component

To identify functional processes, first we should consider which objects of interests (OOI's) must be identified. For any object, which is going to end up as a dimension table in a star scheme, it is strongly advised to identify an object of interest. Code tables however should be excluded. Code tables can only be OOI's if there is some explicit user functionality to maintain these tables in the system to be measured, which is probably not the case in data warehouse systems.

There can be numerous objects of interest in one operational data source, and I propose that for each of these a separate OOI's, a functional process must be identified to extract the data groups from the operational data source, to transform the data attributes the way is described by the metadata management system and to load the data groups into the staging area.

Identify data movements

For each functional process, which is executed in the staging area component, we identify one Entry to enter the datagroup into the functional process and one write to store it in the staging area component. Because of the fact that there will be some form of data cleansing in this functional process, there must be at least one Read data movement. There will probably be some kind of errorlogging, so we identify 1 standard exit (X) data movement for messages. Because of the fact that the Read is being done in another peer-component, additional data movements need to be identified to perform the exchange of data. A visualization of this can be seen in figure 8.

Note that the actual transformation of the data group according to the metadata rules is not being counted in COSMIC-FFP. The complexity of internal processing logic should be taken into account when deciding upon the most appropriate project delivery rate (PDR), when estimating the amount of time that the development of the system is going to take.

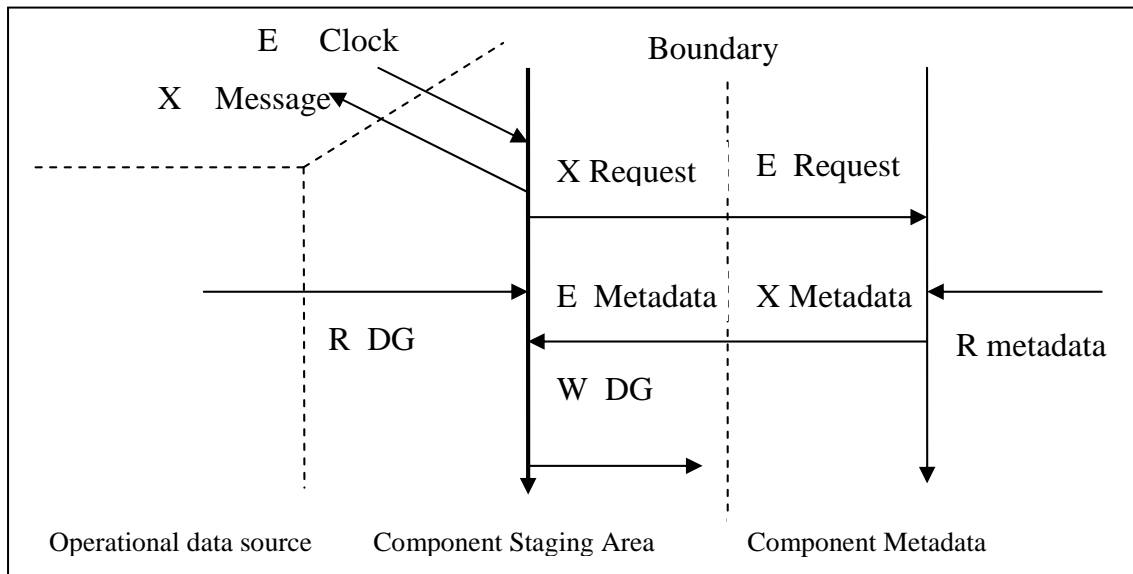


Figure 8: Data movements in the Staging Area

Data warehouse component

Identifying objects of interests (OOI), data groups and functional processes (FP)

In this component we find the functional processes that feed the data warehouse from the OOI's that are stored in the flat files in the staging area.

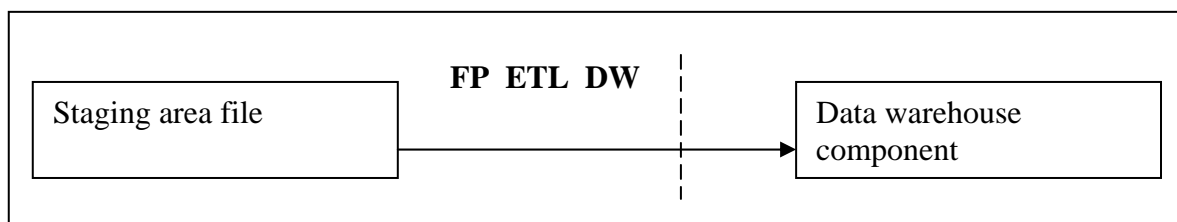


Figure 9: Functional processes within Data warehouse component

Again, for each OOI a separate functional process is to be identified, that extracts the data groups from the staging area file, transforms the data attributes the way described by the metadata management system, and then loads into the data warehouse component. The assumption has been made that this functional process runs in the datawarehouse component.

Identify data movements

There will be an entry for the start of this process (which is most probably a clock timing triggering event). Then there must occur a Read to read the datagroup in the staging area file. Because of the fact that this Read is taking place in another component, this is being counted as an Exit for the datawarehouse component (request for information), which becomes an Entry for the Staging area component. The staging area component then performs the Read and then there will be an Exit to move the results to the datawarehouse component. These results are then imported in the datawarehouse component with an Entry data movement.

In this functional process, there will be some kind of data cleansing or transforming as well, we must read the relevant metadata table, provided that these tables are an OOI for any of the

users mentioned before. Then, identify a Write to store the datagroup in the data warehouse. Then, identify the standard exit for messages. This is presented in figure 10.

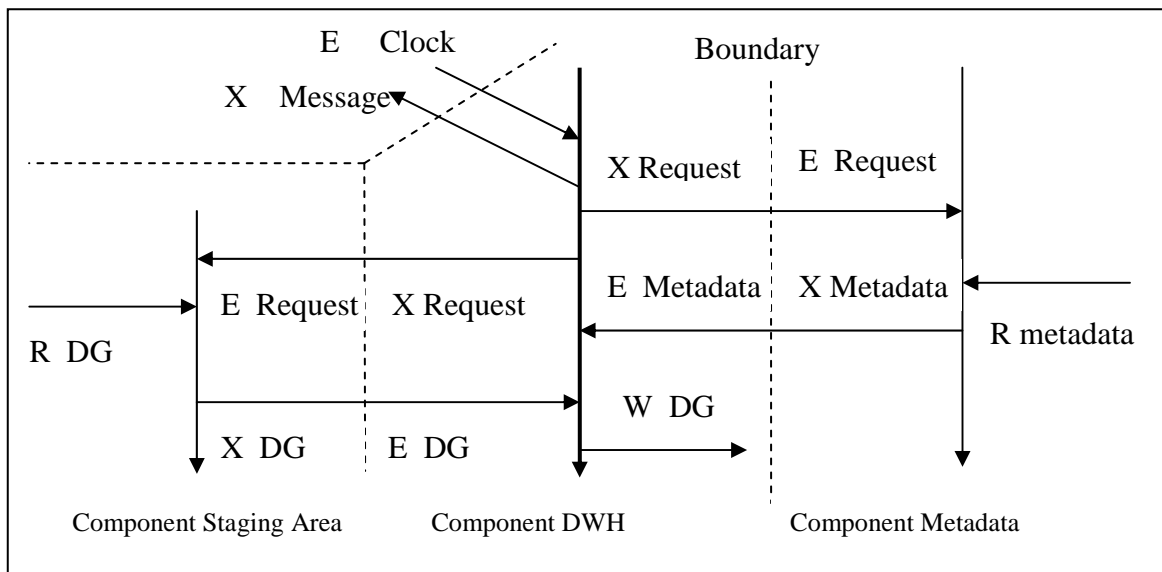


Figure 10: Data movements in the DWH component

Data mart component

Identifying objects of interests (OOI), data groups and functional processes (FP)

In this component we find the functional processes that feed the data marts from the OOI's that are stored in the data warehouse component.

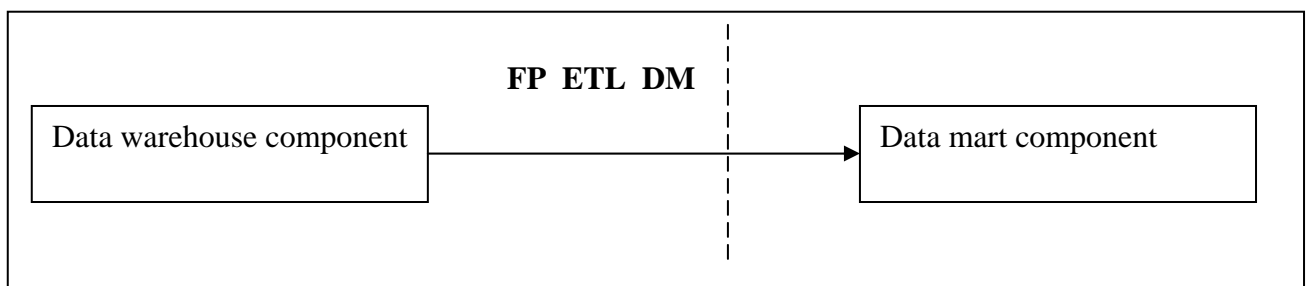


Figure 11: Functional processes within Data mart component

In this component, the data will be stored in a dimensional way, thus in star schemas. The OOI's in the data warehouse component will be loaded into the dimension tables. Additionally, the various fact tables are created. Each dimension table and each fact table should be identified to be an OOI.

For each dimension table there is one functional process to load the datagroup from the data warehouse component, to transform some data attributes and to load the datagroup into the data mart dimension table. Furthermore, we should consider how to size the different fact tables that are created within the process. It is important to realize that fact tables consist of aggregations and calculations in order to maximize system performance when end users ask for this data. With this in mind, the assumption that fact tables are created only for technical reasons is valid. Fact tables are tables in which transient data groups are stored, waiting to be

accessed by users. When the user actually asks for these facts, they will be counted as being separate exits, because of the fact that they are transient objects of interest to the user, but the storage of these facts are not being counted as writes.

Identify data movements

FP ETL DM: The identification of the data movements is the same as in the data warehouse component. So, there will be an entry for the start of this process (which is most probably a clock timing triggering event). Additionally, there will be a Read data movement to read the datagroup in the data warehouse component. In this functional process, there will be again some kind of data cleansing or transforming, so we have to read the relevant metadata table. Then, identify a Write to store the datagroup into the data mart component. Of course, the standard exit for messages should not be forgotten.

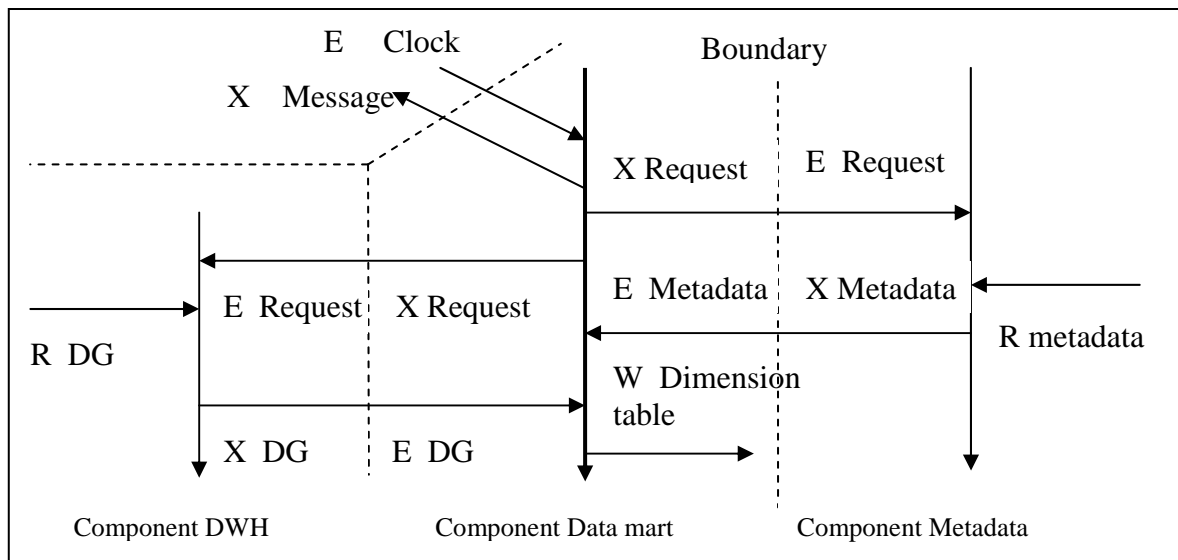


Figure 12: Data movements in the Data mart component

Business Intelligence component

Identifying objects of interests (OOI), data groups and functional processes (FP)

In this component, the information from the data marts is being made available for the end-user to extract.

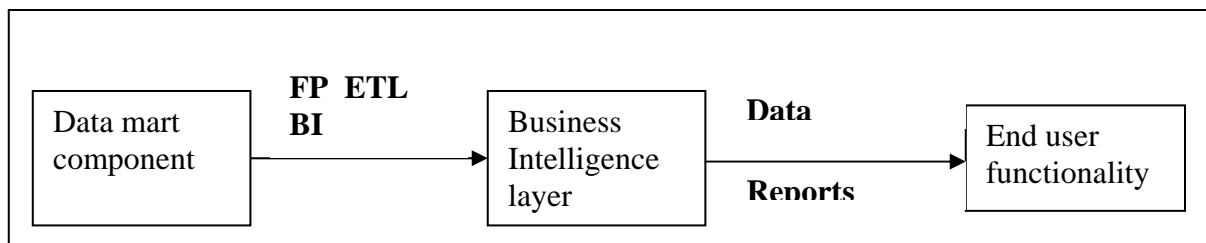


Figure 13: Functional processes within Business Intelligence component

FP ETL BI: The identification of the data movements is the same as in the previous components. So, there will be an entry for the start of this process (clock timing triggering

event). There will be a Read data movement to read the data group in the data mart component, at least one Read to read the relevant metadata table. Both Reads are being counted the way is proposed in the previous examples. Then, identify a Write to store the datagroup in the business Intelligence component. Of course, the standard exit for messages must be identified.

The end user functionality can be very varied. There can be all kinds of predefined queries, for which a functional process must be identified. There can be data mining tools, for which at least one functional process should be identified. There can also be functionality present with which the user can build his own queries. In this case, each unique possible query with its own processing logic should be identified as a distinct functional process.

Identify data movements

This is the area in which the COSMIC-FFP advantage of unlimited cfsu per functional process can provide its use in full. Each distinct function is in FPA counted as being one External Output, which can get a maximum of 7 function points. In COSMIC-FFP the number of cfsu is determined by the number of data movements, which means that there will be a Read and an Exit for each data group (dimension and/or facts) that is presented in one functional process. Furthermore, for each derived data attribute, an analysis should be made to establish whether or not it is a distinct transient OOI. If so, it counts for another Exit. This way, only the fact tables that are requested by the users will be identified as a transient OOI and thus an Exit will be counted for each fact that is presented to the user.

Note that 'drilling down' or 'rolling up' a dimension does not lead to another functional process, because only the selection values are different, while the selection criteria are still the same. The examples in the COSMIC-FFP Business Application Guideline can clarify how to do this [8].

Metadata management component

Identifying objects of interests (OOI), data groups and functional processes (FP)

In this component, the metadata administrator will have a number of functional processes to his disposal with which he can create new meta data rules, maintain existing rules or delete metadata rules. The number of objects of interest can vary widely between different data warehouse systems. Technical metadata, like update frequency or system versioning should not be considered an OOI, but for instance user profiles, access privilege files, data processing rules and use statistics can be possible OOI's for the metadata administrator [1]. Business metadata, like data dictionaries, Data on historical aspects, data on a data owner, can very well be objects of interests. For each OOI, at least one functional process should be defined, but more likely there will be functional processes to create, to update, to delete and to report the OOI.

Phase 3: Measurement phase

In the measurement phase, the number of cfsu (Cosmic functional size unit) is calculated, by adding the number of data movements per identified functional process. Each data movement equals 1 Cosmic functional Size Unit. When estimating the amount of time required for building the system, it could be wise to use a different PDR for the different components. This way, there is a distinction between the amount of time per cfsu that is required to build the business intelligence component (which is primarily GUI functionality)

and the amount of time per cfsu that is required to build all the database IO in the other components. Of course, the effort administration should also be ordered in a way that this distinction can be made while analyzing the results of completed projects.

6. Comments & Conclusions

There is little public knowledge available on functional size measurements of data warehouse applications. This is also reflected by a search through the ISBSG data [9] that registers only one data warehouse system, build in MS-ACCESS. In literature, a number of ways are proposed to measure data warehouse systems using FPA. However, differences between the data warehouse concepts and the transaction processing system, which was the basis for the development of functional sizing methods, make it hard to apply FPA on data warehouse systems. Although a number of attempts have been made, there is no consensus between authors how to map the data warehouse FUR's upon the FPA BFC's.

The fact that the different typical data warehouse components can be separately measured in COSMIC-FFP helps to make a more realistic measurement in relation to the viewpoint of FPA, in which only the functionality of the user interaction is measured. Besides, the Business Intelligence component, where the end-user data extraction and presentation functionality is located, can be measured more accurately with COSMIC-FFP, because of the fact that the number of size units a function can get is not limited. This brings more nuances to the measurement. Another advantage of applying COSMIC-FFP is that the logical data model is not being valued directly, like FPA does. In FPA, the number of function points heavily depends to the number of logical files involved, but in datawarehouse systems this leads to an overemphasis on the logical files, while the functionality to process the data is undervalued.

Datawarehouse experts seem to prefer COSMIC-FFP to size datawarehouse systems to be developed, but they object to the fact that the actual transformation of data, which they claim cost most of the effort, is not weighed in both measurement methods. Opposite to FPA, COSMIC-FFP offers the possibility to create local extensions of the method. It might be helpful to investigate in future studies the possibility of valuing the actual transformation in the ETL process, for instance by assigning values to the different data manipulation types that are normally considered to be part of the subprocess type (see figure 4).

Still, further investigations and case studies are required to decide upon the most accurate way to measure the functional size of datawarehouse systems.

References

- [1] Santillo, L., "Size & Estimation of data warehouse systems", in FESMA DASMA 2001 conference proceedings, Heidelberg (Germany), May 2001.
- [2] Schipper, H., "Wat kost dat nou, zo'n data warehouse?", Database magazine, June 2005, pp. 10.
- [3] Abran, A. e.a., "COSMIC FFP Measurement manual 2.2", Jan 2003, <http://www.lrgl.uqam.ca/cosmic-fpp>.
- [4] Chaudhuri, S. and Dayal, U., "An Overview of Data Warehousing and OLAP Technology", ACM Sigmod record vol. 26 (1), 1997, pp. 65-74.
- [5] Dekkers, C., "Function Point Analysis for data warehouses", workshop course, Quality Plus Technologies, Inc., <http://www.qualityplustech.com>, 2005.
- [6] Eveleens, R., "FPA-Telrichtlijn voor Data warehousing", Nesma conference, 2006.
- [7] ISO/IEC (1998), "14143 Software Engineering - Software Measurement - Functional Size Measurement - Part 1: Definitions of concepts".

- [8] Lesterhuis, A. and Symons, C., "Guideline for sizing business applications using COSMIC-FFP", Version 1.0, December 2005.
- [9] ISBSG database, version 9.
- [10] Inmon, W.H., "What is a Data Warehouse?", Prism, Volume 1, Number 1, 1995.
- [11] Method Update Bulletin 1, Proposed Improvements to the Definition and Characteristics of a software 'Layer', 2004.