We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

154
Countries delivered to

Our authors are among the
TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™
Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Business Intelligence – CDMB – Implementing BI-CMDB to Lower Operation Cost Expenses and Satisfy Increasing User Expectations

Vlad Nicolicin-Georgescu[1,2], Vincent Bénatier[1],
Rémi Lehn[2] and Henri Briand[2]
[2]*Polytechnic School of Nantes University*
[1]*SP2 Solutions*
France

1. Introduction

Since the beginning of *Decision Support Systems* (DSS), benchmarks published by hardware and software vendors show that DSS technologies are more efficient by offering better performance for a stable cost and are aiming at lowering operations costs with resources mutualisation functionalities. Nevertheless, as a paradox, whereas data quality and company politics improves significantly, user dissatisfaction with poor performances increases constantly (Pendse, 2007).
In reality, this surprising result points the difficulties in globally defining the efficiency of a DSS. As a matter of fact, we see that whereas a "performance/cost" ratio is good for a technician, from the user perspective the "real performance/ expected performance" ratio is bad. Therefore, efficient may mean all or nothing, due to the characteristic of user perspective: a DSS may be very efficient for a developers needs, and completely inefficient for production objectives.
In order to improve the global DSS efficiency, companies can act upon (i) cost and/or (ii) expectations. They must be capable of well defining and exhaustively taking into account the costs and expectations relative to their DSS.
Concerning the first aspect of cost, the *Total Cost of Ownership* (TCO) provides a good assessment. It contains two principal aspects: hardware/software costs and cost of operations. With hardware/software, companies are limited by market offers and technological evolutions. The cost of operations, in addition to the potential gain by resource mutualisation, strongly influences the organization costs in rapport with its pertinence. Over this problematic, companies have implemented rationalization based on best practice guidelines, such as the *Information Technology Integration Library* (ITIL). Nevertheless, an internal study we have conducted at SP2 Solutions in 2007 showed that these guidelines are little or not-known to the DSS experts within or outside company IT departments. Our experience shows that this hasn't changed almost at all since then.
Relating with the second aspect, we state that user expectations cannot be ignored when evaluating the efficiency of a DSS. We estimate that the expected results must be based on the *Quality of Service* (QoS) and not on the raw technical performances. This way we define the DSS efficiency as a ratio between the DSS QoS and the DSS TCO, which is the foundation of our propositions.

Having described the context in which we position ourselves, we have developed and we offer several solutions to improve the DSS efficiency.
(1) First, we note that an improvement process cannot exist without a prior monitoring process. We propose to adapt and apply best practices, already deployed for operational systems, by implementing a system which permits the measuring of the DSS efficiency. Following ITIL guidelines, this kind of system is known as the *Configuration Management Data Base* (CMDB), and relates with ITIL specifications for: (i) service support and (ii) service offering (Dumont, 2007). The CMDB integrates all the information required to manage an information system. This information varies from incidents and known problems, configuration and change management, to management of availability, capacity, continuity and levels of service.
(2) Second, we show the necessity of taking into account DSS specifics (in contrast with operational systems), with the purpose of elaborating the right measures for the efficiency of a DSS. These measures are different from the ones used with operational systems, which focus on the problematic of real time utilization. DSS are often defined in comparison with operational information systems as the distinction between *On-Line Analytical Processing* (OLAP) and *On-Line Transaction Processing* (OLTP) shows it (Inmon, 2005), (Inmon, 2010). The two are part of fundamentally different worlds over aspects such as: the contained data (raw/aggregated), data organization (data bases/data warehouses), orientation (application/subject) or utilization (continuous/periods of utilization). DSS are classically aimed at managers and rely on analytical data, which is derived from the primitive data provided by operational systems.
Our major proposition presented here is the elaboration of the ***Business Intelligence CMDB (BI-CMDB)*** for the supervision of a DSS. It serves as a "decisional of the decisional" solution, and, to our knowledge and experience, it has never been approached so far with an integrated proposition. The BI-CMDB follows the principles of ITIL's CMDB and contains all the elements required for managing the DSS: architecture, configuration, performances, service levels, known problems, best practices etc. It integrates information both structured (e.g. configuration) and non-structured (e.g. best practices). With the help of the BI-CMDB, SLA/Os (Service Level Agreements / Objectives) are taken into account when computing performance, thus focusing on improving user satisfaction over technical performance.
Following this, a second contribution is to optimize the value offered by the CMDB. We combine BI, OLAP and semantic web technologies in order to improve DSS Management efficiency, while offering autonomic self management capabilities with references from (Nicolicin-Georgescu et al., 2009).
The chapter is organized as follows. Section 2 presents how monitoring of the DSS is done so far and what elements are followed in concordance with the ITIL guidelines. In Section 3 we focus on the description of the DSS specifics in comparison with the operational systems, and how performance measurement changes with these specifics. The subjective aspect of user perceived performance is equally discussed. Further, we present our proposition, the BI-CMDB, in Section 4. The A-CMDB and the integrated monitoring and analysis solutions for DSS management are some of the described elements. We equally note that the presented work has already been implemented and the solutions area available for use (SP2 Solutions, 2010). Section 5 consists of a series of possible shortcomings and limitations of the BI-CMDB, so we can conclude in Section 6 with an overview of the chapter and the future doors that BI-CMDB opens for DSS management.

2. Monitoring Decision Support Systems

As with any type of system, prior to any analysis and assessment, monitoring of the interesting elements is required. "You can't manage what you don't measure" is an old well known management adage, which seems to apply more than ever nowadays.

2.1 Monitoring challenges and guidelines

There are two challenges with monitoring, particularly underlined by the DSS: (i) getting some data and (ii) getting the right data.
The first aspect is obvious, and challenges here usually relate with technical barriers. For example, consider an interactive reporting application, which builds reports based on data from various data warehouses. If we are interested in the functioning of this application, we need to trace its running, usually from application and operation system logs (files, data bases etc.). The organization of these sources is usually very specific and requires specialized Extraction Transform Load (ETL) software. Therefore, the process is purely technical.
The second aspect of getting the interesting data proves to be much more subjective-oriented, as a natural question arises: Who needs this data? Continuing with our example, if the interesting party is a technician, he will most likely want to know if the available resources are enough for the application to work properly. If the interesting party is a business department responsible, he is more likely interested in knowing how often the application is used and if it offers any added value to the business process. Choosing the right data based on the specific needs is either done directly when monitoring is performed (i.e. gather only the interesting data), or is filtered afterwards from a pool of potentially interesting data.
Getting the right data to the right party is strongly related with the elaborated SLAs. In the context of information systems, an SLA is defined as a compact between a customer and a provider of an IT service that specifies the levels of availability, serviceability, performance, reporting, security or other attributes of a service, often established via negotiation from the two parts. It basically identifies the client's needs. IT organizations are measured on end-to-end business system response times, the degree of efficiency with which the system is used, and their ability to adapt to dynamic workloads (Ganek & Corbi, 2003). Breaking the SLAs may lead to financial and even legal problems, hence the need of their integration with the monitoring processes.
A very clear and detailed example of this aspect is shown by (Agrwala et al., 2006). The authors explain that Quality of Service is a particular concern for enterprise monitoring, exemplifying with an online airplane booking system. Formally, the QoS is defined as the level of current obtained data (usually performance) in rapport with what is expected. They describe QMON a QoS-capable monitoring system, which acts upon the system according to the importance of the monitored elements, importance expressed through the SLAs. The provided conclusion underlines the fact that QoS should be dynamically configurable and supported at monitoring level, allowing balance between monitoring overheads and improvement in application utility.
With DSSs, integration of SLAs and elaboration of QoS metrics is a central focus point. Unfortunately, best practices and guidelines to do so are not that elaborated as they are for operational system. One of the common mistakes done when managing a DSS is to use existing practices for operational systems and adopt them for DSSs without taking into

account their particularities (Inmon, 2005). Nevertheless, ITIL provides a good foundation with the CMDB over service sustainability and offering (Dumont, 2007).

Service sustainability corresponds to the interface between the service user and the system providing the service. The CMDB integrates monitored data from: incidents, known problems and errors, configuration elements, changing and new versions. Each of the five data types corresponds to a management module (e.g. incident management, configuration management etc.). In turn, each of the management modules is retaken by the user to ensure service maintenance.

Service offering formalizes the interface between the clients and the system. This time, it refers to offering new services in rapport with the defined policies. Starting from management and infrastructure software (like the CMDB), the service levels are assessed based on SLA/Os. Elaboration of these agreements and objectives include availability management, capacity management, financial service management, service continuity, management of the levels of service and the assurance of a high QoS.

2.2 Monitoring solutions with DSS software

The described ITIL CMDB is general for an IS. As DSSs lack proper monitoring, elaborating a CMDB for a DSS is the first step towards a more efficient DSS. At this point a discussion is required over how monitoring is achieved with various DSS software solutions. The three most common ways are: (i) logging files, (ii) logging databases and (iii) console specific commands.

Logging files are the most used source of monitoring information. They contain all messages regarding the functioning of the DSS. For example, IBM's Cognos 8 Financial Performance Management (IBM, 2011) solutions offer a list of various log files and tracking levels by purpose. These include: the transfer log file for recording activity from the installation wizard with the transferred files; the startup configuration file for logging configuration choices; the run-time log for tracking all events from the server while at run time. The information from the logging files is normally semi-structured, and in order to be useful, additional parsing and analysis must be performed.

Logging databases are preferred over log files as they provide a better information structure, but are equally harder to implement. A good example of this is the audit data base (the Audit_event table) from the Business Objects Enterprise solution (Business Objects, 2008). The table contains information about run time events, such as the event id, the name of the user, the timestamp etc. Logging of these events can be activated or deactivated by the user, as it slightly affects performance. This also comes from the fact that the audit table is constructed by accessing a list of services provided by the BO server.

Console specific commands are especially used when recovering data concerning configuration parameters, which are isolated within logs, rendering them harder to identify. Sometimes, they are even impossible to recover, if the respective log that contains the last parameter modification has been archived and is no longer part of the current logs. To exemplify, the Oracle Hyperion Essbase solution employs a specific command line utility console, the "essmsh", to retrieve information such as the file sizes of the data marts or the cache memory configuration parameters (Oracle, 2010). Even if this solution is the hardest to use, from the monitoring point of view, it proves useful in situations such as above.

Either of the data sources used, the monitored data is usually raw basic data. Nevertheless, a distinction is done with this data at a first level of analysis which results in a combined monitored data. Some of the solutions, such as the BO monitoring data bases, offer a list of combined data parameters. To provide with a very simple example, consider that a data mart with two size indicators: the index file size and the size of the data file. We can obtain the two separately, but we have no indicator of the total size of the data mart. This is obtained by adding the two and 'creating' a new combined monitored indicator.

3. Intelligent performance with DSS specifics

Having the monitored data, the next natural phase is to use it in order to measure the performance levels of the DSS, consequently allowing the assessment of its efficiency. There are several particularities that render the process more delicate than for operational systems.

3.1 DSS specifics

We make a presentation of the most notable DSS particularities, as taken from (Inmon, 2005).

The data warehouse is the heart of a DSS, and represents the architecture used for storing the analytical data. The core of data warehouses is the data mart, which is a repository of data gathered with the help of ETL software from operational data and other sources, with the purpose of serving a particular community of knowledge workers (SAP, 2010). Metaphorically speaking, data marts are the bricks that are used to construct the data warehouse building. Inmon identifies four main considerations for making the separation between the analytical and operational data: (i) analysis data is physically different from operational data, (ii) the technology used for operational systems is fundamentally different from the one used with DSS, (iii) the processing characteristics for the two are different and (iv) the analytical and operational communities are two different worlds.

DSS are *subject oriented*, with the goal towards business objectives, whereas operational systems are application specific focusing on application requirements. That is why measuring the performance of a DSS is harder as it is based on user expectations.

The *resource utilization patterns* of DSS are completely different from the ones of operational systems. There is a notion of time relaxation, as data warehouses are not used constantly with the same load. There are periods of maximum load, which are also known as critical periods and periods of small load or inactivity. For example, a budget planning application will only be used the first week at the beginning of each month, thus requiring the maximum of available resources during this period only. The other three weeks, it can function with the minimum needed resources. We identify two major criteria for the distinction of the utilization patterns: (i) *the utilization periods* and (iii) *the utilization purpose*.

The utilization periods reflect the critical periods and are usually specified with the SLAs. The performance objectives per utilization period and per user type, alongside the specification of the critical functioning periods, are regular elements of the agreement documents.

The utilization purpose formalizes the two main data cycles of the data warehouse: (i) the usage of the data for the generation of reports, and, (ii) the recalculation of the data for data warehouse update. They are also known as the day (production) and the night (batch) usage. During day, users access the various data from the data warehouses in order to build analysis reports, which help them with the process of decision making. The data warehouses are exposed to intense data retrieval operations. Even if the response times are very small (e.g. 80% of OLAP queries must be under a second (Codd et al., 1993)) the user activity is very high (even thousands of users). During night, batch operations are executed to integrate the new operational data (from the day that has just passed) into the data warehouse. The executed calculation and restructuration operations are very resource and time consuming. Moreover, the new data must be integrated before the start of the next day in order for the users to have the correct reports.

One last DSS characteristic we want to discuss here is the motto of the decision expert: *"Give me what I say I want, so I can tell you what I really want"* (Inmon et al., 1999). This expresses the fact that data warehouse users access a large variety of data from the data warehouse, always being in search of explanations over the observed reports. Therefore anticipation and prediction of the requested data, data usage or system charge is very hard to implement, leading to (not necessarily a good thing) more relaxed performance requirements.

3.2 Performance

Performance measurement with DSSs brings into discussion, along the technical aspect (from operational systems), the user satisfaction aspect (form the business objective oriented point of view of analytical systems). There are two main types of performance (Agarwala et al., 2006):

Raw technical performance – related to the technical indicators for measuring system performance (such as response times for recovering data from the data warehouse, report generation, data calculation etc.)

Objective performances – relating to user expectations. These measures are based on predefined policies and rules that indicate weather or not satisfactory levels are achieved by the system (e.g. response time relative to the importance of the data warehouse or with the user activity).

Retaking the example shown by the authors of (Agarwala et al., 2006), a detailing of the QoS formula is shown below.

$$QoS = \frac{MonitoredValue}{ObjectiveValue} \tag{1}$$

and

$$QoS_{global} = \frac{\sum_{1}^{n} QoS_i * ScalingFactor_i}{\sum_{1}^{n} ScalingFactor_i} \tag{2}$$

The basic single parameter QoS is computed by the ratio between the monitored values and the specified expected values. If several parameters are responsible for a global QoS, then a scaling factor is multiplied with each individual QoS, and, in the end, an average of all the scaled QoS is computed to obtain the overall system QoS value.

The data warehouse quality model is goal oriented and roots from the goal-question-metric model. Whether metrics satisfy or not the goal determinates the quality of the data warehouse (Vassiliadis et al., 1999). Data warehouses must provide high QoS, translated into features such as coherency, accessibility and performance, by building a quality meta-model and establishing goal metrics through quality questions. The main advantage of a quality driven model is the increase in the service levels offered to the user and implicitly increase in the user satisfaction. QoS specifications are usually described by the SLAs and the SLOs.

We have seen what SLAs stand for. In a similar manner, SLOs describe service objectives, but on a more specific levels. For example, a SLA would describe that an application is critical on the first week of each month, whereas the SLO would state that during critical periods the query response time on the data marts used by the application should not be greater than 1s.
With DSSs and data warehouses, one important area where service levels are crucial is the utilization periods, in direct link with the answer to the question: *what is used, by whom and when*? Data warehouse high charge periods are specified with SLAs. When building a new DSS and putting in place the data warehouse architecture, the utilization periods are specified for the various applications. To paraphrase a study from Oracle, *the key to enterprise performance management is to identify an organization's value drivers, focus on them, and align the organization to drive results* (Oracle, 2008).

4. The BI-CMDB

In this section we present, in an exhaustively manner, our view and proposition for building a CMDB adapted to decisional systems, which we consequently call the BI-CMDB. In order to understand how this is built, we present a schema of the data flow and the organization of five modules that are part of our approach.

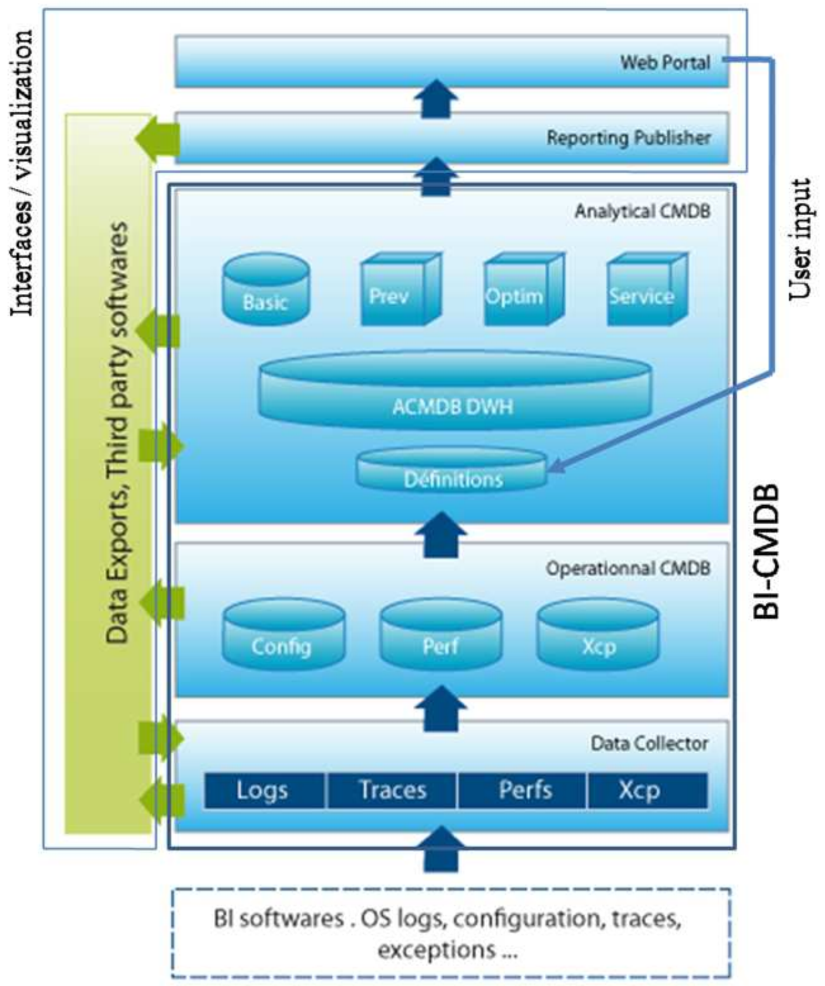


Fig. 1. The SP2 BI-CMDB simplified schema

On the bottom level we have the *data sources* that provide the monitored data. As mentioned, these are distributed, non-centralized and under different formats. Next, there is a *data collector* module, which gathers and archives the data sources. This is the first level of centralization while greatly reducing the size of this data by archiving. Following, an *Operational CMDB* is build with the data collector centralized information. It is the first level of filtering (parsing and interpretation) of the monitored data, which allows to keep only the interesting data for further analysis.

Analysis of the O-CMDB is brought climbing up to the *Analytical CMDB*, which builds a data warehouse architecture, by specific objective data marts, with the data from the O-CMDB. We note that the elements of the O-CMDB and of the A-CMDB correspond to ITIL's guidelines, with the addition of elements that correspond to the specifics of DSS. Once the A-CMDB has been constructed, two more levels of information access (by user) are added, with a module for *publishing the reports* built on the A-CMDB data warehouse and a *Web Portal* for consulting these reports and for dynamically exploring the data from the CMDBs. These last two are not of interest for this chapter, but equally make a challenge, from the study of computer-human interaction and GUI design.

A detailed description of the first three modules that are interesting to the BI-CMDB is done in the following. Examples for each of them are provided, in the light of the differences with the traditional CMDB and on the characteristics of DSS.

4.1 The data collector

The data collector is the first module in the data flow towards the construction of the BI CMDB. Its main objective is to gather and archive the various interesting monitoring information.

The core of the data collector architecture is the *connector*. A connector represents a generic interface for defining the elements that allow the gathering of information from a data source and the methods used for this process. The parameters of a connector allow him to connect to a target machine (i.e. name of the machine, connection port, data source location and type etc.), recover and archive the data from the sources and send these archives via ftp connections to the machine which is in charge of archive analysis and construction of the O-CMDB. Depending of the target machine installed BI software solution, various implementations of the generic connector are required (e.g. an Oracle Hyperion Essabse connector or a BO connector).

As we have seen previously in the monitoring section, there are three types of data sources: logging files, logging databases and specific console commands. A connector can specify any (or all) of these types.

Following the ITIL guidelines, a separation of the *data fluxes* is done for each connector (where possible). There are five types of data fluxes: (i) logging (*log*), (ii) exception (*xcp*), (iii) parameters (*param*), (iv) performance (*perf*) and (v) errors (*err*). This separation persists further on with the construction of both the O-CDMB and the A-CMDB. Another argument for this split is the fact that some solutions use different data sources for different fluxes. For instance, with Essbase, the monitored information is usually stored in plain text files (.log and .xcp). These files allow the identification of all the required data with the exception of the configuration parameters. Due to the fact that they change rarely, their identification through log files is very difficult or even impossible. This is why for this task, specific console commands are used for the *param* flux.

Example. We show below an example of data collection for a connector implementation for the Oracle Hyperion Essbase server, with the various connector data fluxes.
The Essbase server log files are under the */log* repository of the Essbase installation. The connector will specify the machine and port (for connection to the Essbase server) and the place of the Essbase installation (e.g. *C:/Hyperion/logs/essbase*). This repository contains a suite of *.log* and *.xcp* files, that are organized by application (here the term application indicates the regrouping several data marts which suit the same business objective)
The *.log* file contains various information about the application and its data marts, which is interesting for the *log*, *perf* and *err* data fluxes. An extract of the file is shown below for a data retrieval operation on a data mart from the application:

```
[Mon Jan 31 17:57:51 2011]Local/budgetApplication///Info(1013210)
User [admin] set active on database [budDM]
[Mon Jan 31 17:57:51 2011]Local/budgetApplication/budDM/admin/Info(1013164)
Received Command [Report] from user [admin]
[Mon Jan 31 17:58:21 2011]Local/budgetApplication/budDM/admin/Info(1001065)
Regular Extractor Elapsed Time : [29.64] seconds
```

These lines permit to identify that there was an Essbase report request (data retrieval) on the budDM data mart, part of the budget application on the on 31st of January at 17:57:51 from the admin user, which lasted 29.64 seconds. These elements represent the interesting monitored information, and are recovered from the last four lines. The first two lines are useless. This filtering process is done at the construction of the O-CMDB.
The *.xcp* file contains the information needed for the *xcp* data flux, logging the fatal crashes of the Essbase server. An extract of this file could be:

```
Current Date & Time:   Fri Sep 19 11:29:01 2008
Application Name:      budgetApplication
Exception Log File: /app/hyp/Hyperion/AnalyticServices/app/budgetApplication/log00001.xcp
Current Thread Id:     2
Signal Number:         0x15=Termination By Kill
```

Last, in order to get the configuration parameters of the application data marts, the esscmd command line console is used (i.e. GETDBINFO, GETAPPINFO etc.). The results of these commands are written into customized text files, which assure the *perf* data flux.
The role of the example was to show that the monitored information is very different, and data sources and formats vary even within the same implementation. The data collector has the role of assuring a first centralized level of information, with no data analysis intelligence. The only intelligence in the module allows an incremental data collection, such that data is not redundant (e.g. if data is monitored once each day, each collection takes only the information from the logs corresponding to the last past day). The following levels of information centralization are shown next with the construction of the O-CMDB and the A-CMDB.

4.2 The O-CMDB

The Operational CMDB is the first intelligent unification of the DSS information. It suits the role of an Operational Data Store (ODS), and it is build from the data collected by the data collector, by a series of filtering and analysis operations. The O-CMDB contains only the interesting information, rejecting any other unrequited data from the collector archives (as exemplified earlier with the Essbase report generation).
At first glance, the O-CMDB may seem as a simple log analyzer. This cannot be further from the truth. In reality, the process of building the O-CMDB makes a sweep of the information

available, holding only the useful one (through interpretation of the log data), and constructing a series of purpose objective data stores with this filtered data. The organization of the data stores retakes the main ITIL groups for the CMDB and continues in the line of the data collector connector data fluxes. Technically, our implementation of these data stores is via relational databases, for both the O and the A CMDB, term which we will further use for a better understanding.
The following O-CMDB databases are constructed.
The *CI (Configuration Items)* tables, holding information about the architectural organization of the target decision system. These tables are different by construction, as they are not built with the data recovered from the data collector, but they have as source specific defined files or databases which we use with our system to let the user 'complete' his DSS architecture. In specific cases some of the CIs can be automatically built from the data from the logs. This operation of DSS entity specification must be performed before the construction of the O-CMDB and is compulsory, as the rest of the tables are based on the DSS entities.
In our conception, we identify six levels of vertical hierarchy: (i) the entire DSS, (ii) the physical servers, (iii) the logical servers, (iv) the applications, (v) the bases and (vi) the programs. In order to keep this as generic as possible we note that depending on the specific BI software implementation, some of these levels disappear or are remapped (e.g. an application for Essbase is well defined with its own parameters while for a BO solution it becomes a report folder as a logical organization of BO reports).
The *PARAM* tables contain information about the configuration indicators. This is raw data, organized by indicator type and by CI type. A reference measure table (configuration and performance), lists the interesting monitored indicators. The data collector archives are analyzed for patterns that allow the identification of these indicators, and then extracted and written in the O-CMDB *PARAM* table. An example is shown in Table 1. We have mentioned that the data contained here is raw data, meaning that there is no treatment done on it. This is what we have earlier seen as basic monitored structured information. However an exception from this rule is done with combined monitored configuration indicators (i.e. deduced from existing indicators). The given example was on the size of an Essbase data mart. There are two basic configuration indicators (data file size and index size) which are extracted directly from the data collector archives and stored in the O-CMDB. Yet, an interesting measure is the total size of the data mart, which expresses as the sum of the two indicators. In this case, when constructing the O-CMDB a second passage is made to compute these combined indicators.

Data collector archive	***O-CMDB CONFIG table extract***		
	Field name	***Field Type***	***Data value***
[Tue May 11 17:57:51 2010] Local/budgetApplication/budDM Index Cache Size : [100] Mo	***CI_CODE***	INT	2
	MEASURE_CODE	INT	10
	MEASURE_VALUE	VARCHAR	100
	MEASURE_TYPE	VARCHAR	Mo
	DATE_CHANGE	DATE	11/05/2010

Table 1. O-CMDG PARAM table extract

The *PERF* tables are similar to the *PARAM* tables with the difference that the contained indicators express performance measures. The distinction between the two is made for the

further analytical purposes, and in concordance with ITIL guidelines. Still, a series of conceptual differences are identified, mainly related to the timeline intervals. Configuration indicators are recorded in relation with the date at which they last change, operations that are rare. On the other hand, performance indicators have a high apparition frequency, and depend on several factors (from the network lag to the machine processor charge). On busy periods, we have met up to several of tens of thousands performance entries in the log files, which is a totally different scale from the configuration case.
In addition to these three 'main' tables, a series of additional tables are build, such as the *XCP* tables with data concerning fatal crashes, the *ERR* tables with data from (non-lethal) warnings and errors or the *MSG* tables with the various functioning messages.
One of the biggest benefits of constructing the O-CMDB is the reduction of the size of the monitored information. By comparing the size of the initial raw monitored data sources with the size of the O-CMDB database after integration of the data from those sources, we have obtained a gain of 1 to 99 (i.e. for each 99Mb of raw data we have an equivalent of 1Mb of integrated interesting data). In addition, this reduction allows keeping track of the data for both short and medium time periods (even long in some cases), which permits further prediction and tendency assessment.

4.3 The A-CDMB

The Analytical CMDB is the third and most advanced level of data integration. Unlike the O-CMDB which contained only raw data, the information in the A-CMDB is derived from this data. It represents the data warehouse of the O-CMDB, with analytical data suitable to business objectives. Considering the fact that the data from the O-CMDB regards a DSS implementation, we call the A-CMDB as the data warehouse of the DSS, or the support for 'the decisional of the decisional'.
The A-CMDB consists of tables (i.e. *AGG_PERF, AGG_PARAM*) which aggregate the O-CMDB tables on several axes of analysis. These axes are the fundamental characteristic of the A-CMDB, and reflect the characteristics of a DSS while suiting the DSS user needs. The main analysis axes are: (i) temporal intervals, (ii) utilization periods, (iii) utilization purpose, (iv) user type, (v) SLA/SLO and, obviously, (vi) the CIs. Data is aggregated on each of these axes, and their intersection provides the DSS user with any valuable information that he requires. We detail each of these axes, as they represent key points of our proposition.
The *CIs* represent the first implicit analysis axis, as it provides the vertical aggregation on the DSS hierarchy. Aggregation operations can either be averages, sums, standard deviations or min/max (e.g. the total size occupied by the entire DSS, the sum of all sizes of its data warehouses and the correspondent data marts, the maximum response time etc.).
Temporal intervals are the correspondent to the fundamental characteristic of the data warehouse, the time aggregation of the operational data. In the same manner, when building the O-CMDB we make a separation of six time intervals: year, month, weak, day, hour, min:sec. Buy using these six pivots, the A-CMDB aggregates the averages over each of them (e.g. the average response time for each week for a specific CI). The CI axis is the base axis which is always found in the aggregation for any of the rest of the axes.
Utilization periods are integrated in the A-CMDB in concordance with the specified data warehouse functioning patterns. By using the temporal intervals we are able to deduce if a specific CI is in its utilization period. The integration of the utilization periods in the *AGG_* tables will further be used for the assessment of the level of user satisfaction and the computation of the Quality of Service, in rapport with the given performance objectives.

Usage purposes are similar to the utilization periods, and are formalized under the same form. By using the hour time axis, the day and night intervals are defined (i.e. from 6:00 to 22:00 there is day, from 22:01 to 05:59 is night). Integrating this axis allows the user to identify or not whether a certain performance analysis is pertinent (i.e. following the restitution times during night periods makes little sense).
The *user type* axis represents the main focus axis when assessing user satisfaction. A response time that is acceptable for the developer who is testing the data warehouse can be completely unacceptable for a DSS user who wants his rapport as soon as possible. In the O-CDMB, the tables have columns which specify the user who triggered the actions (e.g. the user who asked for a report, the user that modified the configuration parameters etc.). When integrated in the A-CMDB, users are organized logically by areas of interest, such as: (i) data warehouse administrator, developer or user; (ii) area of responsibility (entire system, department, application); (iii) type of responsibility (technical, business objective). By intersecting these axes, the A-CMDB can rapidly respond at any questions concerning the user activity, thus covering a total 'surveillance' of *who is using what and when*.
Lastly, the *SLA/SLO* axis integrates all the elements related to business / technical objectives. For instance, defining the time intervals for the utilization and usage periods as business objectives and specifying the performance thresholds for each of these periods as technical objectives. By having these values we can afterwards compute the QoS indicators, with regards to any intersection of the analysis axes, as the A-CMDB offers all the data required to do so. The SLA and SLO data are not obtained from the O-CMDB and are not part of the data recovered by the data collector. They are usually integrated when building the A-CMDB, either by being manually specified or by using ETLs to load them from the source where they are specified. Even it may seem a paradox, in many enterprises today SLAs and SLOs are specified in semi-formatted documents (e.g. word documents) and their integration is only possible manually. This represents a big drawback for automatic integration, and by respecting the A-CMDB specifications, enterprises are 'forced' to render these elements into formats that allow automatic manipulation (such as spreadsheets, .xml or even data bases). Even if this alignment operation poses an effort in start, the return over investment makes its implementation worth.
We provide an example with an extract of some of the columns from the AGG_PERF table, which aggregates the O-CMDB performance tables.

Field name (Field type)	***CI_CODE (INT)***	***YEAR (INT)***	***WEEK (INT)***	***MEAS_CODE (INT)***
Data value	5	2010	23	15
Field name (Field type)	***USER (VARCHAR)***	***USAGE (VARCHAR)***	***SLO_OBJECTIVE (FLOAT) (user specified)***	***QOS (FLOAT)***
Data value	admin	Day	3	0.56
Field name (Field type)	***AGG_VALUE (FLOAT)***	***AGG_TYPE (VARCHAR)***	***UTILIZATION (VARCHAR)***	
Data value	5.3	avg	high	

Table 2. AGG_PERF tables extract

The table shows the fact that for the CI with code 5 (e.g. is an Essbase logical server), during the 23rd week of 2010, the measure response time (code = 15), has an average of 5.3 seconds.

Moreover, this average is computed for the admin user, during a high utilization periods and during day usage. Finally, there is an SLO objective of 3 seconds for the admin user under the same conditions, which leads to a quality of service of 0.56 (=3 / 5.3), meaning that the admin user is only 56% satisfied with the logical server response times. If we consider that under a level of satisfaction of 80%, more details are required, automatically we have the need to drill into the data in order to know exactly what are the applications and further the bases and the programs that generate this dissatisfaction. This functioning is characteristic for any data warehouse (from the DSS user motto).
Therefore, the A-CMDB contains all the information needed for the analysis of the functioning of the DSS. It is a question of vision and development of the ITIL's CMDB as a decisional component. Moreover, it is about applying the specific DSS characteristics for its construction, such that the relevant measures and indicators are well defined and that DSS supervisors have a very clear view over the performance of their system from the perspective of user and their satisfaction levels when using the DSS.

4.4 Advanced practices for the evolution of the BI-CMDB

Having seen how the BI-CMDB is built by processes from data collection, to construction of the operational CMDB and finally of the analytical CMDB, a series of advanced practices and topics are presented next, most of which open doors to future evolution of the model.
The benefice of the BI-CMDB is that it allows a complete (on every level) view of the functioning of an enterprise DSS.
One first very good application of this view is prediction and prevention. Using prediction algorithms and analyzing the data from the A-CMDB, we can see the potential bad tendencies of a DSS. For instance a slow but constant higher memory usage can lead to a system crash when the RAM memory is no longer sufficient. Moreover, by looking at past evolutions, prevention can be enabled by avoiding the previously done 'mistakes'. For example, modifying the available cache values of a data mart which leads to disastrous results can be avoided from repeating itself.
The aspects of prediction and prevention lead to a second application case, which is *change impact analysis*. What happens to my DSS if I change this or that parameter, if I buy additional resources, if I change the version of my BI software solution etc.? are commonly asked questions. So far, answers to these questions were practically impossible to provide specifically. With the help of the BI-CMDB detail levels, impact analysis can be rapidly performed by comparing the various situations and intersecting the analysis axes (e.g. the average response times per logical servers with different data mart cache values and different RAM memory available on the server). By intersecting the various configuration axes of the A-CMDB with the performance axis, one can isolate and explain performance variations. Moreover, by building a knowledge base around the BI-CMDB, the question of change management and migration can be responded with the integration of all the external 'meta' knowledge regarding the DSS management.
This is where the third advanced topic is discussed: providing meta-data and semantics with the BI-CMDB. For example, integrating all the information from the release Readme documents of the software, in order to identify the editor specified known and solved bugs or software compatibilities. When performing a migration such information is vital, and having it integrated with the BI-CMDB under a unified machine understandable form would simplify a lot the processes. In order to achieve this, solutions that imply the usage of web semantic technologies, specifically ontologies (Gruber, 1992), have been recently proposed for exploration (Nicolicin-Georgescu et al., 2010), and have been the result of our

work over the last years. Using ontologies to formalize the DSS management aspects (such as best practices, subjective quality measurement, business rules etc.) go hand in hand with the BI-CMDB elaboration. Moreover, by using ontologies, it is possible to specify horizontal relations between the CI, as an addition to the hierarchical vertical relations of the DSS architecture, thus allowing a full description of the elements and their connections.
Last, but not least, having a complete integrated view of the data and information describing the DSS, automatic behaviors can be adopted, such that the DSS human expert is helped with managing his system. To this end, solutions as the Autonomic Computing (IBM, 2006) have been developed, with the declared purpose of helping IT professionals with low level, repetitive tasks, so they can focus on high level, business objectives. Inspired by the functioning of the human body, this autonomy model proposes four principles to assure that a system can function by 'itself', self: configuration, healing, optimization and protection. An intelligent control loop implemented by several Autonomic Computing Managers, assure these functions. It has four phases, which correspond to a self sufficient process: monitor, analyze, plan and execute, all around a central knowledge base.
The problem with autonomic computing is that it has been elaborated with the real time operational systems as targets. In order to apply the autonomic model for DSS, similar to the passage from the CMDB to the BI-CMDB, the characteristics of DSS must be taken into consideration. The works of (Nicolicin-Georgescu et al., 2009) have shown an example of optimizing shared resources allocations with data warehouses, by focusing on the data warehouse importance and priority to the user rather then on raw technical indicators. Moreover, to comply with the DSS functioning, changes in the traditional loop functioning were proposed, via the description of heuristics targeted at the SLAs and the improvement of the QoS.

5. Possible limitations and shortcomings

Before concluding this chapter, an overview of the limitations and shortcomings of the BI-CMDB is done. We note that with some of them we have already been confronted, whereas others are presented only as possible to arrive.
The first limitation when building the BI-CMDB is *generics*, specifically the difference between CI modeling of the different BI software solutions. As presented, the CI tables in the O-CMDB consist of six hierarchy levels, which were thought to fit a maximum number of software implementations. Yet, with every one of them, the connotations of a specific CI table change (e.g. a base table in the case of OLAP software becomes a report table in the case of reporting software). Moreover, the elements composing the DSS are not always logically organized hierarchically (not on all levels). For instance the reports and data fluxes with BO, are related through horizontal links. So vertically aggregating on the CI hierarchic sometimes doesn't make sense. Nevertheless, what is important is that the model offers the specification capacity, thus finally generics is assured through small (of semantic nature) changes.
Another limitation of the solution is related to the construction process time of the O and A CMDBs. One may object the fact that due to the times needed to build these databases, real time analysis cannot be achieved. Yet, is there a real need for real time? The answer may be yes in the case of technical alerts (which simplifies a lot the processes), but no in the case of DSS analysis. In this case, the 'standard' time interval is an operational day, following the DSS usage purpose characteristic (i.e. the day/night usage cycles). At the end of each operational day, the data collector recovers the new information via the configured connectors, archives it and passes it for integration into the O-CMDB. This interval evidently

is configurable, based on the specific needs (e.g. once each week, once each 3 days etc.), and user needs for this interval are mostly higher than the time needed to integrate the data into the BI-CMDB.
Following, one shortcoming may be the unpredictability of usage of the system, which makes scaling effects harder to measure. We have here a decisional of the decisional solution, which metaphorically is like doubling the DSS expert motto. A user of the BI-CMDB will constantly require new rapports and find new analysis axes to intersect. This means, that alongside the list of the main analysis axis that are implemented, depending on the users needs and objectives a series of new analysis pivots are described. As these needs vary from case to case, it is hard to asses the usage process of the BI-CMDB. Its performances depend strongly on the chosen client implementation. Moreover, through the web portal interface the user has the liberty to define his custom queries towards the BI-CMDB, a liberty which can make the system overcharge (similar to the data warehouse user freedom problematic).
Also, as we have spoken of the combination of the BI-CMDB with ontologies, this is regarded as a future evolution. For now, assuring model consistency and completeness (from the contained data point of view) is done 'semi-manually', after the report generation (therefore at the final stage). We try to move this verification at database construction level, as to avoid any unnecessary report generation (or worst give the bad report to the bad receiver). Implementing an ontology model and by benefitting from the inference engines capacities, such verifications can be done at the construction stage, before any utilization of the BI-CMDB data. A classic example is the mixing of the CIs from the hierarchy. The CI specification is build from the manual user specification of his system, thus is prone to human error. A physical server which becomes an application by accident would lead to completely incorrect rapports (from what the user was expecting).
Finally, we note that semantic and autonomic evolutions of the BI-CMDB arise a series of questions, from knowledge integration to supervision of autonomic behaviors and rule consistency.

6. Conclusion

We have detailed in this chapter an implementation of the Business Intelligence – CMDB, with the purpose of improving the efficiency of a DSS. Based on the fact that despite the technological advancements, user satisfaction with DSS performance keeps decreasing, we have isolated some of the elements that are responsible for this paradox, among which the Quality of Service, the Total Cost of Ownership and the integration of Service Level Agreements policies when computing DSS performance. We state that user perception is a vital factor that cannot be ignored in the detriment of the raw technical performance indicators.
In conclusion, implementing the BI-CMDB for DSS offers a series of elements we consider compulsory to an efficient DSSs: a CMDB for DSS, a complete TCO, an integration of the SLA/Os, an adoption of ITIL guidelines and best practices, all these with lower costs and the improvement of the user's satisfaction levels. The BI-CMDB proposition for the 'decisional of the decisional', integrated with cutting edge semantic and autonomic technologies opens the doors to a whole new area of research towards the future of efficient DSS.
Nevertheless, developing the BI-CMDB is a first successful step towards the future of DSS. With the development of semantics and autonomic technologies, we strongly believe that future steps include these aspects, towards the development of a complete BI-CMDB,

integrated with all pieces of knowledge required for managing a DSS. The ultimate *autonomic semantic based BI-CMDB* for DSS is a planet which seems "light years" away, but currently technology evolution speeds rapidly approach it.

7. Acknowledgment

We would like to thank Mr. Eng. Adrien Letournel and Mr. Eng. Guillaume St-Criq, both employees of SP2 Solutions, for their support with providing the data and the elements for the examples and for their feedback and support with this chapter.

8. References

Agarwala, S., Chen, Y., Milojicic, D. & Schwan, K. (2006), Qmon: Qos-and utility-aware monitoring in enterprise systems, *in* 'IEEE International Conference on Autonomic Computing', pp. 124–133.

Codd, E. F., Codd, S. & Salley, C. (1993), 'Providing olap to user-analysts: An it mandate'.

Dumont, C. (2007), *ITIL pour un service informatique optimal 2e édition*, Eyrolles.

Ganek, A. G. & Corbi, T. A. (2003), 'The dawning of the autonomic computing era', *IBM Systems Journal* 43(1), 5–18.

Gruber, T. (1992), *What is an ontology?*, Academic Press Pub.

IBM (2006), *An architectural blueprint for autonomic computing, 4th Edition*, IBM Corporation.

(2011), 'Ibm cognos 8 financial performance management'. Last accessed - January 2011. http://publib.boulder.ibm.com/infocenter/c8fpm/v8r4m0/index.jsp?topic=/com.ibm.swg.im.cognos.ug_cra.8.4.0.doc/ug_cra_id30642LogFiles.html

Inmon, W.H. (2010), 'Data warehousing: Kimball vs. inmon (by inmon)'. last accessed july 2010. http://www.b-eye-network.com/view/14115/

Inmon, W. H. (2005), *Building the data warehouse, fourth edition*, Wiley Publishing.

Inmon, W. H., Rudin, K., Buss, C. K. & Sousa, R. (1999), *Data Warehouse Performance*, John WIley & Sons, Inc.

Nicolicin-Georgescu, V., Benatier, V., Lehn, R. & Briand, H. (2009), An ontology-based autonomic system for improving data warehouse performances, *in* 'Knowledge-Based and Intelligent Information and Engineering Systems, 13th International Conference, KES2009', pp. 261–268.

Nicolicin-Georgescu, V., Benatier, V., Lehn, R. & Briand, H. (2010), Ontology-based autonomic computing for resource sharing between data warehouses in decision support systems, *in* '12th International Conference on Enterprise Information Systems, ICEIS 2010', pp. 199–206.

Objects, B. (2008), Business Objects enterprise xi white paper, Technical report, SAP Company.

Oracle (2008), Management excellence: The metrics reloaded, Technical report, Oracle.

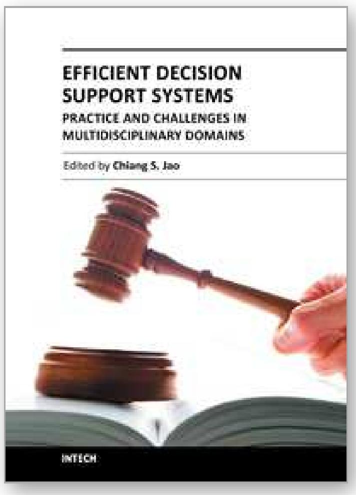
Oracle (2010), 'Oracle Hyperion Essbase'. last accessed July 2010. http://www.oracle.com/technology/products/bi/essbase/index.html

Pendse, N. (2007), 'Performance matters - the OLAP surveys hold some performance surprises'. www.olapreport.com

SAP (2010), Data, data everywhere: A special report on managing information, Technical report, SAP America, Inc.

SP2Solutions (2010), 'SP2 Solutions homepage'. last accessed December 2010, www.sp2.fr

Vassiliadis, P., Bouzeghoub, M. & Quix, C. (1999), Towards quality-oriented data warehouse usage and evolution, *in* 'Proceedings of the 11th International Conference on Advanced Information Systems Engineering, CAISE 99', pp. 164–179.


EFFICIENT DECISION SUPPORT SYSTEMS
PRACTICE AND CHALLENGES IN MULTIDISCIPLINARY DOMAINS
Edited by Chiang S. Jao
INTECH

Efficient Decision Support Systems - Practice and Challenges in Multidisciplinary Domains
Edited by Prof. Chiang Jao

ISBN 978-953-307-441-2
Hard cover, 478 pages
Publisher InTech
Published online 06, September, 2011
Published in print edition September, 2011

This series is directed to diverse managerial professionals who are leading the transformation of individual domains by using expert information and domain knowledge to drive decision support systems (DSSs). The series offers a broad range of subjects addressed in specific areas such as health care, business management, banking, agriculture, environmental improvement, natural resource and spatial management, aviation administration, and hybrid applications of information technology aimed to interdisciplinary issues. This book series is composed of three volumes: Volume 1 consists of general concepts and methodology of DSSs; Volume 2 consists of applications of DSSs in the biomedical domain; Volume 3 consists of hybrid applications of DSSs in multidisciplinary domains. The book is shaped decision support strategies in the new infrastructure that assists the readers in full use of the creative technology to manipulate input data and to transform information into useful decisions for decision makers.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Vlad Nicolicin-Georgescu, Vincent Bénatier, Rémi Lehn and Henri Briand (2011). Business Intelligence – CDMB – Implementing BI-CMDB to Lower Operation Cost Expenses and Satisfy Increasing User Expectations, Efficient Decision Support Systems - Practice and Challenges in Multidisciplinary Domains, Prof. Chiang Jao (Ed.), ISBN: 978-953-307-441-2, InTech, Available from: http://www.intechopen.com/books/efficient-decision-support-systems-practice-and-challenges-in-multidisciplinary-domains/business-intelligence-cdmb-implementing-bi-cmdb-to-lower-operation-cost-expenses-and-satisfy-increas

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2011 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.