SIZE & ESTIMATION OF DATA WAREHOUSE SYSTEMS

Luca Santillo
(luca.santillo@gmail.com)

Abstract

Data Warehouse Systems are a special context for the application of functional software metrics. The use of an unique standard, as Function Point, gives serious comparability issues with traditional systems or other paradigms, in terms of both numerical size and implementation effort estimation. Peculiar guidelines are therefore necessary in order to identify the user view, the software boundaries, the data and the transactional components of such systems. Particularly, the boundary identification may strongly affect the measurement result for a data warehouse project; consequently, one can find huge, unacceptable deviations in the estimation of effort, time and cost for the given project.
This paper shows the substantial differences between "traditional" software and data warehouse systems, the main guidelines that one can use when measuring the latter, and peculiar considerations for differentiating the effort estimation by measured element types.
The depicted case studies highlight the fundamental relevance of the concept of "layer", as explicitly stated by the most recent evolutions in the functional metrics field (COSMIC Full Function Point) in evaluating those functions which are seemingly transparent to the final user, but which cannot be neglected when estimating the implementation effort of the measured system.

Keywords: functional measurement, effort estimation, data warehouse.

INTRODUCTION

Software functional measurement methods aim to provide an objective, technology-independent, user-significant measure of the size of software systems. IFPUG Function Point method is a set of practices intended to be applied to every domain or application typology. Despite of their generality, the IFPUG counting practices are not always easy to apply in real or innovative environments. Apart from possible enhancements to the expression of the practices, the key concept is that the recognizability of the functional sizing elements of a software systems depends on the system user view, and this point of view can widely change from one domain to another. It's therefore necessary to assess the correct approach to the sizing of a given system typology (data warehouse, in our case), by means of providing domain-specific counting guidelines. The proposed approach should not be considered as a different sizing method, but rather as an "instantiation" of the general method concepts in a specific environment or domain.

On the other hand, if we use a specific measurement approach for the given domain, we have to face the fact that effort estimation (of development or enhancement activities) from this measurement cannot be obtained from general models (unless we accept the strong risk of large estimation errors). Therefore, an "instantiation" of a generic effort model is to be used.

DATA WAREHOUSE DEFINITIONS

Data Warehouse System

A data warehouse contains cleansed and organized data that allows decision makers to make business decisions based on facts, not on intuition; it includes a repository of information that is built using data from the far-flung, and often departmentally isolated, systems of enterprise-wide computing (operational systems, or "data sources"). Creating data to be analysed requires that the data be *subject-oriented, integrated, time referenced* and *non-volatile*. Making sure that the data can be accessed quickly and can meet the *ad hoc* queries that users need requires that the data be organized in a new database design, the s*tar (schema) or multidimensional data model*. See Tab. 1 for an overview of peculiar aspects of data warehouse systems, versus operational (transactional) systems.

	Transaction Processing	**Data Warehouse**
Purpose	Run day-to-day operations	Information retrieval and analysis
Structure	RDBMS optimised for Transaction Processing	RDBMS optimised for Query Processing
Data Model	Normalised	Multi-dimensional
Access	SQL	SQL, plus Advanced Analytical tools.
Type of Data	Data that runs the business	Data to analyse 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

Table 1. Data Warehouse systems versus transactional systems.

Enterprise Data Warehouse (EDW)

An EDW contains detailed (and possibly summarized) data captured from one or more operational systems, cleaned, transformed, integrated and loaded into a separate subject-oriented database. As data flows from an operational system into an EDW, it does not replace existing data in the EDW, but is instead accumulated to show a historical record of business operations over a period of time that may range from a few months to many years. The historical nature of the data in an EDW supports detailed analysis of business trends, and this style of warehouse is used for short- and long-term business planning and decision making covering multiple business units.

Data Mart (DM)

A DM is a subset of corporate data that is of value to a specific business unit, department, or set of users. This subset consists of historical, summarized, and possibly detailed data captured from operational systems (*independent data marts*) , or from an

EDW (*dependent data marts*). Since two or more data marts can use the same data sources, an EDW can feed both sets of data marts and information queries, thereby reducing redundant work.

Data Access Tools (OLAP, On-line Analytical Processing)

OLAP is the technology that enables users to access the data "multidimensionally" in a fast, interactive, easy-to-use manner and performs advanced metric computations such as comparison, percentage variations, and ranking. 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*.

Metadata

Simply stated, metadata is data about data. Metadata keeps track of what is where in the data warehouse.

Extraction, Transformation, & Loading (ETL)

These are the typical phases required to create and update a data warehouse DB:

- In the *Extraction phase*, operational data are moved into the EDW (or independent DM). The operational data can be in form of records in the tables of a RDBMS or flat files where each field is separated by a delimiter.
- *Transformation phase* changes the structure of data storage. The transformation process is carried out after designing the datamart schema. It is a process that ensures that data is moved into the datamart, it changes the structure of data suitable for transaction processing to a structure that is most suitable for DSS analysis, providing a cleaning of the data when necessary, as defined from the data warehouse manager.
- *Loading phase* represents an iterative process. The data warehouse has to be populated continually and incrementally to reflect the changes in the operational system(s).

Dimensions

A dimension is a structure that categorizes data in order to enable end users to answer business questions. Commonly used dimensions are Customer, Product, and Time. The data in the structure of a data warehouse system has two important components: dimensions and facts. The dimensions are products, locations (stores), promotions, and time, and similar *attributes*. The facts are sales (units sold or rented), profits, and similar *measures*. A typical dimensional cube is shown in Fig. 1.

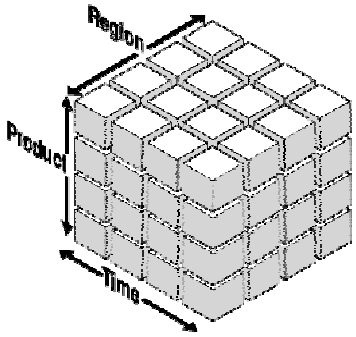

Figure 1. Sample Dimensional Cube.

Star Schema

Star Schema is a data analysis model analogue to a (multi)dimensional cube view. The center of the star is the *fact (or measure) table*, while the others are *dimensional tables.* Fig. 2 shows an example of star schema.

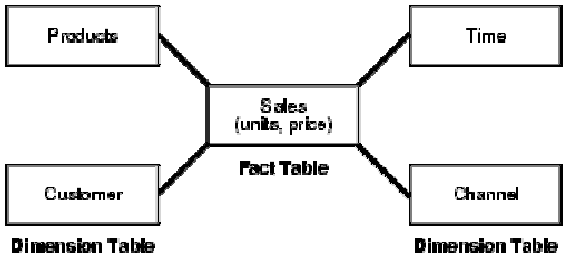

Figure 2. Example of Star Schema.

Specifically, dimension values are usually organized into hierarchies. Going up a level in the hierarchy is called *rolling up* the data and going down a level in the hierarchy is called *drilling down* the data. For example, within the time dimension, months roll up to quarters, quarters roll up to years, and years roll up to all years, while within the location dimension, stores roll up to cities, cities roll up to states, states roll up to regions, regions roll up to countries, and countries roll up to all countries. Data analysis typically starts at higher levels in the dimensional hierarchy and gradually drills down if the situation warrants such analysis.

FUNCTIONAL MEASUREMENT DEFINITIONS

Functional Size

The size of a (software) system as viewed from a logical, non-technical point of view. It is more significant to the user than physical or technical size, as for example Lines of Code. This size should be shared between users and developers of the given system.

IFPUG Function Point

IFPUG Function Point measure is obtained by summing up the data and the transactional functions, classified as Internal Logical Files, External Interface Files, and External Inputs, Outputs, or Inquiries, with respect to the application boundary, which divides the measured system from the user domain(or interfaced systems). See Tab. 2 for an overview of the numerical weights (here "complexity" depends depends on logical structure of each element, in terms of quantities of logical attributes and referenced files contained or used by files or transactions).

	Low Complexity	Average Complexity	High Complexity
ILF	7	10	15
EIF	5	7	10
EI	3	4	6
EO	4	5	7
EQ	3	4	6

Table 2. Function Point elements' weights.

COSMIC Full Function Point

COSMIC Full Function Point has been proposed as a superset of functional metrics, which provides wider applicability than the IFPUG method. Its key concepts are the possibility of viewing the measured system under different linked layers (different levels of conceptual abstraction of the system functions) and the possibility to extend its practices with "local extensions".

FUNCTIONAL SIZE OF DATAWAREHOUSE SYSTEMS

IFPUG official documentation doesn't provide specific examples for counting data warehouse systems; on the other hand, this kind of system is growing in importance and diffusion among private and public companies. We therefore need some guidelines, especially if we consider the special user view and consequent specific data models for this system type.

A generic data warehouse system can be viewed as made of three segments: Data Assembling (see ETL above), System Administration (see also Metadata above), and Data Access (see OLAP above).

Type of Count

Determining the type of count (Development project, Enhancement Project, or Application) is usually easy, and doesn't require specific guidelines for data warehouse systems. We just remind that adding or changing functionality of a given system could be considered as one or more (development and enhancement) projects, depending on which system boundaries are firstly identified.

User View

Many figures contributes to constitute a data warehouse user:

- ETL procedures administrator,
- DB administrator,
- OLAP (or other access means) administrator,
- final user (who have access to the data warehouse information),
- any system providing or receiving data to or from the data warehouse system (for example, operational systems which automatically send data to the ETL procedures).

Application Boundary

When considering the separation between systems in the data warehouse domain, the application boundaries should:

- be coherent with the organizational structure (e.g. each department has its own DM)
- reflect the project management autonomy of the EDW with respect to any DM,
- reflect the project management autonomy of each DM with respect to any other.

The following picture (Fig. 3) shows the proposed approach to the boundary question in a data warehouse context. Note that the shown boundaries are orthogonal to the segmentation by phase of the data warehouse system (ETL, Administration, Access).

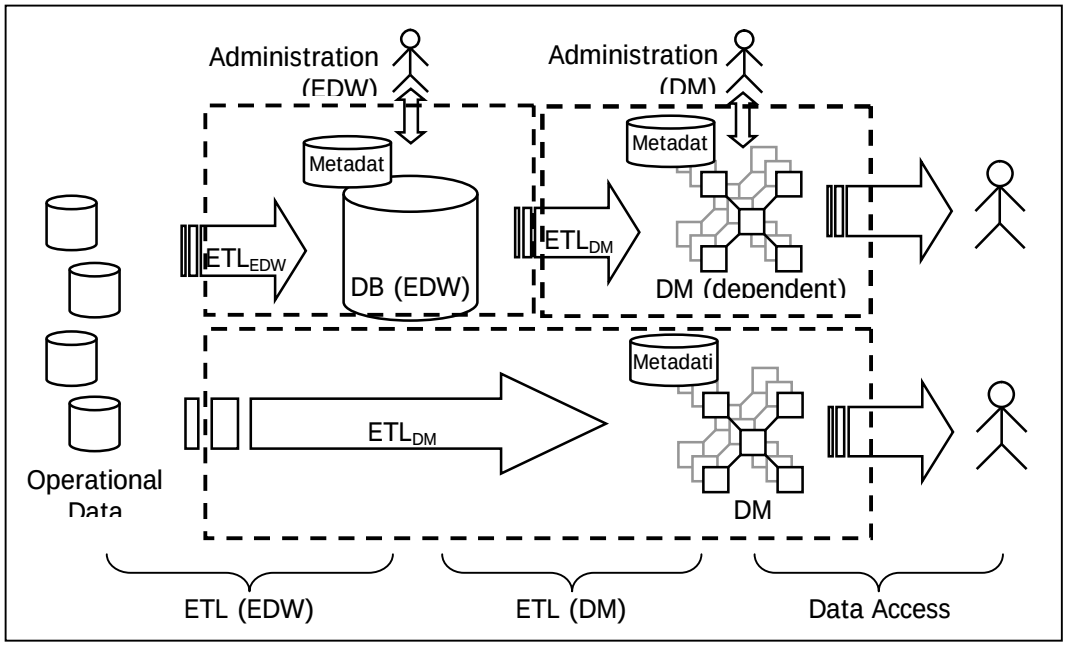

Figure 3. Boundary scheme for EDW, dependent DM, and independent DM.

Comments on boundaries

Note that, as stated also by the IFPUG Counting Practices Manual, some systems could share some functionality, and each of them should count those functions. For example, 2 or more (dependent / independent) DMs' can make use of the same external source files (EDW / operational) in order to load their own data. While counting these shared functions for each system which uses them, we should not ignore some reuse consideration, when deriving the effort estimation for each system development or enhancement project.

Boundary re-definition should be performed only in special cases, as the merge of 2 DMs' into one, or the split of 1 DM into more than one system. In doing such a re-definition, we have to mark some functions as deleted without effort (in the merge case), or as duplicated without effort.

Data Functions

Operational source data

These are EIFs' for the EDW or the independent DM which use them in the ETL segment. While the separation into distinct logical files is performed from the point of view of the operational system which provides and contains them as its own ILFs', their content, in terms of Data Element Types, and Record Element Types, should be counted from the point of view of the target system. Note that simple physical duplicates on different areas are usually not counted as different logical files.

A special case of the ETL procedure is when the operational system provides by its own procedures the information to the EDW (or independent DM); in this case, no EIF is counted for the latter, since we have External Outputs sending out of the source system the information required, and not the target system reading and collecting the data.

Data warehouse internal data - Star schema data model

While counters are provided with sufficient guidelines and example for entity-relationship data models, we have to face the case of star schema data models, which correspond to the multidimensional cube views.

Since the fact table is not significant to the data warehouse user, without its dimensional tables, and vice versa, we suggest the strong guideline that each "logical" star is an ILF for the EDW or DM being counted. Each (fact and dimensional) table is a Record Element Type for such a logical file. IN analogy with this, each "logical" cube is an ILF, with N+1 RET, where N is the number of its dimensions (the axes of the cube).

In case of the so-called snow-flake schema, where the hierarchical dimensions are exploded into their levels (e.g. month – quarter - year), the second order tables do not represent other RETs', since the counted RET is for the whole dimension ("time" in the cited example).

The DETs' of each hierarchy are only two, dimension level and value (e.g. "time level", which can be "month", "quarter", "year", and "time value", which can be "January", "February", …, "I", "II", …, "1999", "2000", …, and so on).

Other attributes in the tables, apart from those who implement a hierarchy , are counted as additional DETs' for the logical file. A special case of data warehouse systems attributes is that of pre-derived data, or data which are firstly derived in the ETL phases, then recorded in the file, and finally accessed by the final user, in order to provide the maximum performance. A logical analysis should be carried in order to distinguish the case when the (final) user recognises these data as contained in the files, and then only retrieved by inquiries, from the case when the user is not aware of such a physical processing, and considers the data as derived online by the required output process.

Metadata

Technical metadata, as update frequency, system versioning, physical-logical files mapping, are not identifiable as logical files. Since the data warehouse administrator is one of the figures which constitute the general system user, some metadata can be recognized and counted as logical files; example are:

- User profiles file
- Access Privileges file
- Data processing rules file
- Use Statistics file

Business metadata are good candidates for being counted as logical files; examples are:

- Data dictionary(what is the meaning of an attribute)
- Data on historical aspects (when a value for an attribute was provided)
- Data on the data owner (who provided a value for an attribute)

Transactional Functions

ETL: we suggest the strong guideline that the overall procedure of reading external source files, cleaning and transforming their contents, reading eventually metadata, and loading the derived information in the target system is a unique process from the data warehouse user point of view; therefore we have only one EI for each target identified ILF. DETs' of such an EI should be all the attributes which enters the boundary of system being counted, plus the eventual output attributes or data, such as messages to the user for error or confirmation.

Administration: The administration segment contains traditional processes, such as the management transactions for creating, updating, deleting, and viewing metadata.

Access: The main functions of the access segment are those who let the user consult information from the data warehouse; such processes are counted as EOs' or EQs', depending on the presence of derived data. Therefore, we have at least 1 process (usually EO) for each identified "logical star" of the data warehouse DB. Note that drilling down o rolling up the same star is equivalent to retrieving the same data, just using different "levels" in the dimensional hierarchies – which are all DETs' of the same star – so different levels of the view are counted only once, as they are the same logical output.

The drill down trigger itself is usually provided by common OLAP tools as a listbox on every "drillable" attribute. Such mechanism is counted as a low complexity EQ (for each distinct attribute of each distinct star), while the productivity coefficient for such a process will strongly reduce its impact.

Function Taxonomy Classes

In order to support the effort estimation, the data and transactional functions should be labelled depending on their role in the data warehouse system being measured. The classes are: ETL (Extraction, Transformation & Loading), ADM (Administration), ACC (Access). Tab. 3 provides examples of such a classification.

Type	Where	Examples
ILF_{ETL}	EDW, DM	• EDW DB logical files • Independent DM DB logical files • Dependent DM DB logical files, when logically distinct from the EDW DB logical files
ILF_{ADM}	EDW, DM	Metadata, significant LOG files, statistics
EIF_{ETL}	EDW, DM	Operational DB logical files
EIF_{EDW}	Dependent DM	EDW's ILFs', when accessed by ETL or Access procedure
EIF_{ADM}	EDW, DM	• Significant support files • Externally maintained metadata
EI_{ETL}	EDW, DM	1 EI for each identified ILF_{ETL}
EI_{ADM}	EDW, DM	Create, update, delete metadata
EO_{ADM}	EDW	View metadata (with derived data)
EQ_{ADM}	EDW	View metadata (without derived data)
EO_{ACC}	DM	1 EO for each identified ILF_{ETL}
EQ_{ACC}	DM	1 EQ for each identified ILF_{ETL} which has no corresponding EO_{ACC}, i.e. view without any derived data
$EQ_{LISTBOX}$	DM	Drill-down triggers, any other List Boxes

Table 3. Function Types Taxonomy.

Value Adjustment Factor (VAF)

At the present moment, a specific ISO Working Group is examining the candidates for a standard software functional measurement definition; one preliminary result is that the 14 General System Characteristics, which constitute the VAF, should not be used; therefore, we neglect VAF, or, that is equivalent, we consider its value equal to 1 in any counting case.

Final Function Point Formulas

Standard formulas are used without specific recommendation. We only recall the use of the proposed taxonomy; that means that, besides total of FP, we have to provide the complete list of different functions, depending on their classes. Since we always assume a VAF = 1 for data warehouse systems, the final count formulas are slightly simplified.

EFFORT ESTIMATION FOR DATAWAREHOUSE SYSTEMS

Data warehouse systems productivity factors

The main peculiar productivity aspects of data warehouse systems are:

- Many data and transactional functions are cut (flatten) because of the limit of "high complexity"of the IFPUG model;
- Internal and external reuse can be very significant;
- Data warehouse and OLAP tools and technology positively impact the implementation productivity, while the analysis phase can be very consuming
- Some segments (as Access) are more impacted by the use of tools.

All these factors lead us to consider an innovative structured approach to the utilization of Function Point measure in the software effort estimation process, when applied to data warehopuse systems. Instead of putting the mere total number of FP for a project in a benchmarking regression equation, we found by empirical and heuristical research some steps which provides an "adjusted" number, that can be seen as "FP-equivalent" for effort estimation purpose. Of course, we should keep the original counted FP as the size of the system in terms of the user view, while this "FP-equivalent" is a more realistic number to use in a software effort estimation model. The coefficients proposed in the following are to be multiplied with the original FP number of the corresponding counted function. Only cases different from unitary (neutral) adjustment are shown.

1. Adjustment by intervention class (only specific classes are shown)

	DEV_{EDW}	ENH_{EDW}	DEV_{DM}	ENH_{DM}
Class	**Coefficient**	**Coefficient**	**Coefficient**	**Coefficient**
ILF_{ETL}	1	1	$\left(1+\frac{RET-4}{4}\right)$	$\left(1+\frac{RET-4}{4}\right)$
EI_{ETL}	$\left(2+\frac{FTR-3}{3}\right)$	$\left(2+\frac{FTR-3}{3}\right)$	$\left(1+\frac{FTR-4}{3}\right)$	$\left(1+\frac{FTR-4}{3}\right)$
EO_{ACC}	$\left(1+\frac{FTR-4}{4}\right)$	$\left(1+\frac{FTR-4}{4}\right)$	1	1
EQ_{ACC}	$\left(1+\frac{FTR-3}{3}\right)$	$\left(1+\frac{FTR-3}{3}\right)$	1	1

Table 4. Adjustment coefficients by intervention class.

2. Adjustment by reuse (NESMA-like model)

2a. Development (both EDW & DM)

Consider each function class in the given count (e.g. all the ILF_{ETL}, then all EIF_{ETL}, and so on). For each distinct function class:

a) Assign a reuse coefficient of **0.50** to each function (except the 1st) of the set of functions which share:
   - 50% or more DETs', **and** 50% or more RETs' or FTRs'.

b) Assign a reuse coefficient of **0.75** to each function (except the 1st) of the residue set of functions which share:
   - 50% or more DETs', **but** less than 50% RETs' or FTRs';
   - less than 50% DETs', **but** 50% or more RETs' or FTRs'.

c) Assign a reuse coefficient of **1.00** (neutral) to the residue.

The "1st function" means the function in the given class with highest functional complexity, highest number of DETs', highest number of RETs' or FTRs'. The percent values of DETs', RETs', and FTRs', are determined with respect to this "1st function".

In the special case of CRUD transactions sets in Administration segment, i.e. Create, Read, Update, and Delete of generic file type, assign a uniform 0.5 adjustment to each transaction in the unique identified CRUD.

2b. Enhancement (both EDW & DM)

Added Functions
Act as for Development.

Internally Changed Functions (i.e. added, changed, deleted DETs', RETs', or FTRs')

$Reuse_{ENH}$		DET%			
		£ 33%	£ 67%	£ 100%	> 100%
RET% or FTR%	£ 33%	0.25	0.50	0.75	1.00
	£ 67%	0.50	0.75	1.00	1.25
	£ 100%	0.75	1.00	1.25	1.50
	> 100%	1.00	1.25	1.50	1.75

Table 5. Reuse coefficients for Internlly Changd Functions.

where the percent values are given by comparing the number of DETs', RETs', FTRs' which are added, modified, or deleted, with respect to their pre-enhancement quantities.

Type Changed Functions (i.e. ILF to EIF, EQ to EO, etc.)
Assign an adjustment reuse coefficient of 0.4.

Mixed Cases
If a function is changed in both internal elements and type, assign the higher of the two adjustment coefficients from the above. For transactions, note that changes in the user interface, layout, or fixed labels, without changes in the processing logic, are not considered.

Deleted Functions
Assign an adjustment reuse coefficient of 0.4.

3. Adjustment by technology (only applied to DM projects, Access segment)

	DEV_{DM}	ENH_{DM}
Class	**Coefficient**	**Coefficient**
EO_{ACC}	0.5	0.5
EQ_{ACC}	0.5	0.5
$EQ_{LISTBOX}$	0.1	0.1

Table 6. DW technology adjustments.

Effort Estimation

After we obtain the "FP-equivalent" frm the previous adjustment, we can put its value in a benchmarking regression equation, as the following, which has been obtained (by filtering on several sample attributes) from the ISBSG Benhmark:

Avg.Eff = 13.92 x FP-equivalent - 371.15

Note that this equation is just an example; more precise estimations can be obtained only by creating a "local benchmark" for the given company, project team, or department. However, one further step is still to be made: specific productivity adjustment of the average effort estimate.

Specific productivity adjustment

This last step is carrie dout by means of the well-known COCOMO II model; we recall that only some factors of the original COCOMO II model are to be used, since, for example, the REUSE factor is already explicitly considered in the previous steps, when calculating the FP-equivalent.

The final effort estimation is therefore:

$$Effort = \overline{Effort} \cdot \prod_{i=1}^{N} CD_i$$

where:

- $\overline{Effort}$ is the Average Effort from the previous step, based on ISBSG or equivalent benchmark.
- CD_i is the coefficient of the ith COCOMO II Cost Driver.

The Cost Driver considered in the actual research are:

- RELY (Required software reliability)
- CPLX (Product complexity)
- DOCU (Documentation match to life-cycle needs)
- PVOL (Platform volatility)
- ACAP (Analyst capabilities)
- PCAP (Programmer capabilities)
- AEXP (Applications experience)
- PEXP (Platform experience)

Readers should refer to the original COCOMO II documentation for exact values of the levels of these drivers.

COMMENTS & CONCLUSIONS

Serious testing of the proposed approach is being carried out at the moment. Further developments will surely come from the adoption of the COSMIC Full Function Point "layer" concept, which will is able to take into account the impact of some specific "segments" of data warehouse systems, as for example a detailed model of the ETL (Extraction, Transformation & Loading) phases, in order to improve the effort estimation precision. Moreover, research pointed out the inadequacy of cut limits in the complexity levels of IFPUG Function Point method, as already shown by the ISBSG research: some rangs in the complexity matrices should be revised or extended. Note that in COSMIC Full Function Point method, there no such artificial cut-off.

Another issue that is to be faced is the creation of specific benchmark for data warehouse domain and technology, since this typology is going to play an relevant role in the future of public and private companies, which have to manage more and more information in less and less time than ever.

REFERENCES

- Baralis E., "**Data Mining**", Politecnico di Torino, 1999
- **COCOMO II Model Definition Manual rel. 1.4**, University of Southern California, 1997
- **COSMIC Full Function Point Measurement Manual, Version 2.0**, Serge Oligny, 1999
- Hennie Huijgens, "**Estimating Cost of Software Maintenance: a Real Case Example**", NESMA , 2000
- Dyché J., "**e-Data: Turning Data into Information with Data Warehousing**", Addison-Wesley, 2000
- **IFPUG Function Point Counting Practices Manual, Release 4.1**, IFPUG, 1999
- **ISBSG Benchmark, Release 6**, ISBSG, 2000
- Torlone R., "**Data Warehousing**", Dipartimento di Informatica e Automazione, Università di Roma Tre, 1999