BUSINESS INTELLIGENCE PERFORMANCE AND CAPACITY IMPACT

Codrin POPA[1]

Complexitatea actuală a diverselor organizaţii având impact asupra Sistemelor Informatice arată că dezvoltarea unor sisteme de tip Business Intelligence trebuie inclusă în cadrul mai larg al Ingineriei de Sistem. Proiectul, în totalitatea sa trebuie să se focalizeze în definirea cerinţelor şi livrabilului final şi să stabilească componentele specifice ce descriu sistemul pe baza unui ciclu de rafinări succesive. Arhitectura software clasică trebuie regândită pentru obţinerea unei abordări complexe în care cerinţele funcţionale şi non-funcţionale sunt depedente. Bazandu-se pe experienta autorului obtinuta in numeroase implementari de tip Business Intelligence, acest articol ia in considerare importanta unei arhitecturi de sistem optimizata si sublineaza relationarea modelului de date cu performanta si capacitatea Sistemelor de tip Data Warehouse.

The actual complexity of various organizations with impact in Information Systems reveals that the development of Business Intelligence Systems must be integrated into the wider concept of the Systems Engineering. The whole project should be focus in defining all the requirements and outputs and to establish the specific pieces that compose the systems in a cycle of successive refinements. The classical software framework must be redesign in order to enable a complex approach where the functional and non-functional requirements are inter related. Based on the experience achieved in several Business Intelligence implementations, this article considers an optimal system architecture and emphasizes the proper linkage between data model and capacity and performance of the Data Warehouse Systems.

Keywords: Business Intelligence, Systems Engineering, Performance, Capacity, Data Model

1. Introduction

One of the keys of management processes is the decision that can have two forms, namely the act itself and the decision making process. The difference between these two approaches is done by the complexity of the environment that involves a decision. Management decision is the choice of action for one or more goals. R.T. Clemen [1] remarks that the decision context is the framework of events that determine the set of objectives that actually matters (and nothing more

[1] IT Architect, "Global Business Services" Department, IBM, Romania, codrin.popa@ro.ibm.com

or less) for a decider at the time of drafting the decision, even if the values remain relatively unchanged.

Even if some simple patterns could be achieved based on generic or structured data, a depth and systematic analysis, using a specific class of information systems in order to assist managerial decision – Decision Support Systems (DSS), could show more complex patterns and trends. The objective of a DSS is to minimize the effects of the limitations and constraints in solving of a large and complex area of decisional problems by implementing automatic processes of decision support [2].

From a historical perspective P. Keen and S. Morton [3] allege that the concept of decision support systems evolved from the theoretical studies of organizational decision making done at the Carnegie Institute of Technology during the late 1950s and early'60s and the technical work on interactive computer systems, mainly carried out at the Massachusetts Institute of Technology in the 1960s.

Most definitions of decision support systems have shown the integration of those systems into the organization framework. Data warehouse system is only one component of decisional support systems. A modern data driven decision support system – Business Intelligence System is consisting of four main components: the data storage component (data warehouse), the extraction and filtration component, data query and analysis component and data presentation and visualization component. The business needs to know what is happening right now, faster, in order to determine and influence what should happen next time. The goal of business intelligence (BI) systems [4] is to capture (data, information, knowledge) and respond to business events and needs better, more informed, and faster, as decisions.

The complexity of such systems involves the integration into the broader vision of system engineering. BI systems could be decomposed into a set of interrelated components based on a flexible and unified architecture that considers all the aspects and process of the information system.

Systems engineering is a robust approach to the design, creation, and operation of systems. In simple terms, the approach consists of identification and quantification of system goals, creation of alternative system design concepts, performance of design trades, selection and implementation of the best design, verification that the design is properly built and integrated, and post-implementation assessment of how well the system meets (or met) the goals [5] .
It represents [6] an interdisciplinary approach and means to enable the realization of successful systems. In the system engineering a system is a set of interrelated components which interact with one another in an organized fashion toward a common purpose. Every system exists in the context of a broader super system, a collection of related systems. It is in that context that the system must be judged.

2. Business Intelligence Systems Lifecycle Processes

The term life-cycle depends on the context and the scope. There are several definitions including Software Development Lifecycle or Systems Development Lifecycle.

P. Rob and C. Coronel allege that the processes used to create and define information systems are known as systems development. The system development lifecycle traces the history (life cycle) of an information system [7].

In conformity with IEEE Standard for Developing Software Life Cycle Processes [8] a Software Life Cycle Process represents the project-specific description of the process that is based on a project's software life cycle (SLC) and the Organizational Process Assets (OPA). The components of the SLCP consist of a number of activities organized into activity groups: Project Management, Pre-Development, Development, Post-Development and Integral.

Depending on the user requirements a number of software development life cycles (SDLC) models have been developed: sequential, incremental and evolutionary.

The sequential model is the first SDLC pattern, it is easy to understand and is based on the assumption that the requirements can be understand and defined before designing the solution. There are two basic sequential models: the waterfall and V-model.

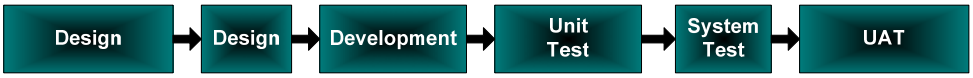

Fig. 1 - The Waterfall Model

The pure sequential models are very expensive because if there are errors in early stages they will be discovered in the last stages thus involve a serious impact on the whole project. In the V-model each task is verified by a corresponding test task.

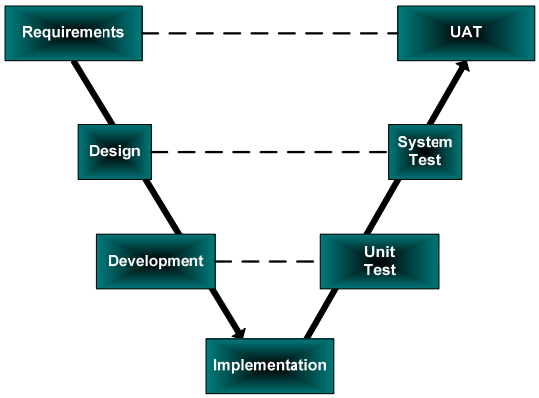

Fig. 2 - The V-Model

The incremental development lifecycle model was created based on the assumption that for reducing and splitting the risks associated with complex developments and implementations another technique must be used - decomposing the project into small pieces.

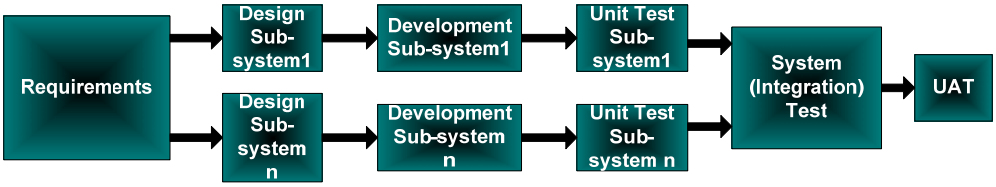

Fig. 3 - The Incremental Model

In the evolutionary models the requirements user needs and system requirements are partially defined up front, then are refined in each succeeding build / release. Sometimes this approach is chosen if the amount of functionality required by the business is too expensive to be implemented in a single release. Only prototypes are developed where all the requirements can not be known in advance. The spiral model is one of the most known designed evolutionary models.

Theoretical, a data warehouse/business intelligence system must be implemented based on the V-model. This is wrong. The complexity of business intelligence – data warehouse systems is very high so it is better to consider from the beginning various foreseen aspects that could impact the overall cost and increase the initial investment of the project but even with a good analysis there are still remaining a large numbers of variables to be considered. The correct life cycle approach is the incremental iterative approach, with different iterations between the requirements phase and development phase, resulting in successive refinements of the architectural decisions, parameters or even requirements.

Based on the various implementations, it result that it is impossible to consider a data warehouse as a black box software system and is more or less a never ending story. After the first released of the project is finished other application will start based on the information 'greed' of the organizations.

S.S. Iliescu [9] remarks that a model is a mathematic representation of measures dependency. If the dependence corresponds to a physical reliable process we can consider a systemic model. The systemic model represents a causality relation between measures. There are input and output measures.

I emphasized that business intelligence development lifecycle shouldn't be considered a common v-model; in fact it is a system with continuous and specific dynamic. A data warehouse can be considered a complex cyclic systemic model.

3. Business Intelligence Layers Dependency

A data model is a graphical, simplified representation of a real world complex environment. It is an abstraction of a complex data structure.

Within the database environment, a data model represents data structure and their characteristics, relations, constraints, transformations and other constructs with the purpose of supporting a specific problem domain [7]

ANSI Standards Planning and Requirements Committee propose a framework based on three levels of abstraction [10]: the conceptual level is based on the entire organization view regarding the database; the internal level matches the conceptual level to the specific implementation model; the external model is a subset of internal model and represents the users' view of the data.

In order to translate the inputs realized into the analysis phase for a complex, organized and scalable system that must fulfill organization requirements, the specific data models can be split into the following categories:

a) Conceptual model is the highest level of abstraction for business organization environment in order to produce a type of conceptual schema or semantic data model of a system. The entity-relationship diagram is the widely used model; it was introduced by Peter Chen [11] in 1976 and represents an informal model for real world representation.

b) Logical model it is between conceptual and physical model in order to translate the functional requirements into the physical requirements.

c) Physical model is describing the database access and storage systems inputs.

Theoretically it seems that this kind of formalization is synchronized with software development life cycle: analysis, design, development and implementation.

To consider all the aspects and process of the data warehouse information system and even the correlation between functional and non-functional requirements with focus on logical model and performance interrelations, I propose an inter-layers schema that will enhance the functionality of the standard ANSI/X3/SPARC framework layers (external, conceptual and internal model) and will show the interdependencies and influences between information requirements, logical design and performance and capacity of the system.

4. Case study – Reporting Component over the BI Performance and Capacity

The primary objectives of the Performance and Capacity Management processes are to ensure that adequate capacity and sufficient performance exist in the IT infrastructure to support day-to-day operations in a cost-effective manner.

W. Inmon [12] defines the data warehouse system as a database environment integrated from various locations with different formats, subject-oriented in order to allow answers to different compartments, time variant having a historical structure unlike the traditionally operational systems and non-volatile, the data from these databases are added and represent a history of organization activity. The dichotomy between transactional systems and business intelligence systems is shown by the data types existing in such systems. While the transactional systems load current transactions and keep a relative small log, a data warehouse works with large volumes of historical data which are then summarized.

There must be distinguished two types of operations on a data warehouse with different impact: extract, transform and load (ETL) batch jobs and queries.

Queries are the primary mechanism for retrieving information from a data warehouse database and consist of questions presented to the database in a predefined format. The methodology for calculating the number of CPU's is based on the expected maximum number of concurrent queries. The procedure is to calculate the maximum number of concurrent users times the maximum number of concurrent reports per user that the end-user business intelligence tool permits. The size of the memory is based on the size of tables, type of sorts, number and types of users, ETL tools transformations

For representing the dependency between data warehouse implementation variables, architectural overall decisions and designer decisions, I propose an inter-layers schema based on influences diagram. These diagrams where proposed by R. Howard and J. Matheson [13] for the purpose of qualitative, intuitive and especially compact representation of all the essential elements of a decision problem [14]. Influence diagrams are directed acyclic graphs that contain three types of nodes: rectangular decision nodes that represent the choices being made, oval chance nodes that represent the relevant random variables, and diamond-shaped value nodes that represent aspects of the decision maker's utility [15].

The inter-layers schema (data warehouse decisions schema) consists of:

a) Decision nodes represent the architectural decisions (database granularity, physical allocated memory and types of memory, degree of parallelism if there are multiple cores, database partitioning option, unique and non unique index usage – the count of output records is less than 15% of the input records) and also the software system parameters or "knobs" listed in Table 1. These parameters can be adjusted to improve the system's performance. There are two kinds of architectural decisions, implementation decisions consist of decisions regarding the physical implementation and placement of the database, ETL and reporting components and logical data model decisions (example: the trade-off between normalized and de normalized tables in terms of redundancy and data anomalies etc).

Table 1

Decision Nodes - Knobs

Parameters	Remarks
Large Pool Cache Buffer PGA	Cache parallel execution message buffers. Holds copies of data blocks read from data files Work areas allocated by memory-intensive operators: Sort-based operators (order by, group-by, rollup, window function), Hash-join
Multi block read count Block Size	Because the blocks are adjacent, I/O calls larger than a single block can be used to speed up the process Database blocks stored in the database files and cached in the SGA

b) Chance nodes (with deterministic nodes subtype), listed in Table 2, represent the relevant random variables or deterministic variables, can be extrapolated to some processes (reading process, sorting process) and are metrics associated with technical performance measurements - the continuing verification of the degree of anticipated and actual achievement of technical parameters [6].

Table 2

Chance nodes

Metrics	Remarks
UW	Number of units of work required to get data from a base table
MB	Message Buffers allow parallel query server processes to communicate with each other
AP	Access path: Table scan, a fast full index scan, or an index scan.
WAB	The amount of data (buffers) to be processed in the work area by memory-intensive operators
LRUB	Buffers processed based on LRU algorithm

c) Value nodes are associated with Measures of Effectiveness [6], operational measures of success that are closely related to the achievement of the mission or operational objective, in the intended operational environment.

The data warehouse decisions schemas should consider the following non-functional requirements metrics that are specific with BI Reporting Environment: Response Time, Static Volumetric, Dynamic Volumetric and Utilization. The variables listed in Tables 3, 4 and 5 represent only a part of the overall data warehouse variables.

Table 3

Response Time

Metrics	Remarks
Response time for simple Queries	Simple queries are assumed to make extensive use of indexes. They are quite CPU-intensive
Response time for medium Queries	Medium queries make some use of indexes
Response time for Complex Queries	Complex queries involve tables scans and make sequential accesses, they are very I/O intensive

Table 4

Static Volumetric

Metrics	Remarks
Number of active users	The total number of users that are connected to the data warehouse during day.
Total raw data	Raw data is not the same as database size as it excludes row overheads, free space, indexes, work areas etc
Active data	The data most users will only be interested in, for example aggregates or results tables.
Disk expansion factor	Convert raw data into disk space sizing, allowing for row overheads, free space, indexes, work areas etc.)
Contingency	Reserve capacity for other applications running alongside the Data Warehouse System
Data touched by avg. query for simple Queries	For simple or medium queries, the use of indexes will generally reduce disk I/O
Data touched by avg. query for medium Queries	
Data touched by avg. query for complex Queries	In the case of a full-table scan this will be the entire table
Query mix for simple query	Specifies the percentage of simple, medium and complex queries for the DW workload
Query mix for medium query	
Query mix for complex query	

Table 5

Dynamic Volumetric

Metrics	Remarks
Simple queries of an active user per day Medium queries of an active user per day Complex queries of an active user per day	Number of daily queries influences the performance of the system.

For the purposes of this workload characterization, simple queries are assumed to make extensive use of indexes, access data in single blocks and are not I/O intensive. Medium queries make some use of indexes, access data in multiple blocks and are somewhat I/O intensive. Complex queries involve full table scans, make use of complex joins, access data sequentially in multiple blocks and are very I/O intensive.

The mixture of queries to be run can greatly affect the amount of resources required. For example although the number of complex queries planned to be run may be small compared to the number of simple queries, complex queries are very resource intensive and may drive the need for a larger system.

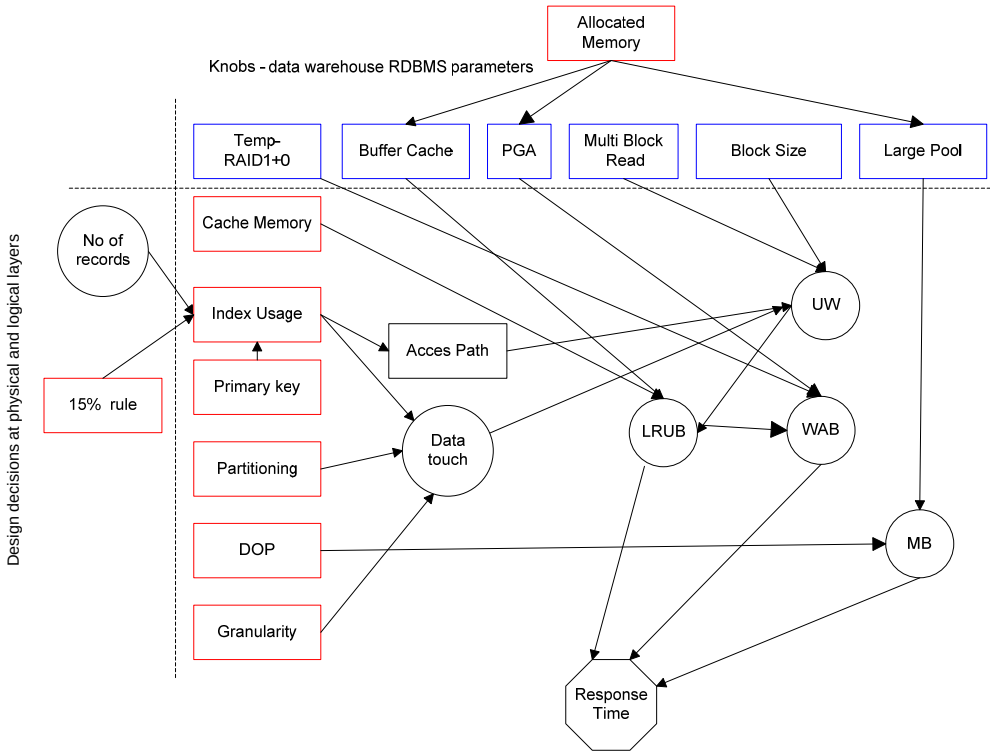

Fig. 4 – Data Warehouse Decisions Schema

The data warehouse decisions schema, is in fact a dependency diagram that considers the dependencies between architectural overall decisions, data warehouse database implementation decisions (Knobs), data model decisions and requirements. In the fig. 4, I considered only some of the important database parameters, overall decisions and designer decisions. In that case study the value node is referring to response time. In a complex environment, this diagram must be extended, all the reports must be categorized, non-functional requirements must be indicated very precisely and decisions nodes must be clear emphasized.

5. Conclusions

The ideas exposed so far have shown the complexity of a business intelligence/data warehouse system and emphasized that a proper approach must be done based on the systems engineering. This article considers the implementation of a flexible data warehouse framework using an inter-layers schema based on influences diagram and also emphasizes that the activities involved in the development of a business intelligence systems are not necessary included into a standard v-model development lifecycle. The inter-layers schema

(data warehouse decisions schema) will show the interdependencies between information requirements, overall architecture and performance and capacity of the system.

An important aspect is the logical data model which according to a classical approach is influenced only by the conceptual level and in turn influences the physical layer. This article allege the importance of considering the necessary relationships between all the systems inputs/outputs and emphasizes the bidirectional link between the design of the data model and capacity and performance of the system in order to obtain an optimal system architecture.

In fact an optimal business intelligence system must enforce user's information analysis with acceptable performance based on the integration and loading criteria's.

R E F E R E N C E S

[1] *R.T Clemen,* Making Hard Decisions. An Introduction to Decision Analysis. 2nd Edition. Duxbury Press, Belmont, 1996

[2] *F.G. Filip*, Sisteme Suport Pentru Decizii: O Incercare de Istorie, Revista Informatica Economica, **vol 29**, no.1, 2004

[3] *P. Keen, S. Morton*, Decision Support Systems: An Organizational Perspective, Addison-Wesley Publishing Company, Reading MA, 1978

[4] *M. Guran, A. Mehanna, B. Hussein*, Real Time On-Line Analytical Processing for Business Intelligence, U.P.B. Sci. Bull., Series C, **vol 7**, no. 3, 2009

[5] National Aeronautics and Space Administration*,* Systems Engineering Handbook, NASA SP-610S, 1995

[6] INCOSE*,* Systems Engineering Handbook, Incose Central Office, Seattle, 2000

[7] *P. Rob, C. Coronel*, Database Systems: Design, Implementation, and Management, Thomson-Course Technology, Boston, Massachusetts, 2007

[8] The Institute of Electrical and Electronics Engineers, IEEE Standard for Developing Software Life Cycle Processes, New York, 1998

[9] *S.S. Iliescu, I. Făgărăşan, D. Pupază*, Analiza de Sistem in Informatica Industriala (System Analysis in Informatics), Academia de Ştiinţe Tehnice din România, Seria Conducerea Proceselor Energetice, Editura AGIR, Bucureşti, 2006

[10] ANSI/X3/SPARC Study Group on Data Base Management Systems: (1975), Interim Report. FDT, ACM SIGMOD bulletin. **vol 7**, no. 2, 1975

[11] *P.P. Chen*, The Entity-Relationship Model-Toward a Unified View of Data, ACM Transactions on Database Systems, **vol 1**, no 1, 1976, pages 9-36

[12] *W. Inmon, C. Kelly*, The Twelve Rules of Data Warehouse for a Client/Server World, Data Management Review, 1994, pp 6-16

[13] *R. Howard, J. Matheson*, Principles and Applications of Decision Analysis, Strategic Decisions Group, Menlo Park, California, p.719-762, 1984

[14] *F.G. Filip*, Decizie asistata de calculator. Metode si tehnici de asistare a deciziilor centrate pe judecata umana, (Computer-assisted Decision. Methods and Techniques Focused on Human Mind), Revista Informatica Economica, **vol 15**, no.3, 2000,

[15] *D. Sullivan, M. Seltzer, A. Pfeffer*, Using probabilistic reasoning to automate software tuning, ACM SIGMETRICS Performance Evaluation Review, **vol 32**, no.1, 2004.