Tool-Based Re-Engineering of a Legacy MIS: An Experience Report

Thomas Kudrass, Marco Lehmbach[1], Alejandro Buchmann
Technical University Darmstadt, Dept. of Computer Science, Database Research Group
Frankfurter Str. 69 A, D-64293 Darmstadt, Germany
{kudrass | buchmann}@dvs1.informatik.th-darmstadt.de

Abstract. In the process of restructuring its computing facilities, Rhône-Poulenc Rorer is eliminating its mainframes. As a consequence, existing legacy systems must be migrated. We present an experience report on the use of CASE tools for re-engineering a legacy MIS and illustrate the process of current state analysis and generation of a re-engineered target system. Based on the experience gained in evaluating representative CASE tools for their suitability in the data re-engineering process and the actual use of the ISEE tool we present a list of needed features missing from today's tools.

1. Introduction

The case study presented in this paper was inspired by the decision of Rhône-Poulenc Rorer to give up its mainframe systems by mid 1997. This implies that existing mainframe systems have to be substituted or migrated. Subject of this study is a custom-made Management Information System for in-house controlling data (CMIS) based on an IBM SQL/DS system running on a mainframe.

Because of a lack of documentation and in-house know-how concerning the legacy MIS it has to be analysed and documented before being replaced. In addition, the documentation is supposed to act as an online help system for the MIS administrator. The documentation should further include an analysis of the system's interfaces in order to maintain these more efficiently.

As a result of multiple modifications over time the data structure of the MIS is messy and the data is partially inconsistent. Therefore, it makes no sense to substitute the MIS by a system with the same data structure. Instead, a target model is to be developed.

The capabilities of the MIS exceed the functionality of a data warehouse because it comprises also operational functions that are logically a part of various operational subsystems (see section 2). The aim is to extract these functions in order to assign them to the subsystems to reduce the MIS to a data warehouse system. In this paper, we address three issues: (a) the procedure of analysing and documenting an existing legacy infor-

1. current address: Rhone-Pôulenc Rorer, Nattermannallee 1, D-50792 Cologne

mation system, (b) deriving a target model in an I-CASE-Tool and (c) describing the specification of an ideal documentation and analysis tool.

This paper is organized as follows. Section 2 describes the environment of the case study. Section 3 contains a short introduction into the I-CASE methodology in general and into the I-CASE tool ISEE in detail. We discuss the procedure of documenting the system's current state and deriving the target model of the data structure in section 4. That leads to the catalogue of requirements concerning an ideal business process analysis and data documentation tool. Finally, section 6 concludes with some remarks about the availability of the discussed features.

2. A Case Study: The Controlling Managament Information System

The pharmaceutical branch of Rhône-Poulenc (France) consolidated with Rorer (USA) in 1990. This consolidation constituted Rhône-Poulenc Rorer Inc., a company with 22,500 employees in more than 140 countries. The former German company A. Nattermann & Cie. GmbH in Cologne acts as the headquarter of Rhône-Poulenc Rorer Germany. More than 1,400 employees in production, marketing, administration and sales work with the German subsidiary of Rhône-Poulenc Rorer Inc. in Cologne and achieved revenues of $ 400 million in 1992. Rhône-Poulenc Rorer Germany is one of the top German pharmaceutical companies and it is the biggest company in terms of revenue in production and sale of over-the-counter drugs.

This study [Lehm95] analyses the mainframe SQL- based Controlling Management Information System (CMIS), which was developed by the information system department (now disbanded) of A. Nattermann & Cie GmbH in 1989. CMIS handles and provides in-house controlling data such as costs, revenues, budgets, etc.

CMIS integrates the order, production and accounting systems in a broad sense. Most of the master data originates from the production system, e.g. the *selling unit*. A selling unit contains all relevant information about the sold products. In order to guarantee a minimum of tolerable inconsistency between the material master data in the production system and the selling unit in the CMIS, a daily batch job is executed which updates the selling unit. The accounting system delivers the actual amount of money that was paid by the customers, and the order system passes information about the amount of products sold to the CMIS. This happens daily, too.

The CMIS is divided into several subsystems like master data administration, reporting, budgeting, key customer management, accounting, investment controlling and advertisement expenditures controlling: Within the budgeting module the expected expenditures for the upcoming year applied to each cost center are planned and transferred to the accounting and to the production system in order to control the budgets. Hence, the CMIS contains also subsystems providing calculation functions that need additional manual input. The subsystems investment controlling and advertisement expenditures

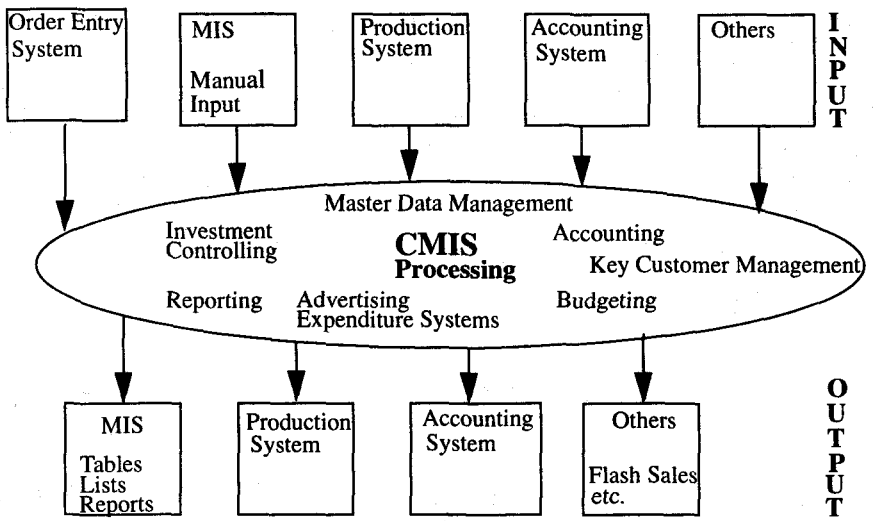

Figure 1: System Environment of the Controlling MIS

collect data about planned investments and advertisements. These data are partially used by the accounting system. The key task of the accounting module within the CMIS is reporting on the sales. The reporting module consists of a collection of standard reports and a query management tool. The standard reports present the development of costs, turnover etc. The reporting functionality could migrate into a future data warehouse system substituting the existing CMIS.

Some key facts about the CMIS are: The size of the data is roughly 1.6 Gigabyte. This quantity is distributed among 250 tables (entity types), characterized by 2000 different attributes. Some of the tables contain more than 700,000 records (e.g. monthly inventory), some only one or two. The SQL system catalogue contains information about 120 different primary keys. Some of these keys occur very often, e.g. the primary key of the selling unit, occurs in 60 tables as a foreign key. There are also tables specified by up to 100 attributes like the master employee data table with a record length of around 2000 bytes.

Figure 2 contains part of the CMIS data structure that will be used as an example throughout the whole paper. Rhône-Poulenc Rorer used to be divided into divisions and subdivisions. After a reorganization of the company the differentiation of subdivisions was eliminated. Due to legal requirements some data (e.g. in the bookkeeping department) have to be available five years which implies the existence of historical data. Therefore, the former one-to-many relationship between division and subdivision was replaced by a one-to-one relationship. A division has several cost centers, e.g. one for human resources, another one for marketing, etc. Every employee belongs to exactly one cost center. A brand has many possible forms of administering (e.g. liquid form or pills). Since pills can be sold in packages of ten, twenty or more units and different laws

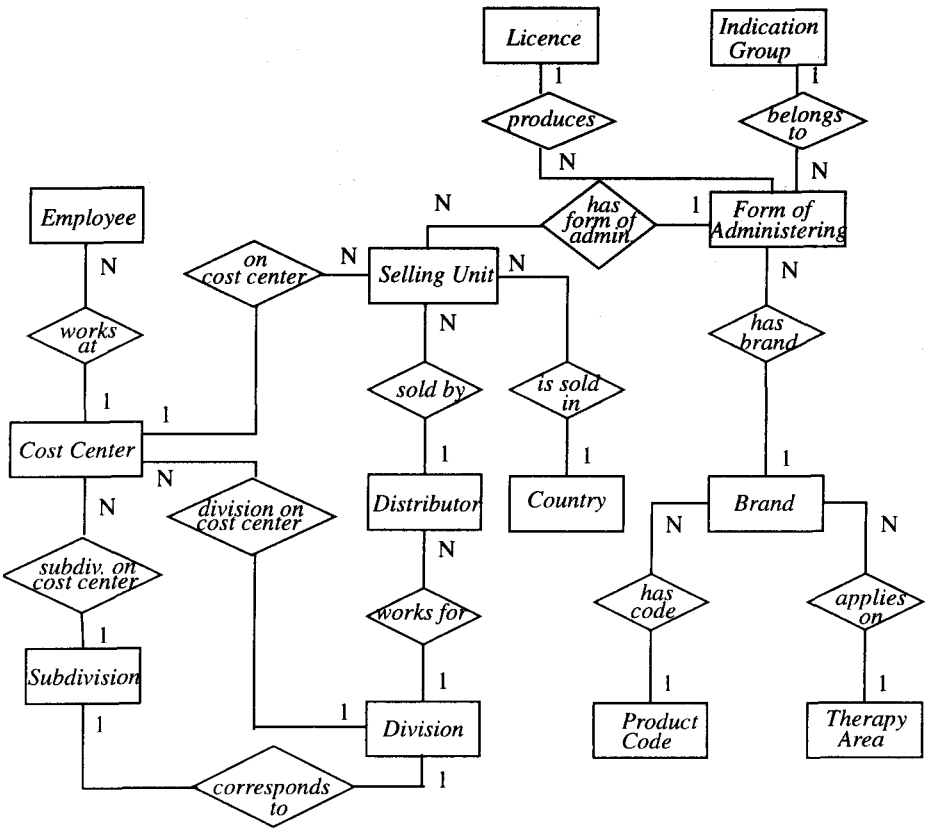

Figure 2: A Part of the CMIS Data Structure

and languages require different packages for each country, various packages, so-called selling units, have to be produced. They are sold by company owned distributors.

3. The Usage of the I-CASE Methodology

3.1 Selecting a Tool

Within the second phase of the software life cycle [Royc70, Somm92], the software and system design, the developer is supported in several tasks by different development tools. We chose three tools which are representative for three different modelling approaches. These approaches are the Structured Design Methodology by Constantine/ Yourdon [CoYo79], the Integrated Computer Aided Manufacturing (ICAM or IDEF) notation of ER diagrams [Chen76] and the ARIS approach (Architecture of Integrated Information Systems) [Sche92]. Because of the widespread use of the IDEF modelling and the Structured Design Method, there are many tools available that are based on them. As representative tools we chose the system design tool ERWin/BPWin from Logic Works [ERwi95] and the I-CASE tool ISEE, version 3.2., from Westmount B.V.

[UMIS93]. The ARIS approach is often referenced with respect to business process re-engineering, especially in the context of SAP/R3. The corresponding product is the ARIS Toolset, our third choice.

Main prerequisites for the application of a tool are:

1. the possibility to document separately single applications in subsystems based on a common enterprise-wide data repository
2. openness and interfaces to other development tools
3. graphical support of data flow diagram (DFD) and ERD techniques
4. access to the data repository
5. a reverse engineering of the data structures

Due to space restrictions we sketch only briefly ERwin/BPwin and ARIS, ISEE is described in more detail in section 3.3.

The DB design tool ERwin by Logic Works represents every application system by a single ER diagram which implies the creation of separate data repositories for each system. Thus, it is impossible to manage several systems and their interfaces (e.g. interdependent data, batch jobs) centrally in one repository. The ARIS Toolset 3.0 comprises a process modelling component as an essential part which supports the detailed registration of all business processes and participating objects. By means of reference models comparison analyses with limited expressiveness can be executed. Within the data view such comparison analyses - like a reverse engineering of the existing data structure - are not supported. There is no possibility to create SQL scripts from the given entity relationship diagrams (forward engineering) in a straightforward manner. Due to the strong coupling of the ARIS Toolset to the SAP software there are few interfaces to other database design and CASE tools.

Table 1 shows that ISEE was the most appropriate tool among the candidates in meeting the above requirements. That led to the decision to apply ISEE for analysis and documentation of all existing application systems at Rhône-Poulenc Rorer.

Tool	Subsystens/ Common Repository	Openness and Interfaces	DFDs and ERDs	Access to Data Repository	Re-Engineering of Data Structures
ERwin & BPwin	-	√	√	(√)	√
ARIS Toolset	√	(√)	(√)	(√)	-
I-CASE ISEE	√	√	√	(√)	(√)

–	not available
√	available
(√)	partially available

Table 1: Evaluation of CASE Tools according to the Selection Criteria

Regardless of the positive evaluation of ISEE it can be stated that its focus is more on the Lower CASE than on the Upper CASE because it is not mandatory to pass through the phases of the Upper CASE during software development. They serve primarily the development preparation and the documentation of the procedure. Hence, online-queries are only limited and analyses of current and target states are impossible. Furthermore the reverse engineering only deduces table structures without relationship types. Because ISEE is a so-called multi-user tool enabling the definition of subsystems within a project it makes possible an appropriate mapping and separation of different application systems of the corporation using one data repository.

In section 5 it is shown that no approach or tool meets the additional requirements we extracted from our experience in the given CMIS environment.

3.2 The I-CASE Methodology

The process of software and system design is commonly divided into subprocesses which is reflected in the structure of the CASE tools. The terms Upper CASE and Lower CASE arrange CASE products in groups depending on their focus. Upper CASE products serve the corporate planning, the description of the corporation and its plans. Lower CASE supports among other things the non-graphic generation of specifications as a first step in the straightforward mapping of data modeling (e.g. data definitions, triggers, stored procedures) to programs. Integrated CASE tools (I-CASE) comprise both the Upper and the Lower CASE using a common repository. The repository, or data dictionary, acts as documentation and information base for all structured and standardized data which are being created or accessed during system development. The repository gives the CASE components access to the necessary information, keeps this data consistent and non-redundant and, hence, guarantees a seamless integration of these components.

3.3 The I-CASE tool ISEE / Westmount

3.3.1 General structure

ISEE from Westmount is based on the Structured Design methodology by Constantine/ Yourdon [CoYo79] supporting I-CASE. ISEE divides the development process into four phases, the analysis, architecture, design and programming phase (see figure 3). The analysis and architecture phases represent the Upper CASE, the design and programming phases the Lower CASE. All phases are passed through gradually, beginning the next phase after the end of the previous one.

ISEE distinguishes three modelling viewpoints: processes, data structures and user interfaces. For each combination of phase and view several modelling techniques are available to the developer.

Within the data view the developer can use the entity-relationship modelling technique in all phases. The process modelling is supported by ISEE only in Upper CASE by a

Data Flow Diagram (DFD) editor in the analysis phase and by a System Architecture Diagram (SAD) editor in the architecture phase. A SAD enhances the data flow modelling by taking required hardware and software components in the system development into account.

Besides the editors, ISEE incorporates a report generator, which gives the developer access to information in the repository and several checking mechanisms, so-called check routines, for the consistency maintenance of the whole system which is to be analysed.

The subsequent discussion is restricted to the two Upper CASE phases because the Lower Case plays a minor role in the analysis and documentation process. Due to the restriction to Upper CASE the consideration is reduced to the DFD and ERD techniques.

			VIEWS		
			Process	Data	User Interface
Upper CASE	PHASES	Analyst	DFD	ERD DSD	
		Architect	SAD	ERD DSD	FSD
Lower CASE		Designer	SCD	ERD	FSD
		Programmer	ERD PRG	ERD PRG	PRG
REPOSITORY					
Techniques & Diagrams					
DFD: Data Flow Diagram DSD: Data Structure Diagram ERD: Entity Relationship Diagram			PRG: Program Editor SAD: System Architecture Diagram SCD: Structure Chart Diagram		

Figure 3: Main Components of the I-CASE tool ISEE

3.3.2 Top-Down Design in I-CASE - Analysis Phase

The design begins with the DFD, which captures data flows between processes and / or data stores from the process view. DFDs allow to form arbitrary hierarchies of complex processes counting each hierarchy level by a current number beginning with level 0 at the topmost layer. While modelling the data view the data stores are analysed and a more detailed specification of the data stores through ERDs is developed. Graphically speaking, each ERD is attached to a data store. Each data flow can be specified in more detail by DSDs. The references originating from these attachments establish relation-

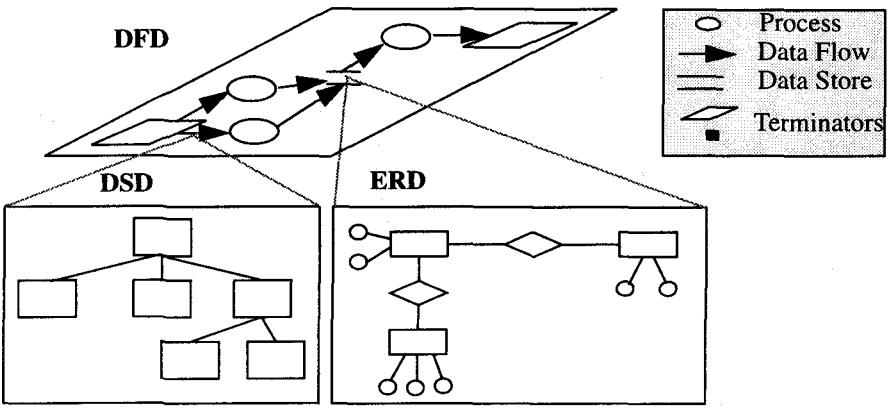

Figure 4: Interdependencies between DFDs, DSDs and ERDs [Prui93]

ships between different diagrams. Those references are managed centrally in the data repository (maintained in a project database in ISEE), which leads to a network of diagrams representing the system to be modelled. This network is built at the bottom layer. ISEE permits forming a hierarchy by creating a project database for each project and decomposing a project into several subsystems.

In the first phase, the analysis phase, a project database is created and the necessary subsystems are defined by a level 1-DFD. The division of subsystems can be done according to the underlying requirements dependent on organizational aspects, applications or tasks. Those classification possibilities of subsystems are valid for all subsequent phases, too. Following this stage the current state of the existing systems is understood by decomposing each level-0-subsystem (their totality is equivalent to the level-1-DFD) of the project. When obtaining a predetermined degree of accuracy the data structure is documented by ERDs.

Only after finishing the current state analysis the derivation of a target model takes place in the second phase, the architecture phase. This phase is characterized by a modification of existing data structures due to revealed anomalies and disadvantages. See section 4.2 for a detailed description of the procedure with an example taken from the CMIS.

4. The Case Study

4.1 Documenting the Current State at Rhône-Poulenc Rorer

This section deals with an actual analysis and documentation procedure using the I-CASE tool ISEE of the existing CMIS at Rhône-Poulenc Rorer. The repository database used in the case study was based on the Ingres database system running on a Unix machine. The procedure of documenting the existing system consists of the following four steps: (1) set up the ISEE project, (2) import the existing datastructure, (3) design the DFDs and (4) design the ERDs.

(1) Set up the ISEE project

First of all we have to define a project database, e.g. the project "Analyse all of the company's systems". Therefore, we name our project "rpr_cologne.pdb". Afterwards we design the level-1-DFD by decomposing the process 0 = "rpr_cologne" into the company's organizational departments like controlling, research, production etc. As a result of this decomposition the subsystems are defined.

(2) Import the existing data structure

In the next step we have to prepare the existing data structure for the ISEE re-engineering option. The data structure has to be in the same format as the project database. Therefore, we import the data structure from the mainframe to the Unix based Ingres database by converting the SQL system *create table* scripts into flat files. After creating the CMIS data structure in an Ingres database, we execute the option *Generate from Database*, which is provided in the analysis phase of ISEE. As a result ISEE creates one ERD for each table containing the table as an entity with its attributes. In addition to that a so-called Top ERD is designed, where all tables without any attributes are included. Since the tables have technical names, we have to extend those names by adding self-explaining aliases, e.g. entity "T11607" is changed to "T11607@division", to facilitate the understanding. Unfortunately, relationships can not be extracted automatically, they have to be deduced manually using foreign keys.

(3) Design the DFDs

Because the CMIS is mainly used and maintained by the controlling department, we will handle all the collected information about the CMIS in the subsystem "Controlling" (figure 5). In doing so we relate an application system to one department. But there is a possible conflict in the separation of organizational units. Problems may arise from their assignment to application systems because two DFDs cannot correspond with one superior DFD at the same time.

The level-1-DFD of the project occurs within the corresponding subsystems as level-0-DFDs with one process representing the subsystem. Since the CMIS ist not a stand-alone-system we create a process "Interfaces", which handles the data transfers between the CMIS and other systems. The CMIS collects most of its data by using batch jobs. These batch jobs are documented in the level-2-DFD "Interfaces". If the desired level of detail is reached one can stop decomposing processes.

(4) Design the ERDs

When the design of the DFDs is completed the process modelling is finished. In the next step we switch to the data view. The task is now to focus on the data stores in the DFDs by creating ERDs for each relevant store. While designing a new ERD of a DFD's data store it is strongly suggested to use the top ERD as a reference copy and modify it as necessary in order to maintain a consistent data repository.

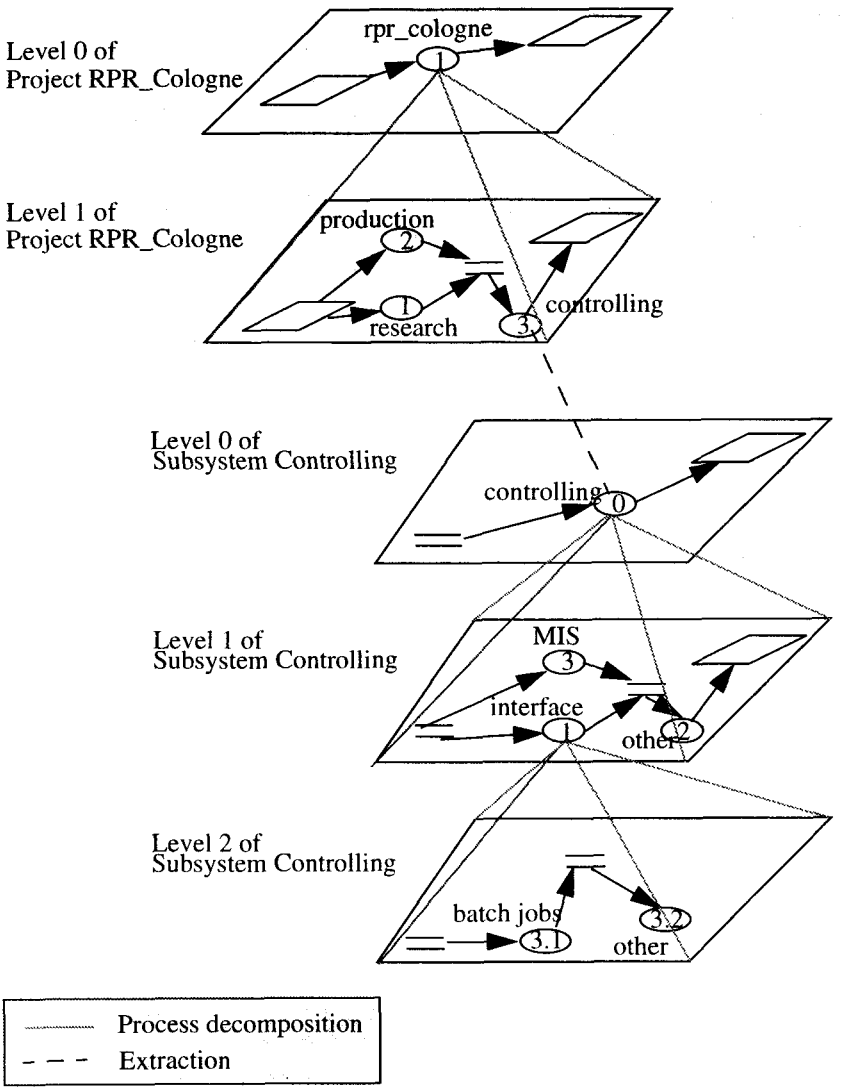

Figure 5: An example of top-down design at Rhône-Poulenc Rorer

4.2 Procedure of Deriving the Target Data Structure in ISEE

As discussed in section 3.5. the ISEE analyst phase is used for documenting the current state of processes and data structures and the ISEE architect phase for modelling the target data structure. Since the project does not include the task of re-engineering the business processes we are not going to modify the DFDs from the analysis phase. This leaves only two steps for deriving the target data structure from the current state: (1) initialize the ISEE architect phase and (2) modify the current data structure.

(1) Initialize the ISEE architect phase

Because we use the same DFDs and therefore the same level-1 DFD, which is the basis for the process of creating the subsystems, we have to create the same subsystems in the architecture phase as there are in the analysis phase.

Afterwards we execute the option "*Import from previous Phase(Analyst)* ", which creates a one-to-one copy of all ERDs and DFDs.

(2) Modify the current data structure

The need to modify the existing ERDs has various reasons. We give three examples:

a) Additional relations in order to enhance query performance

b) Specialization of entities due to encoded semantic information and wrong level of abstraction

c) Decomposition in order to avoid redundancies and to save disk space

a) Additional relations in order to enhance query performance

Most of the daily, weekly and monthly reports are based on queries where relationships between brand and e.g. distributor, country or other entity types are built related directly to selling unit. These queries always use the same two joins to create the relationship between selling unit and brand. In order to enhance the querie's performance an additional persistent table *Selling Unit-Brand* is to be modelled in the target state (figure 6). The update frequency is very low in the given example because the affected data are master data, which means a number of about 5 updates per year (as a result of the introduction of a new brand). There are around 20 weekly and 40 monthly reports using the combination of selling unit, form of administering and brand. In the given example redundant base tables or materialized views are possible solutions. The decision of what approach to use depends on the availability of materialized views in the target system and can only be made after the commitment to a new platform.

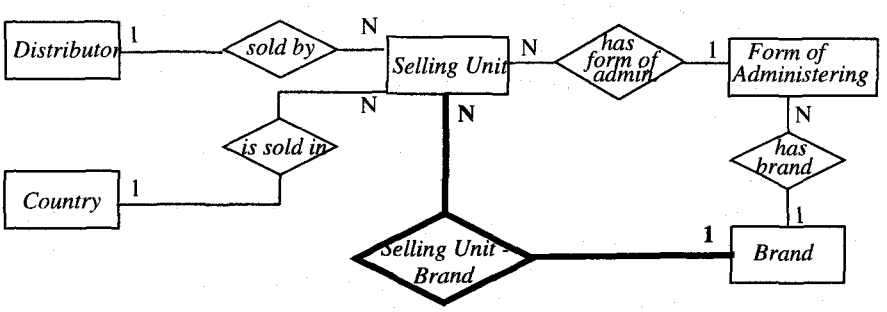

Figure 6: Additional relations (Example)

b) Specialization of entities due to encoded semantic information and wrong level of abstraction

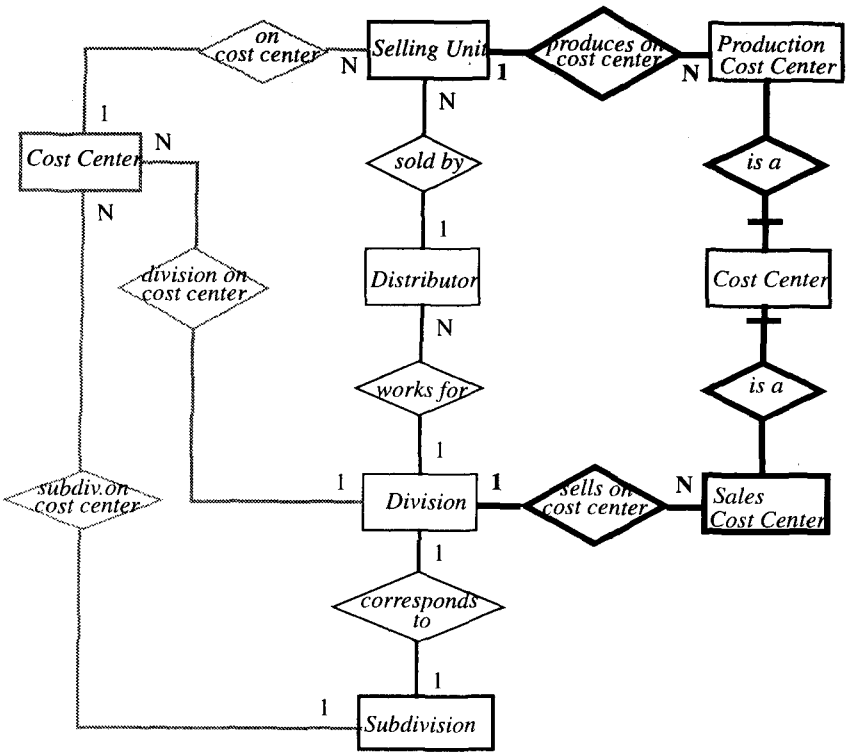

Figure 7: Specialization (Example)

Right now the MIS manages all cost centers in one entity type *cost center* where *cost center id* is the primary key with a domain ranging from 1000 to 9999. The primary key contains encoded semantic information. Cost center numbers 1000 to 1999 are reserved for production cost centers, 2000 to 2999 are reserved for sales activities. In order to integrate that kind of information in the target model a decomposition of the entity cost center has to take place. In our example we broke the *cost center* down into *production* and *sales cost center* and related them to their corresponding entities (figure 7, bold face). The entity *cost center* in the target model results from the disjoint generalization of the two specialized cost centers. Because *division* and *subdivision* have a one-to-one relationship, a relationship between *subdivision* and *sales cost center* is not necessary.

c) Decomposition in order to avoid redundancies and to save diskspace

The MIS manages information about the employees. The main table representing that information is named *employee master data.* This table is characterized by 174 attributes with a record length of 1976 byte. The table keeps the following information about the employees: personal data, children, education, position and status (job

description), wages, savings agreements, pension data, health insurance, social insurance, information about handicaps, etc.

The table structure to be redesigned is characterized by all the teething troubles of the early data processing period causing many problems now. For example, there are 10 columns to store a variable number of children to capture the "worst case" of 10 children.

It is obvious that a lot of disk space is wasted, because information about handicaps exists only for handicapped employees, information about pension data, health insurance, social insurance is only relevant for regular employees and not for temporary workers. Since all former employees receive a company pension the actual table contains more than 3000 records, where only 1400 people are working with the company right now. In order to enhance the performance of application modules reading this table and in order to save disk space it is necessary to split up the table into its logical units, as itemized above.

5. A Requirements Catalogue for Analysis and Documentation Tools

5.1 Features Facilitating the Modelling

In this section we describe features that are desirable and identify tools supporting these features. However, at present no tool supports the combination of all features.

5.1.1 Powerful Database Re-Engineering

We focus on the case of re-engineering relational databases. Given a relational schema, the database re-engineering process aims at the creation of a conceptual model like the ER model by extracting the design elements from the existing tables. There are some authors ([BaCN92], [HCTJ93],[ChBS94]) giving a comprehensive overview about algorithms how to reverse engineer a conceptual model from a source relational schema. They presuppose some knowledge about the relational database by analysing not only the relational schema, but also data instances. There are commercially available tools providing a "Reverse Engineering" option (e.g. Bachman/Analyst [Bach88]), but most of them restrict themselves to reading the data structure, e. g. the data dictionary. Typically, the captured information comprises the names of tables and attributes, attribute properties (like type, length) and primary key information. Assuming the ER model is the target, the result of the re-engineering is a collection of ER diagrams each corresponding to one table or a global ER diagramm corresponding to the whole data structure. In both cases relationships between entities are often ignored.

Based on the assumption that the underlying database supports the primary key concept and there is a consistent naming policy, tools should apply search algorithms, as e.g. proposed in [BaCN92], capable of detecting all potential relationships and transform

them into the graphic ER representation of the tool. Furthermore, if the tool can process the extension of the database to be re-engineered, the cardinalities of the relationships can be determined by simple rules. But it has to be considered that the usage of such rules may lead to wrong results. The main reasons for it are the evaluation of small data sets that do not cover all cases or the analysis of historical data producing incorrect findings. Considering the relationship in our universe of discourse between *division* and *subdivision*, before January 1 1993 one *division* was related with several *subdivisions*. After a reorganization of the company now there is a one-to-one relationship between both entity types. Because the corresponding table contains also the historical assignments due to book-keeping reasons an algorithm would deduce a one-to-many relationship which is not valid at present time.

In order to use the advantages of an automatic deduction of relationships it is desirable to mark the relationships as confirmed or proposed. The analysis is only finished after the verification of all relationships. Up to now there are no tools available providing this kind of interactive capabilities.

5.1.2 Support of Task Orders in DFDs

There are processes that have to be executed in a fixed order like batch jobs with several subroutines (processes). A technique has to be provided or the DFD methodology has to be enhanced to become capable of taking into account precedence relationships. Some mechanisms like task chain diagrams or event-driven process chains as they are available in the ARIS Toolset are good enhancements.

5.1.3 Support of Complexity and Modularity

Because of the complex system structure it is desirable to have facilities to create subsystems within subsystems. Assuming the ER diagram is the dominant conceptual model it is necessary to support the designer in decomposing ER diagrams in different views and to merge smaller diagrams containing common entities. View integration algorithms as proposed in [DaHw84] and [NaEL86] could be a good basis of tools supporting this functionality. The analysis of existing systems requires a modular approach which conflicts with the top-down approach supported by most tools. Because many CASE tools follow the top-down strategy no links between different subsystems can be drawn. Within the considered project logical links like input/output relations and common used data could not be ignored. A technique for logical links between different subsystems is still missing.

As already stated in section 4.1 there is a conflict of the organizational hierarchy vs. the application system hierarchy.

5.1.4 Detection of Synonyms

It is desirable to register synonyms in the tool. We found in the data structure of the CMIS many synonyms. Synonyms are caused by the integration of different subsystems, e.g. the ident number of the type selling unit identifies the same objects as they are stored in the production system identified by a material number. The hardest problem is to define detection algorithms because they require user-specific knowledge.

5.1.5 Global ERDs and Local ERDs

A global ERD is a part of the whole conceptual model that is frequently used in describing the structure of multiple data stores, for example master data and the product hierarchy. By way of contrast a local ERD is used by only one data store. A change of a global ERD has to be propagated to all ERDs where this global ERD has been used as a component. Figure 8 illustrates the use of a global ERD for the product hierarchy, used in various local ERDs. The global ERDs are marked in bold face.

The definition of global and local ERDs can not be described in a uniform way. Commonly, useful local ERDs are composed of one global ERD and directly adjacent entity types. Global ERDs model typically hierarchical master data or recurrent concepts.

When creating local and global ERDs it has to be considered that global ERDs are not only black boxes, i.e. it must be possible to establish relationships between entities inside and outside the global ERD that can be maintained by the data repository.

After modelling the DFDs and the attached ERDs of one subsystem a subsystem ERD could be generated automatically by merging local ERDs. Thus, local as well as global ERDs are parts of the project ERD. That makes it possible to use search algorithms as the basis of an object browser.

The concept of global and local ERDs serves the simplification of the modelling, the consistence maintenance after changes, a better control of complex ERDs and supports the modular procedure common in the current state analysis. A tool realizing this concept should provide the capability of merging single local ERDs to project or subsystem ERDs which leads to more clarity. Up to now there are no tools available supporting this concept.

5.2 Presentation

5.2.1 Interactive Data Repository Query Facilities

To support the DB administrator in changing the schema towards the target model in the re-engineering process it is necessary to have an "Interactive Data Repository Query Tool". While modelling or changing the data structure questions like "Which entities also contain this attribute ?", "In which relationships / subsystems does this entity occur ?" etc. arise very often. In case of erroneous queries or batch jobs[1] the DB administrator needs information about the participating tables and their relationships

Figure 8.a: DFD *"Get Tables of the product hierarchy"*

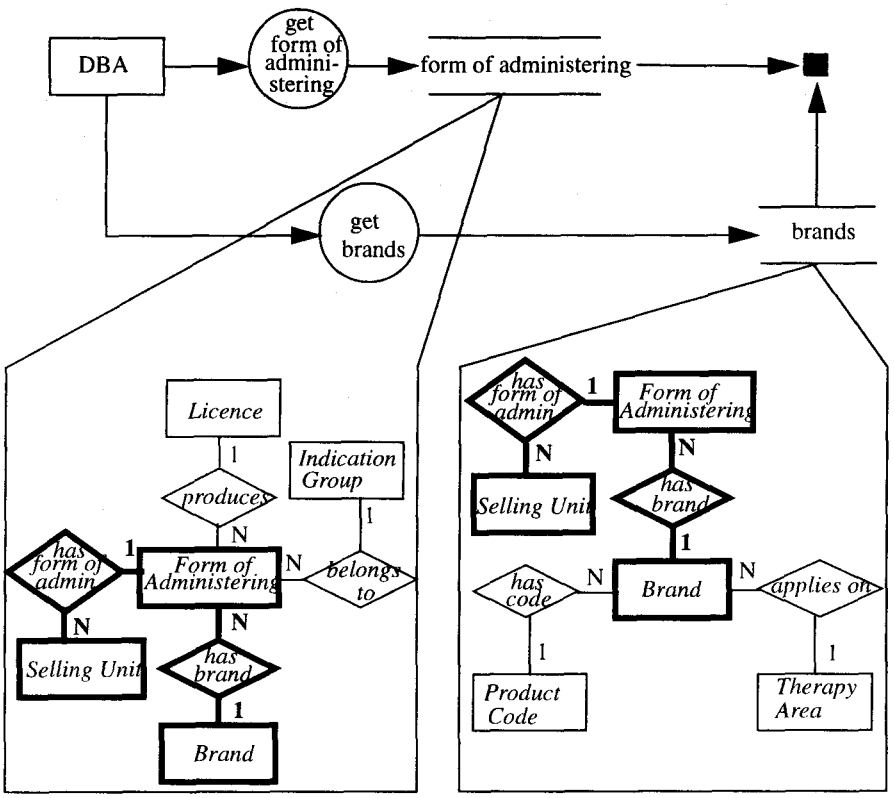

Figure 8.b: Local ERD *"Form of Administering"*

Figure 8.c: Local ERD *"Brand"*

with other tables and the key attributes to assess the consequences of necessary changes. Furthermore, similar information is required for the integration of new batch jobs. CASE tools often provide only report components that do not allow interactive queries on the data dictionary / repository. Our experience shows that it is desirable to enhance documentation and analysis tools by interactive data repository query facilities as sketched above.

If a tool supports several views, e.g. data or process views, a navigation component should be available to traverse an object within different views in order to find out the connection with other objects. Simultaneous presentation of the various views with cross links among views would improve overall clarity. Unfortunately, none of the tools we analysed offers this feature.

1. Batch jobs handle most of the data transfers between the MIS and the other systems.

5.2.2 Requirements of the ER Model Representation

Complexity notation

We want to sketch some features that make an ER diagram more comprehensible according to our experience.
Many facts require the (min,max) notation to capture structural constraints between the modeled entities going beyond the cardinality definitions. In our model a *selling unit* is the unit of sale of a certain product or a certain brand. A *sample* is a free sample in the form of a selling unit that is, however, marked as sample not for sale. A selling unit is assigned at most n or even no samples. A sample is assigned at least one or n selling units, for example the sample "Biovital 50 ml" is used as sample for the selling unit "Biovital 50 ml" as well as for all other offered sizes.

Figure 9: Complexity Notation (Example)

Some design tools [ERwi95] use alternative notations that render readability difficult instead of the clear (min,max) notation.

Representation of relationships

The representation of relationships as diamonds facilitated the clearness of our models. Other notations like the representation in ERwin using edges to symbolize relationships lead to a tangled net of edges when more than 50 entity types are modelled. The ERwin tool only creates one ER diagram for each application system. In our CMIS case study that implies a diagram containing more than 200 entity types and around 800 edges (relationship types). Hence, subsystems have to be formed which is not supported by ERwin.

5.2.3 Storing Unstructured Information

When analysing data and processes it is desirable to keep information about objects (processes, entities) that can only be modeled as unstructured text. This information comprises general descriptions of objects or specific object information, such as its history or the old structure of a table that was changed. The old structure is a specific object information because it may be important for the understanding of old reports or queries. This information cannot easily be structured and has to be stored in so-called memo fields. The main drawback of memo fields is that they are costly to generate without being accessed later. Many tools provide the feature to store unstructured information with the definition of objects. However, unstructured information is rarely retrieved, because it is not visible. The proposal is to mark objects in the diagram (ERD, DFD) whether they carry unstructured information.

5.3 Treatment / Analysis of Current and Target System States

5.3.1 Support of a Dynamic Transition between Desired State and Current State

The long-term goal of the case study is to develop an enterprise-wide data and process model. Application systems are assigned to different subsystems. From the analysis of the current state a target model is derived which serves as a guideline in the decision making process. It must be decided how single modules, systems or subsystems within the target concept can be realized, either by standard software, by enhancing existing application systems or by proprietary solutions. The aim is to update the enterprise model in the future to document when a desired state has become a current state.

At present, the current state is documented in the first phase, the analysis phase, and the target state in the second phase, the architecture phase. Following the top-down approach the transition from the current to the target state is feasible. Once the target state has been realized, it becomes in fact a current state which has to be documented in the analysis phase. This kind of bottom-up shift is not foreseen in the I-CASE tool ISEE we used and in many others.

5.3.2 Analysis Tool with Respect to Current and Target States

Several tool vendors provide so-called analysis components. For example, the analysis component of the business re-engineering tool ARIS compares reference models with desired models derived from them. The result of the analysis is a comparison of single objects. The differences between original and derived model can comprise object changes, insertions or deletions and are displayed by graphical means.

CASE tools contain similar comparison components. They are based on a version comparison related to objects but are only available within one phase (the comparison result is printed as report).

In our case study the current and the desired state are split up into two phases. That implies a comparison component that works across both phases top-down and bottom-up. At the deduction of the desired state from the current state the object structures are affected. Only few application systems are stand-alone systems, most of them deal with interdependent data objects. Structure changes of these objects require propagation processes that guarantee a continuous communication with dependent systems.

6. Conclusion

Both data warehousing and business process re-engineering require a data re-engineering process which has been described in this paper. To develop the CMIS towards a data warehouse system the existing subsystems have to be analysed and described in a clear manner.

Within the re-engineering process a tool is needed, which supports the phase of documenting the current state of processes and data structures of an information system and provides help in deriving a target model from the generated documentation.

This paper addressed the issues of presentation, modelling and analysis techniques, which will ease the process of restructuring existing information systems and maintaining the system's documentation. Hence, the paper is supposed to encourage (motivate) business tool and I-CASE tool producers to develop a truly integrated and comprehensive process and data documentation and analysis tool that meets all requirements as mentioned above.

As far as we know there is no satisfying tool available. Most of our requirements are realized seperately by different software tools, but no tool combines all of them. The main reasons are:

- CASE tools provide most of the necessary techniques, but they always start from zero and strictly follow the top-down philosophy, which makes it impossible to describe an existing system because the analysis requires a modular procedure.
- The presentation of repository data in CASE tools can be considered as rather poor, because most of the time they only provide a more or less standard report interface.
- Business re-engineering and database design tools rarely support the decomposition of systems into subsystems, and a differentiation between current and target state is not intended. On the other hand they offer the necessary presentation techniques.
- Aspects like decomposition and merging of ERDs must be better supported.

Acknowledgements

We would like to thank Rhone-Poulenc Rorer, Cologne, for their support of the work of Marco Lehmbach while doing his Masters Thesis there.

References

[Bach88] Bachman C.: A CASE for Reverse Engineering, *IEEE Datamation*, July 1988.

[BaCN92] Batini C., Ceri S., Navathe S.B.: Conceptual Database Design: An Entity-Relationship-Approach, The Benjamin/Cummings Publishing Company, Inc., 1992, pp. 328-340.

[Chen76] Chen P.: The Entity Relationship Model - Toward a Unified View of Data, *ACM Trans. on Database Systems*, Vol. 1, No. 1 (March).

[ChBS94] Chiang R.H.L, Barron T.M., Storey V.C.: Reverse engineering of relational databases: Extraction of an EER model from a relational database, *Data & Knowledge Engineering*, Vol. 10, No. 12 (March).

[CoYo79] Constantine, Yourdon: Structured Design, Prentice Hall, Englewood Cliffs, N.J., 1979.

[DaHw84] Dayal U., Hwang H.: View definition and generalization for database integration in a multidatabase system, *IEEE Trans. Software Engineering*, Vol. 10, No. 6 (Nov.).

[ERwi95] Logic Works: Erwin Users Guide, 1995.

[HCTJ93] Hainaut J.-L., Chandelon M., Tonneau C., Joris M.: Schema Transformation Techniques for Database Reverse Engineering. In *Proc. of the 12th Int. Conf. on ER Approach,* 1993.

[UMIS93] Westmount Technology B.V.: ISEE User Manual 3.0, Delft, 1993.

[Lehm95] Lehmbach M.: Development of an enterprise-wide data model by analysis of existing applications, Masters thesis, Tech. Univ. Darmstadt, Dept. of Computer Science, 1995 (in German).

[NaEL86] Navathe S., Elmasri R., Larson J.: Integrating user views in database design, *IEEE Computer*, Vol. 19, No. 1 (Jan.).

[Prui93] Pruijt: Structured Analysis and Design with ISEE, Westmount Technology B.V., Delft, 1993.

[Royc70] Royce, W.: Managing the Development of Large Software Systems: Concepts and Techniques, IEEE Report R-77-320, 1970.

[Sche92] Scheer A.W.: Architecture of Integrated Information Systems, Springer Verlag, 1992 (in German).

[Somm92] Sommerville, I.: Software Engineering, Addison Wesley, Wokingham, 1992.