ELECTRONIC **WORKSHOPS IN COMPUTING**

Series edited by Professor C.J. van Rijsbergen

Rainer Manthey and Viacheslav Wolfengagen (Eds)

Advances in Databases and Information Systems 1997

Proceedings of the First East-European Symposium on Advances in Databases and Information Systems, (ADBIS'97), St Petersburg, 2-5 September 1997

Comparison of the Decision Support Systems and the Transaction Support System Development Methodologies

Vinko Cippico

Published in collaboration with the
British Computer Society

ISBN: 3-540-76227-2

Comparison of the Decision Support systems and the Transaction Support System Development Methodologies

Vinko Cippico,
Croatian Radio Television
Prisavlje 3, 10000 Zagreb, Croatia
Tel.: + 385 1 616 3796
Fax: + 385 1 611 2425
E-mail: Vinko.Cippico@.hrt.hr

Abstract

There is a substantial difference between the development methodologies of the following two Information Systems: Transaction Support System (TSS) and Decision Support System (DSS). The central concern of the Transaction Support System that supports the operational world in organizations is transaction processing. The retrieval of information for analytical purposes is in the very focus of the Decision Support System. The DSS provoked a series of changes in RDBMS technology. In 1995 the data warehousing was recognized as a new concept. Multi-dimensional architecture of a data model emerged beside the existing relational one. The following year brought an explosion of the OLAP tools. Now the time has come for the full recognition of the DSS development methodology as a separate discipline. This methodology is presented on the following pages. There are obstacles that make it difficult to accept or have a clear view of it. These obstacles are highlighted through the comparison to the TSS development methodology that represents the traditional point of view. Tasks, deliverables, techniques, approaches, differences, similarities and analogies are discussed for each development activity. It is irrelevant for the discussion whether the DSS comprises data warehouse or/and some other store of data of the similar type, e.g. data mart or/and operational data store (ODS). This paper tries to reveal why it is so hard to accept the Decision Support System development methodology.

Keywords: development methodology, DSS, TSS, data modeling, data warehouse, OLAP

1 Introduction

In general there are two categories of tasks in all organizations. The first one deals with daily operational tasks. The goal of the second one is making business decisions. The idea that the traditional computer based Information System i.e. the Transaction Support System **cannot support both categories** of tasks, has not been widely accepted until recently.

The Transaction Support System (TSS) is an Information System (IS) that supports the operational world. Transaction processing is in the very focus of the TSS. In the focus of the **Decision Support System (DSS)** there is the retrieval of information for analytical purposes. This two Information Systems represent two different - even opposed - views of the same data. The distinction between them is as profound as between the mentioned two categories of tasks.

Although the TSS is hardly able to satisfy the on-line analytical processing (OLAP) tasks for different reasons that are going to be clarified afterwards, the **resistance** exists to the implementation of the DSS by both groups involved: the users and the computer professionals. Paradoxically, it seems that computer professionals take the main responsibility.

Let me describe a very common situation. During the TSS development designers and particularly information officers have been likely to convince users that all they need is to implement the TSS and drop in the data. This has been perfectly sufficient - as users are persuaded - not only to support operational tasks but also to get out all possible analytical reports required for business improvements. What is the result? With a few fixed

reports included in the implemented TSS, the users' expectations are not met. They keep on wondering: — *if all the data are available in the database, why can't I drag out all that I need*. Consequently, users consider it a partial or even total failure despite the fact the TSS implemented represents a qualitative support for operational tasks.

Some of us represent things in such a manner, maybe consciously thinking that this little trick would be challenging enough as to draw users into "our" IS world from where there is no way back. Generally, that's how we have contributed to the systematically created **myth on the self–sufficiency** of transactional databases and the TSS.

After being frustrated and disappointed with the Transaction Support System the user community turns a deaf ear to our explanations so that now there is a need for the development of the Decision Support System - an entirely new IS that is going to extend capabilities of the existing Transaction Support System and meet the users' remaining needs.

The reaction of information officers is similar, including the chief information officer (CIO). The approval of new resources - time, people, investments, hardware capacities - for the user that already has a working TSS is often *totally out of the question*.

More surprisingly, the need for the DSS has **not been accepted even by designers**. The primary reasons, that are going to be discussed later on, are contained in some aspects of the DSS development that can be easily misunderstood:

- there has to be a *data model prior to the requirements*; at first sight it does not look like engineering but rather like some kind of a hacker job
- the data warehouse is *awfully denormalized*
- *traditional user interface is not sufficient*; there is a strong need for software solutions atypical for development of on–line transaction processing (OLTP) tools.

2 TSS and DSS – two different worlds

In the last few years the decision support systems have provoked a great deal of *changes in RDBMS technology*. The most important and, at the same time, the most interesting ones are related to the following fields: *data model, diagramming techniques, terminology, DBMS servers, extended SQL, specialized hardware, front-end tools, DBA role, development methodology.*

- The DSS is based upon the data warehouse. It represent a multi-dimensional architecture of a data model instead of a relational one. The data warehouse is excessively denormalized thus contrasting highly normalized and structured TSS database
- The DSS uses the star-join diagramming technique instead of the entity-relationship (E-R) diagramming technique.
- The DSS introduces some new terms. This is quite important if we accept the assertion that what we are capable of thinking depends on the language we use for thinking.
- Appearance of specialized DBMS servers is connected with the DSS, although their usage is not mandatory. When implemented, they are capable of retrieving amounts of data inconceivable to ordinary RDBMS servers.
- Appearance of the extended SQL is connected with the DSS. Standard SQL is an inferior language for the purpose of business analyses. For example it has no possibilities that depends on the sort order of the records, for example moving averages, rankings, n-tile orderings. There is no important statistical functions like variance or standard deviations. There is no stop mechanism for the extremely long lasted queries.
- Nowadays, there are specialized hardware in the market that can satisfactorily handle terabyte databases and tables with a million million records. Some of the features are: parallel processing architectures and large quantities of physical disc units e.g. hundreds and even up to a thousand disc units.

- As a front-end tool DSS uses OLAP tools, instead of OLTP tools. OLAP tool is a user interface adjusted to the analytical processing that is completely different from the transaction processing.
- The role of a data warehouse administrator (DWA) emerged. The management of the data warehouse, that is an on-line analytical environment is similar, yet fundamentally different in some important ways from the traditional DBA role.
- There is a great difference between the process driven development of the traditional TSS, and the DSS development, that is data driven. Furthermore, the development of the DSS is an explicitly heuristic process. It is heuristic to such an extent that by great number of iterations it is not only repeatedly changing the user-interface but data model, too. For the TSS development, that relies on data model stability, nothing else could be said but that it is a heresy.

As we know, the need for an organized and methodological approach to the TSS development has been widely accepted for a long while. On the contrary, there is frequently a **lack of awareness** even of the existence of the DSS methodology. These considerations are put forward as an introduction to the next discussion of the questions: what exactly are the differences between the two methodologies? Are they really so essential? There are not only differences, that facilitate comparative analysis, but also intersections that make difficult the identification of the DSS methodology as one that is completely separate. Some more important methodological aspects, like data modeling, star-join diagramming technique, risk assessment and physical data design are going to be elaborated more widely.

3 Development phases and activities

We could name here, as in other engineering disciplines, three fundamental phases of development: **design, construction and maintenance**. DSS and TSS developments are not exceptions. From that point of view the main characteristics of three stages are applied to both of them: the first phase is concerned with *logical design*, the second one with building and implementation, in other words, with *physical design*, and the third one deals with the *working product*.

The TSS development activity (shown in Figure 1) is represented as a *combination of two approaches: systems development life cycle (SDLC) and controlled prototyping*. Traditional SDLC is applied to easily understandable IS segments where all activities are expected to be carried out as a sequential process. Prototyping is used for more complex parts. It is a reversible process that drives development process back to repeating a string of activities. Backwards arrows specified in figures detect potentially reversible activities.

The DSS development activities, shown in Figure 2, are not a sequential process but an *exceptionally heuristic one*. This could be spotted instantaneously from the great number of backwards arrows.

The intensive heuristic character of the DSS contributes to the confusion. For instance, it could make the designer unaware of the development activity he is currently performing. This is pretty essential, because when you know where you are it is quite easy to continue the process without missing anything.

For better understanding it is important to point to some facts valid for both the DSS and the TSS development:

- It is not important how long it takes for some activity, as long as it is not skipped over
- Design process is full of iterative reverse cycles. Therefore note that if we are once in a construction phase, it doesn't mean that we will not come back to some activity belonging to the design phase
- Different parts of projects (software) are usually neither at the same point of development activities nor phases.

All activities shown in figures along with their deliverables (outputs) are discussed from the aspects relevant to the comparison of the DSS and TSS development methodologies. It can be noticed that there are plenty of confusing elements like: common activities with conforming techniques, common activities with opposing techniques, missing activities in the TSS development, missing activities in the DSS development, activities with the difference in their order of appearance.

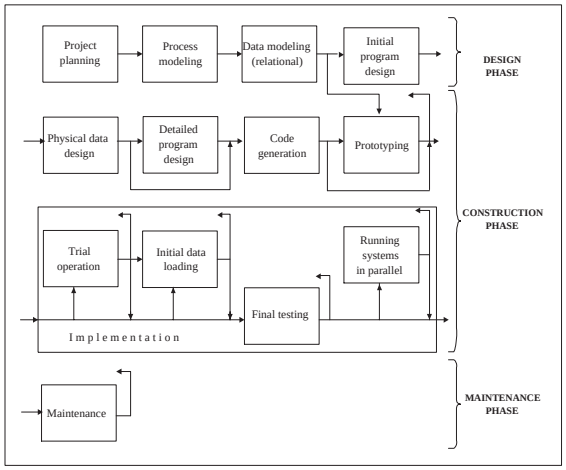

Figure 1: TSS development phases and activities

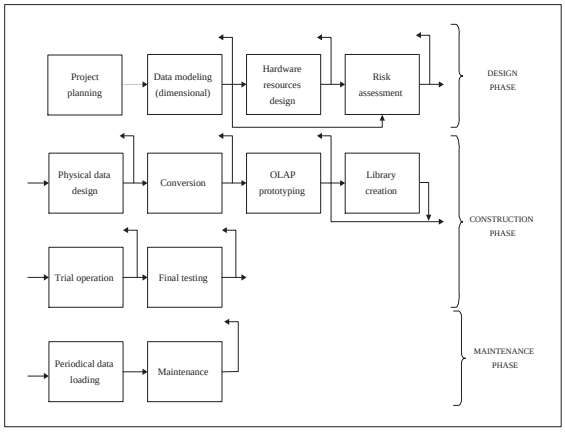

Figure 2: DSS development phases and activities

3.1 DSS concepts

There are contradictory explanations for many terms in the DSS area. In order to follow the text more easily, I will first explain a few terms that are substantial for the clear understanding.

♢ MULTI–DIMENSIONAL MODEL (MD MODEL)

The two–dimensional model represents a coordinate system or a table where rows and columns are the two dimensions. The three-dimensional model can be imagined as a cube. If there is a real value assigned to each of the dimensions there is a concrete business measurement (fact) at their cross section. The multi-dimensional model consisting of three or more dimensions has no physical interpretation. An MD model is graphically represented by a star-join schema, star schema for short (Figure 3). The term is used with or without the multi prefix depending on the context. There are few variations: (multi) dimensional model or cube, (multi) dimensional diagram, (multi) dimensional database.

♢ DATA WAREHOUSE

The data warehouse is a DBMS that consists of a set of MD models. It represents centralized and integrated corporate data. Besides, it enables cross departmental and cross functional analysis.

♢ DECISION SUPPORT SYSTEM (DSS)

It is an information system consisting of five components e.g. layers. Upper components are: OLAP tool, data warehouse and, as the middle component that enables the communication of the previous two, aggregate navigator (see section Physical data design). Lower components are: legacy systems and the transformation component that extracts data from legacy systems, transform, clean and loads data into the data warehouse.

♢ OLAP TOOLS

It is a front end interface that enables preparation and generation of the queries against the DBMS and presentation of the answer set of data received from the DBMS. For the purpose of this text, all span of functionality is comprehended under the term OLAP, e.g. query, reporting, analytical and data mining functionality. The OLAP tool is not aware of the existence of the aggregate navigator component. comprehended

4 Design phase

4.1 Project planning

♢ TSS DEVELOPMENT

Within the project planning, the starting activity of the TSS development, the following tasks should be accomplished: to define what business objectives are meant to be achieved, to describe the planned information system and to divide it into subsystems, to set priorities and deliverables, to determine resources and the overall technical solution, to assess the required hardware resources, and to set the project schedule as well as responsibilities.

♢ DSS DEVELOPMENT

On the most general level there is a concordance. All the above mentioned elements are included. However, there are differences in their implementation. It happens mostly under the influence of the vital DSS characteristic – **scalability**. It affects the whole development process from the various aspects. At this point, as we will see, it decreases the scope of the project planning activity in its first iteration.

The first step is to define business areas that are going to be included in the DSS project. We call them subjects of the data warehouse. There is an analogy in the sense that a subsystem is an elementary component of the Transaction Support System and a business area of the Decision Support System. Data that describe a business area, for example orders or sales, is dispersed in legacy system or systems. One business area can be

supported by one part of TSS subsystem, by the whole of it or even by a few of them. One subject is represented by one or more MD models.

One of the business areas should be chosen and divided into subjects. Initially, it is recommended to **take only one subject** and put it through all the stages of development. In the process of selection we rely upon two basic often contradictory criteria. (1) The starting subject should be of the special interest to the user and (2) it should not be too complex . If the most important subject is too complex, we should choose the next on the scale of importance or we should reduce the complexity. There are various methods for reducing complexity considered within the physical data design section.

The number of assumptions should be reduced to a controllable level. It is neither proper to start with a parallel development of the DSS over multiple subjects nor to simultaneously develop all the aspects of an extremely complex subject. The reason for such an approach to the DSS development is its distinctively heuristic character. Experience accumulated through all the cycles of development over one subject will prove valuable at later cycles. Gradual expansion and increase in complexity are an absolutely proper approach that is in accordance with the earlier mentioned DSS feature - scalability.

Radical narrowing of the field of interest and rejection of some aspects of the selected subsystem **is not acceptable in the TSS** development. As an example, we could observe the development of one subsystem, for example invoices. If our development is based on one type of invoice, but in the middle of the process we realize that there are actually several types, it can be destructive for our data model and already constructed software. Overlooking an important component of the system is a serious mistake with a high price to pay as a consequence. The effects are incomparable more disastrous when we detect the fault at a later stage.

Gradual upgrading of the business area or adding a new business area into the DSS **does not endanger the efforts** already involved. Moreover, it brings along a set of benefits: short developmental cycle of the particular segment of the integrated DSS, quick accumulation of experience, a high degree of its useful exploitation, as well as the efficient distribution of resources.

On the other hand, these characteristics at the same time could cause the DSS development problems to be underestimated. The DSS tasks, especially the initial ones, may seem trivial if we look at them superficially. So, we shouldn't be surprised by an attitude such as: "That will be taken care of by one of our staff members when he finds time".

4.2 Process modeling

♢ TSS DEVELOPMENT

The usual method of process modeling is Structured Systems Analysis (SAS). The basic deliverables are data flow diagrams that accurately represent the real system. Somewhat simplified we can say: on data flow diagrams processes are shown, there are data as an input, the processes change them and at the output there are changed data. At this stage we are actually considering the documents that come in and out of the processes. The structure of the documents is not analyzed yet. The process represents a set of tasks being performed on the data. The SAS breaks down the complex problems into the lower levels of complexity.

♢ DSS DEVELOPMENT

The main final consequence of the process modeling is to enable the users of the information system that is going to be implemented to perform data changes. With the DSS there is an apparent difference since **no data is changed by the user**. Data warehouse is a read-only database and there is no need for this activity.

4.3 Data modeling

♢ TSS DEVELOPMENT AND RELATIONAL DATA MODEL

Through the modeling of the operational database we strive towards data normalization in order to support transaction processing. The result is a hardly legible entity-relationship (E-R) diagram that comprises a set of

tables twisted with numerous relationships, many of which are on the compound key bases. Any complex query can be formulated only by the computer professional who knows well the particular database, the field of the business and relational theory. For example, in order to get an information on the sale ratio of a particular product for two quarters of a year in a particular town you have to apply **bottom-up method** based on the knowledge of a variety of facts about the way the data are stored on the database. The query itself should perform aggregations over a huge number of data dispersed in many tables.

♢ DSS DEVELOPMENT AND DIMENSIONAL DATA MODEL

On the other hand, the DSS is based on a (multi) dimensional model represented by a **star- join schema** (Figure 3) that tends to highlight the natural dimensions that will provide simple access to data needed for the analysis. We use much simpler **top-down method** for making a query. Data models represent all necessary business dimensions (e.g. time, product, store) and their single joins attaching them to the central table, the so called fact table. That is the table where there are business measurements. This method provides a natural way of thinking about business matters. The simplicity of the model has three **far-reaching consequences**:

- it enables intuitive implementation even by the user without any relational theory knowledge
- it enables automatic formulation of a query against database via the OLAP tool with no code writing
- it enables quick retrieval of data because of the small number of joined tables and because of a single key on every dimension table

Let us summarize what is characteristic for a multi-dimensional database model:

- It introduces the concepts of the fact and dimension tables and of the fact and dimension tables attributes. Dimension attributes are business aspects (time, department, customer, product). Fact attributes are numerical measurements of the business (quantity, price, ...).
- MD model is an asymmetrical model. In the center of the model there is a dominant fact table, whereas in the E-R model we cannot distinguish what tables are crucial and business measurements are scattered around. It is not uncommon for a fact table to have an enormous number of records, a couple of million even a thousand million records.
- The compound key exists only on the fact table and multiple join is attached only to the fact table. On all dimension tables there is single key and all of them are attached to the fact table by a single join.

♢ DSS DEVELOPMENT: Star–join diagramming technique

It has been known that diagramming techniques play a crucial part in design and development. Diagrams are an essential communication tool. Diagramming technique is needed to enable the interchange of ideas. Their distinctive part is the involvement of end users enabling them to understand diagrams and to participate in development.

The star join schema looks as a normal E-R diagram applied to the model simplified at that extent that at first sight it does not seem worth mentioning. However, the star schema is capable of visualizing all the above mentioned features of an MD model in an extraordinary **legible way**. It represents a very important part of documentation and it is an excellent intermediary for the use of the OLAP tools. Due to all this, the star schema is an important instrument in the implementation of the DSS concepts.

♢ DSS DEVELOPMENT: DIMENSIONAL DATA MODELING

An MD model is defined by dimensional data modeling. The result of the modeling is a star- join schema. These are the steps of the modeling:

1. Fact attributes are defined, for example price and number of units

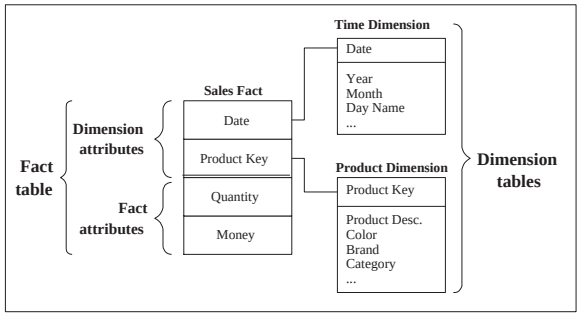

Figure 3: Star-join schema

2. Business dimensions needed for the analyses are defined. First, we define the time dimension which is almost always present and then the other dimensions, for example the product dimension and the customer dimension. The granularity of the fact table is settled by business dimensions.
3. Other attributes of dimension tables are defined.

The following elements are added to the model **through iterations**: new dimension tables, new fact attributes and new dimension tables' attributes. Iterations are performed by the multiple repetition of three steps. It is not only the standard procedure but it is often required to return from some of the later activities in order to modify the MD model. That kind of approach is opposite to the TSS development where it is insisted on the **stability of the relational database model**.

There are different opinions about which is the right sequence of steps to build an MD model. The above said sequence of steps is ordinary more applicable on simple MD models when there are no doubts within the departmental management about the business facts that are relevant for the business analysis.

In case of modeling a complex business subject there are rarely doubts about the time granularity. In this case, it is recommended to first define whether the analyses needed by the management should be made daily and/or monthly etc. Then, we define other business dimensions and, after that, we try to define the first few fact attributes. Additional fact attributes will be established in later iterations.

♢ DSS DEVELOPMENT: SCALABILITY

As it has already been mentioned, scalability is an extremely important feature of the Decision Support System. It is important to realize that by upgrading and extending of the DSS we expand the analytical possibilities without endangering the functioning of the already introduced system. We can scale to new dimensions, to new facts, to a large number of subjects and to a large number of data. Here are some examples.

- In case of the addition of new fact attributes or/and new dimension tables or/and new query possibilities of a higher granularity (e.g. from store to region level) all that we have to do is to embed new options in the OLAP tool and to load new data into the data warehouse. There is often no need for modifying the existing programs for the extraction of data from the legacy system, because data can be derived from the existing data warehouse.
- When we include the possibility of a lower granularity (e.g. from monthly to daily), along with the above said, we add the new dimension of the lower hierarchical level to one of dimension tables and therefore we need to change the dimension and the fact table key

The addition of new subjects to the DSS means there is a need to maintain the data warehouse integrity.

- Some of the changes cause the significant increase of the data amount in the data warehouse thus reducing the performance. We solve this problem with the help of techniques noted later in the text.

4.4 Initial program design

♢ TSS DEVELOPMENT

This is the first time we actually design the front end tool. Data flow diagrams and E-R diagrams are used as input documentation. Software units are specified along with their connections, which gives us the initial architecture of the future front end tool. It is specified **what** should be accomplished in software units e.g. software outputs and **against what data of the logical data model** they are going to operate. At this stage it is not going to be evaluated **how** programs should operate.

Software units can be divided into standard and non-standard. The standard units are, for example: master table browsing unit, browsing unit for the master-detail pair of tables, record update unit, record insert unit. Assuming that we shall use any type of application generator, for that kind of software units it is sufficient to specify the type of the unit and tables against which the process operates. We prepare more descriptive specification for non-standard units.

♢ DSS DEVELOPMENT

The analogue observation as for the activity of process modeling is valid for the DSS development. There is no need for programs that will enable users to change data because a data warehouse is a read-only database. The front-end tool should have the ability to retrieve data from the data warehouse and to represent it with the help of various OLAP techniques. That kind of tasks is going to be solved by building the OLAP tool during the prototyping activity. Requirements are still not defined at this point, although we are already familiar with data.

4.5 Hardware resource modeling

♢ TSS DEVELOPMENT

One of the deliverables of the previous project planning activity is the specification of the necessary resources. During the TSS development hardware load can be assessed more precisely because it is **not dynamically changing** as it is during the DSS development. Therefore, it is not treated as a separate activity.

♢ DSS DEVELOPMENT

The load of the hardware resources is extremely **dynamic**. There are several conditions that force the resources to be re-examined through the risk assessment activity. If it is necessary, i.e. if the existing hardware resources do not provide the required performance or storage capacities, they should be remodeled.

More attention then in the TSS area should be paid to network. The TSS usually retrieves a row of data in a time, while DSS usually retrieves tens, hundreds and more rows. Network traffic in the DSS client/server architecture is very often a bottle neck for achieving the required performance.

The time consuming procedures of loading, indexing and retrieval of the large amount of data are exclusively DSS area problems. The characteristic of the really big data warehouses is the utilization of the special servers with parallel processing architecture and large quantities of disc units.

4.6 Risk assessment

♢ TSS DEVELOPMENT

The risk assessment objective is to establish to what extent the project resources and performances of the future system come into danger by the implementation of the changes. In the TSS area, changes are **driven by the processes** (see section Prototyping). The risk assessment is not treated as a separate activity but as a prototyping sub-activity, since there exists a fairly solid base for the long term assessment of the hardware resources and performances. The controllable level of the data model changes and of the amount of data is much higher than in the DSS due to these two main reasons:

- To keep the project under control serious changes of the data model is usually avoided.
- In the properly designed TSS the time period of the data represented in the database is already defined in advance during the project planning activity. Usually it covers 6 months, or one or two years. Old data are periodically transferred to tapes or loaded into the data warehouse. After that they are deleted from the operational database.

♢ DSS DEVELOPMENT

Because of the distinctively iterative nature of the DSS the potential possibilities of changes are huge. It is a very dynamic information system. Consequently, risk assessment is highlighted as a special activity of the design phase. The main **causes of the risk** are:

- The changes of the data model (see Scalability under the section Data modeling).
- The changes of the amount of data. It is not only the result of the introduction of the lower level of granularity into the data model. Apart from that, the quantity of data is steadily growing from day to day because of the inflow from the legacy systems.
- The growth of the number of users

Because of the first two causes, we can say that this activity is **data driven**. Performances and hardware resources come into danger as a result of the changes. There are risks not even known in the TSS area. The most specific risks are:

- the growth of the amount of data jeopardizes not only the performances but storage capacities too
- the performance characteristics of the data loading and quality of the loaded data come into danger
- the network traffic is in much greater danger (see section Hardware resource modeling)

While in the TSS, the relation towards changes is restrictive, the DSS is **open to the changes**. The expansion and/or changes of the hardware, software and data model are a standard approach. There is a variety of methods and techniques that will be applied prior to the refusal of the changes:

- different data modeling techniques (see section Physical data design)
- remodeling of the hardware resources
- distribution of the system over a number of the databases and/or servers
- moving of the obsolete or so called aged data to on-line devices for mass-storage, e.g. optical discs and HSM devices (juke box) or to some slow medium for later use

5 Construction phase

Previous activities deal with the logical design. At present, there begins the physical implementation of the specifications produced in the design phase.

5.1 Physical data design

Through the activity of data modeling in the TSS and DSS we get the E-R diagram or star schema respectively as an output. It is a logical data model of the database that represents the conceptual view of the database. Both logical data models, relational and (multi) dimensional, should be disburdened from the physical implementation. Physical data design actually means the tuning of the logical database model. The goal is to achieve the satisfactory performances of the system. It is accomplished mostly by the help of **denormalization method** both in the TSS and DSS, but **different techniques are used**. In the DSS area normalization method could be used as well. The changes of the DSS data model are **incomparably greater** than those of the TSS data model.

♢ TSS DEVELOPMENT

Since a long time ago the practice has acknowledged that the data model, which respects all the theoretical postulates of the normalized relational data model, has unacceptable, sometimes even catastrophic performances. That is the reason why redundancies are built in, which helps to gain dramatic improvements of the retrieval time. Physical data model is represented with the same E-R diagram. The most usual techniques are:

- surrogate key: the addition of the single key or the compound key with the minimal number of elements
- vertical partitioning: long and/or rarely used attribute is moved to the separate table
- redundant attributes: the goal is to lower the number of tables in joins

♢ DSS DEVELOPMENT

Although the logical data model of the data warehouse is already denormalized by definition, if it is necessary, it is going to be further denormalized by the help of different techniques:

- horizontal partitioning: partitioning of the fact table into two or more fact tables after one of the elements of the fact table key which represents one of the dimensions; for example, partitioning after the geographical regions
- multi granularity level: lowering the granularity of the fact table; two or more instead of one fact table are used; dimension table is not joined only to the base-level fact table any more but to the aggregated fact table(s) too; fact constellation schema is used (Figure 4)
- surrogate artificial key: single key is generated for the aggregated fact table(s); it is possible when the aggregated fact table is going to be accessed exclusively across the base-level fact table
- dimension table normalization: generation of two or more dimension tables from one dimension table; each smaller dimension table is pointing to one of the aggregated fact tables; in fact, combination of the normalization over the dimension table) and denormalization over the fact table method is applied; snowflake schema is used (Figure 5)
- aggregate fact records: the number of tables stays the same; dimension table records are added for every record of the higher level of hierarchy, and for each of them one record is added into the fact table; the level attribute must exist in the dimension table

The constellation schema and the snowflake schema are variants of the classic star schema. They are used for the physical implementation of the data model logically described with the classic star schema. The prestored aggregates are included in those schemas.

Some of the above techniques are used in case when there are dimensions with hierarchies. For example, product-brand-category hierarchy in the product dimension. There are different attitudes about what the best approach is. That also depends, to a great extent on the MD model, on the amount of data and on the complexity of the requirements included in the OLAP tool.

♢ DSS DEVELOPMENT: AGGREGATE NAVIGATOR

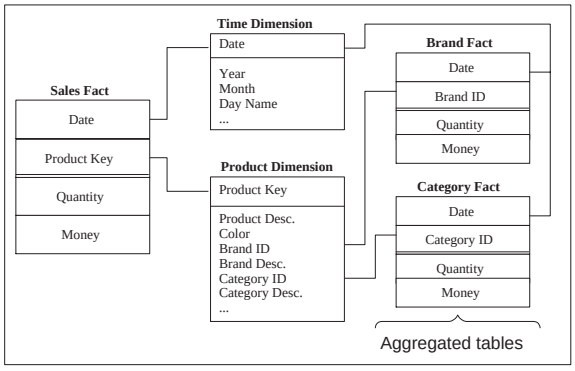

Figure 4: Fact constellation schema

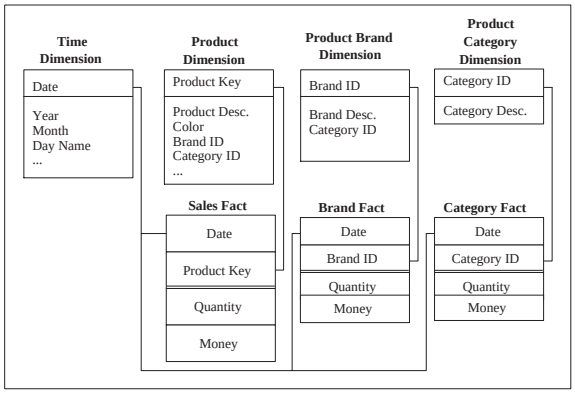

Figure 5: Snowflake schema

In the full and proper DSS implementation, the OLAP tool is not directly implemented over the data warehouse. There is a component placed between them with the task to receive SQLs and to transform it in the form that will **take the advantage of the prestored aggregates**. This component is a software module called aggregate navigator. The use of the prestored aggregates (pre-calculations or consolidated data) is one of the most significant concepts of the DSS, since it is the **most efficient method for the control of performances**. There is another very important benefit of the aggregate navigator. This enables the collection of the statistics of the users' utilization of the aggregations. The analysis of the statistics provides the possibility for the extremely fine tuning.

The aggregate navigator knows if there is a need for the transformation of the SQL and of the ways of obtaining the optimal SQL owing to the knowledge about the available prestored aggregates that is stored in meta-data. The seductive strength of the navigator lies in the fact that nor the user neither the designer of the original star-join schema do not need to be aware of the physical implementation of the multi-dimensional model. That is the job of the data warehouse administrator. The OLAP tool that sends queries against the data warehouse may persistently generate simple SQL syntax based on the registered simple star schema. Of course, such a valuable solution is not free of cost. The use of the aggregate navigator makes the data warehouse administration substantially more complex.

5.2 Detailed program design

♢ TSS DEVELOPMENT

Inputs are the physical data model and all the specifications produced through the initial program design activity. There is no need for this activity for the majority of the standard software units. In that case, the specifications of those software units are entered through the activity that follows - the code generation. Here, it is specified **how** the program should operate for the rest of the units logically designed during the design phase. The physical data model is taken under consideration due to the possible changes of the logical data model during the physical data design activity.

The method of the functional decomposition is used again but to the lower level of details. In that way we break down the complex problems atomizing processes to the level where "point and change" access to data through the user interface could be applied. Common techniques are: procedure flowchart, pseudo code, structured English, navigational diagrams, action diagrams, decision trees, decision tables. As opposed to the DSS environment, the concept of the row level access to data is emphasized here.

♢ DSS DEVELOPMENT

Since the initial program design activity does not exist in the DSS development, there is no need for the detailed program design. The development of the front end tool proceeds in the TSS through several activities placed on different levels of the development life cycle (see figure 1). All of them have to gradually build the front end tool as a goal, while in the DSS the whole activity of building the front end tool is performed through the OLAP prototyping.

5.3 Code generation

♢ TSS DEVELOPMENT

The tools for code (application) generation are being used. In the case of the graphical environment we are dealing with object-oriented tools. The specification of the parameters is used to manipulate these advanced development tools. It is used various documentation that resulted from the initial or detailed program design activities. During the process of development parameters should be specified, screen forms edited, and if necessary, parts of the manual code embedded. The more atypical the software unit is, the bigger the need for manual coding is. In some cases 3GL is appropriate.

♢ DSS DEVELOPMENT

Analogously with the observation for the previous activity, this activity is not performed.

5.4 Conversion

♢ DSS DEVELOPMENT

The goal of this activity is to extract data from the legacy system, to transform, clean and integrate data and to load the data warehouse. Luckily, we need not perform this activity in the TSS environment!

At first sight, the task does not seem particularly demanding. However, in practice, **80 per cent** of the whole DSS development is dedicated to tasks conducted here and to the analogous tasks during the maintenance phase [5].

The results of this activity are transformation programs, metadata and a full data warehouse. Metadata store the knowledge about the data sources and about the ways of mapping data from the sources into the data warehouse.

This is a complex problem. The biggest obstacles are how to achieve **data quality** in the data warehouse, as well as the enormously time-consuming data loading. Legacy systems are actually responsible for the problems of data quality. Here are some of the reasons:

- data inconsistency, as well as data redundancy within an operational database
- data inconsistency, as well as data redundancy across different operational databases
- so-called ”dirty data” in operational databases
- incomplete documentation.

During the DSS development one can frequently notice an astonishing lack of the quality of data extracted from legacy systems. The following question arises: ”How is it possible for our operational systems to function at all?”. It is not an exaggerated assertion that data quality is a predominant critical success factor in data warehousing.

When a data warehouse is really large, in other words, there are large amounts of historical data and there is a high level of details represented in it, loading of the data warehouse is a serious bottle neck. This problem dictates a different approach to hardware issues and to RDBM system issues so that there is a need for the specialized hardware and/or RDBM system.

5.5 Prototyping

♢ TSS DEVELOPMENT

The goal of prototyping is to build the final software unit or branches of units. Each and every non-standard unit is a typical candidate for the prototyping activity. The following features are relevant for our observation.

- The involvement of the end user is extremely emphasized.
- The essential sub-activity is testing. Usually, it takes up most of the duration of prototyping. The objectives are to establish whether software complies with the specified requirements as well as to control performances. In case of an inadequate response time, tuning has to be performed. Most commonly, the improvements of SQL code are sufficient, but sometimes there is a need for more radical changes, for example key substitution and other data model changes.
- Another essential sub-activity is risk assessment. During the process of prototyping, the need generally arises for a huge number of changes. The risk assessment objective is to establish to what extent the project resources and performances of the future system come in danger by the implementation of the changes. The most common ones are: modifications of the screen forms, improvements of existing

functions or addition of the new ones. The alterations of the database are accepted as long as they are at a controllable level (for example addition of a new attribute that is not a foreign key). However, the particularly important concept of the TSS development is the stability of a relational database model. It is very problematic to approve the fundamental change as it is to add an entirely new table with the relationships attaching it to the existing ones, to change table key, etc.

- High quality of the project could not be achieved if documentation does not follow alterations accomplished during the process of development. Yet, it is a great burden and a time-consuming task.

We may conclude that prototyping in TSS is a **long-term activity** with an extremely **iterative character**. The need for **manual programming** is highly emphasized. The process of development very often returns to the previous activities in the construction phase. It also returns to the activities in the design phase, usually for the purpose of database modeling. However, this kind of iterations **should be avoided** or allowed only for minor changes. That represents the essential rule for the TSS development. Otherwise, there is a tendency that the scope of the project and its resources could not be controlled.

It should be emphasized that the building of the front end tool during the construction phase consumes most of the duration of the whole development life cycle of the TSS. There are usually 80 to 90 per cent of the standard units in the well designed front end tools. However, only 10 to 20 per cent is needed for their building, while the atypical units take up the rest of time.

♢ DSS DEVELOPMENT

There are **two categories** of the OLAP tools: third party vendor tools and in-house developed custom OLAP tools. The most common approach is to choose an OLAP tool from the first category. In that case there is **no programming** or it is comprised incomparably less than in the development of TSS front end tool.

The second relevant distinction is that in the DSS, there is not only one OLAP tool that is used but two or more, often by different vendors. They are **used in combination** due to their comparative advantages. That is an approach often present in practice.

There is an additional significant difference. It is not difficult to change the OLAP tool by **porting** already implemented query possibilities from one tool into another that is proved better or appeared later. Just imagine what happens in the TSS area when the need appears to port a front end tool. That regularly means that it should be built from the scratch with some other development tool. It is usually such a big thorn in the side that it indicates a **strategic turning point**.

User engagement is an important characteristic as well as in the TSS environment. The difference is in the category of users, since the OLAP tool is **dedicated to the management**.

5.6 Library creation

♢ DSS DEVELOPMENT

Through the activity of library creation a **set of standard queries** is created. It is prepared for the executive mode of use.

Standard queries are queries with **limited** or with no possibilities **for setting additional constraints**. In fact, it is a library of parametrized reports that should meet the requirements of instant availability and **simplicity of use**. Availability is accomplished through the well- organized pull down menus from where we run queries with one click on the button and the result is obtained after no more than a second.

Quite often users refuse to employ complex OLAP tool. For an inexperienced OLAP developer this is an unexpected, though a very frequent phenomenon. There is a wide group of users that do not incline to pivoting or drilling up and down and that hate doing sophisticated analytical explorations by themselves, but prefer simple reports and simple usage. It is true especially for top managers.

Note that these features remind of the Management Information System (**MIS**) and the Executive Information System (**EIS**) approach where it is typical that higher levels of granularity are represented, detailed data are not accessible, the possibility of applying techniques for dynamic analysis is extremely limited, that only

repetitively used queries are included and that it is dedicated to top and middle management category of users. A significant characteristic could be noticed: **the DSS comprises EIS and MIS capabilities**.

The critical period when users give up most easily is during the initial implementation of the OLAP tool. The initial phase is burdened with still not tuned enough performances and without a perfectly designed user interface. When we lose the OLAP user, it is very hard to get him back. It is even harder when he is a top manager. The **important mission** of the library creation activity is not to discourage users during the initial critical period but even to attract them.

The useful set of queries is detected during the prototyping activity when they are stored for later use. We should repeatedly return to this activity and never cease to take care about the maintenance and enrichment of the library.

♢ TSS DEVELOPMENT

The approach described is, in fact, **well known from the TSS** world. However, it is hard to accomplish this task against the operational databases. The standard difficulties are: even simple queries are too slow in the on-line mode, when run in the off-line batch mode, the possibility for dynamically set parameters is missing, it is frequently not possible to retrieve from the legacy systems where data are scattered across several databases.

It is certain that problems in the full implementation of this concept against the relational databases have been the **initial impulses** for the evolution of the DSS point of view.

5.7 Trial Operation

♢ DSS AND TSS DEVELOPMENT

Units and/or branches of the front end tool that is built up and tuned through prototyping activity should be delivered to the user for autonomous trial purposes. User objections are used for the improvement process.

5.8 Initial data loading

♢ TSS DEVELOPMENT

Prior to launching the TSS project, some pre-loading of the database should be made, for example: the list of employees or customers, some structured documents like price-lists. If data exist in legacy systems they can be loaded in the batch mode. Sometimes the user refuses to use the system unless it is pre-loaded. It is an one-time activity that is important as a condition for the start of the next activity, and that is running systems in parallel.

♢ DSS DEVELOPMENT

There is no analogous activity in the DSS development.

5.9 Final testing

♢ DSS AND TSS DEVELOPMENT

Tests are carried out aiming to show whether performances and functionality of the system or a part of the system are in accordance with the previously documented requirements of the user. Through this activity, the construction phase for the part of the system or for the whole system is formally being closed and it is officially delivered to the user.

As far as the TSS is concerned, it still does not mean that the information system is going to be really implemented because it has to get through the great dangers hidden in the next activity.

5.10 Running systems in parallel

♢ TSS DEVELOPMENT

This activity is as crucial as launching of the ship into the sea. The result of the activity is the activation of the continuous real-time operation of the new system and the closure of the old (computerized or still non-computerized) information system. At this moment, the everyday **operational work is in danger**. That is why it is an excessively critical moment. The risk is so great that there is a possibility for the new system not to be accepted at all. In that case, all the hard labor involved in the long lasting development process is falling apart.

To lower the risk, the usual approach is to define the time period during which the old and new systems will run in parallel. The drawback is that the user is additionally burdened. After the period of the parallel work there comes the D day. The definitive closure of the old system is the most critical moment. It is an activity full of tensions and anxiety for all the participants.

It is not a mandatory activity for some software units of a peripheral importance. Therefore, they can be implemented before, as well as after the D day.

♢ DSS DEVELOPMENT

There is no analogous activity in the DSS development, because there is no that kind of risk. Putting the DSS in function cannot endanger the operational job. In case the DSS is not implemented **nothing will stop**. In some cases, the operational tasks that cannot be delayed could become the ally for the TSS implementation which could force the user to use the new system. On the other hand, there is no such help for the DSS. The OLAP tools can stay unused for ages and nothing will happen.

6 Maintenance phase

Now, both systems run in a real-time mode of operation. The maintenance activities last as long as the system. The most important is the role of the DBA. In the DSS environment they changed the name - they are **DWA** now (Data Warehouse Administrator).

RDBMS have to be managed even when there are no new requirements. There are similarities between DBA and DWA but there are great many differences in their roles. DWA should perform some **special types of management** or do the old job in a new way. For example:

- manage the big volumes of data in order to load, index, test the quality and tune them
- manage metadata used for mapping data from legacy systems into the data warehouse
- manage metadata used by the aggregation navigator
- utilization of the specific tools for the data warehouse management
- tune the performance of the specialized hardware

6.1 Periodical data loading

♢ TSS DEVELOPMENT

There is no analogous activity in the TSS development.

♢ DSS DEVELOPMENT

The data warehouse should be **regularly loaded** with the new data. Loading can last for hours and reloading the whole day.

The **data quality testing** and management should be performed in conjunction with every single loading. There is a great number of potential causes of pure data quality. Tests are carried out mainly automatically.

6.2 Maintenance

◊ TSS DEVELOPMENT

It is usually a very **complex and annoying** task to maintain a transaction support system and to support all the changes that happen in real life. It is known from the statistics to what extent it is a burden to keep the existing system alive. There is a tendency that the most of the IT department staff eventually deal with the maintenance activities.

◊ DSS DEVELOPMENT

There is a variety of important tasks that should be accomplished through the DSS maintenance. Changes in the real life are affecting the date warehouse, for example the reorganization. Changes should be applied on the data warehouse too. Performances are continuously monitored since the amount of data is constantly growing. The response time should be tuned through different methods, for example, aggregation navigator management, erasure of the old data etc. Historical data could be saved on some other slow mediums. Inconsistencies detected during the previous activity are corrected. New subjects are added to the data warehouse and accordingly, the OLAP tool is being extended in capabilities. Interestingly, it is hard to satisfy the user. The more querying and reporting possibilities are included in his/her OLAP tool the bigger is his/her hunger for more.

Although it is a complex activity, the occupation of the IT department staff with the DSS maintenance tasks **cannot be compared** with those in the TSS area.

7 Summary

This paper exposes great differences between the development methodologies of the Decision Support System (DSS) and Transaction Support System (TSS). It goes through all of the activities of the full development life cycles depicting those two methodologies as **two distinct concepts**.

Despite all the differences, solutions and knowledge that are introduced by the DSS methodology, seem to be quite understandable for any computer professional with the experience in designing the TSS. But there is an important condition. The old point of view should be discarded and **new glasses should be put on**. However, often an unexpected phenomenon occurs: the resistance of the designers to the DSS points of view. The paper also emphasizes those points of the DSS development methodology that are obstacles for its acceptance. The most important ones are:

- The focus is on the data and their presentation. Detection of the needed data initiates and leads the development. The **data model exists prior to the requirements**. The whole process is to much **heuristic and iterative** when looking through the TSS glasses
- **Multi-dimensional data modeling** is used instead of the traditional relational data modeling. The data model is enormously denormalized, which is unacceptable from the TSS point of view. Bottom-up approach is discarded and top-down approach is applied. The complex entity-relationship (E-R) diagramming technique is forgotten and new diagramming techniques are used. First off all, the star-join schema that enables natural view of the business dimensions and measurements. Also, the star constellation schema and the snowflake schema, which enables the DSS to minimize the amount of I/O's and to lower the resource utilization.
- **Changes of the data model** during the development are enormous. The amount of the data is constantly growing thus threatening the performances. The **aggregate navigator** appears as a totally new concept. The role of the DBA changes and converts to the role f the data warehouse administrator (**DWA**).
- There is a dramatic **reduction of the efforts needed to build a front end tool**. Great possibilities of the usage of the third party vendor tools where there is a minimum or even no need for the manual

programming have emerged. The most demanding area is shifted towards the data warehouse administration and towards maintaining the quality of data, which become the **most time consuming issues in the DSS** environment.

Except the possibility of turning raw data into the knowledge, the most impressive concepts built into the DSS and highlighted in the paper are: **scalability**, the **speed of the development** of the front end tools and **high performances over the huge amount of data**. That is what the today's world of the fast communications, Internet and rapidly changing business conditions is setting as a standard each day more and more resolutely. The goal of the DSS methodology is to provide methods and techniques for accomplishing those concepts.

References

[1] Avison, D.E. and Fitzgerald, G. (1995), *Information Systems Development: Methodologies, Techniques and Tools*, McGraw-Hill

[2] Cippico, V. (1996), *TSS Construction Phase*, CASE 8 Conference, Opatija, 19-40

[3] Cippico, V. (1996), *DSS Development Methodology*, Information Systems Conference, Varazdin, 305-319

[4] Ince, D. (1994), *Software Quality Assurance and its Implementation*, McGraw-Hill

[5] Inmon, W. (1993), *Developing Client/Server Applications*, J.Wiley & Sons

[6] Inmon, W., Zachman, J., and Geiger, J. (1997), *Data Stores, Data Warehousing, and the Zachman Framework*, McGraw-Hill

[7] Kimball, R. (1996), *The Data Warehouse Toolkit*, J.Wiley & Sons

[8] Martin, J. and McClure, C. (1985), *Diagramming Techniques*, Prentice-Hall

[9] Pavlic, M. (1996), *Information Systems Development*, Znak, Zagreb

[10] Pressman, R. (1994), *Software Engineering*, McGraw-Hill