APPLICATION CONTROLS IN A CLIENT/SERVER ENVIRONMENT

Migration of controls towards the database?

Fred de Koning

Professor of Accounting Information Systems at Nyenrode University, The Netherlands
Partner of Mazars Paardekooper Hoffman, Utrecht, The Netherlands

Abstract: In client/server systems the integrity of data processing is threatened by the lack of control over the front end applications. The migration of controls from the application level to the database level might be a solution for this problem. This paper will analyze how essential application controls can be implemented in the database environment.

Validation controls are important to check the accuracy of input, especially in case of manual input. The validation routines at the front end level can be considered to be more important for the support of the user ('self-control') than for the integrity of the corporate data. Essential validation routines should be implemented at the server level. Referential integrity, integrity constraints, stored procedures and database triggers can be used to support validation controls.

For management and control purposes effective and reliable information about business processes is needed. It will be shown that this information can only be produced when the database that generates the information is based on a data model, that reflects the successive stages of the financial, logistic and physical flows in a company. The reliability of information about business processes can be further secured by reconciliation of control totals, generated for each stage. Data about variances between successive control totals, or between control totals and related standards, should be kept available for further investigation.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35693-8_16

M. Gertz (ed.), *Integrity and Internal Control in Information Systems V*

Key words: Application controls, integrity of data, separation of duties, authorization of users, accounting information systems, data model, database management systems.

1. THE CLIENT/SERVER ENVIRONMENT

1.1 Introduction

In modern client/server systems the integrity of data processing is threatened by the lack of control over the front-end (or client) applications. The migration of controls from the applications level to the database level might be a solution for this problem. This paper will analyze how essential controls can be implemented within the database management system, that means as close to the data as possible. It will be shown that some controls can be effectuated by means of an effective data model, that is to say a data model that reflects the successive stages in the business processes. Other controls can be effectuated by means of the facilities of modern relational data base systems.

This paper will not advocate that all controls should be located in or around the data base. In order to give the user adequate support, the front-end applications should provide all the controls that are necessary to prevent wrong input of data. Most of these controls can be categorized as 'self-controls' for the user. The controls that are important for the organization as a whole, that is to say the controls that assure integrity of the corporate data, should be centralized by means of server applications, the database management system, or the structure of the database. In some cases it can not be avoided that these controls duplicate the controls in the front-end applications. This kind of redundancy, however, is generally no problem and can even benefit the integrity of data.

1.2 The client/server architecture

The Gartner Group defines client/server architecture as the application of co-operative processing, in which a programmable workstation executes part of the application, including the user interface. Client/server architectures can take different forms, depending on the proportion of distribution or decentralization of the application software. The well-known Gartner Group model [1] presents different stages in distribution or decentralization of the database itself, the related functionality and the presentation layers.

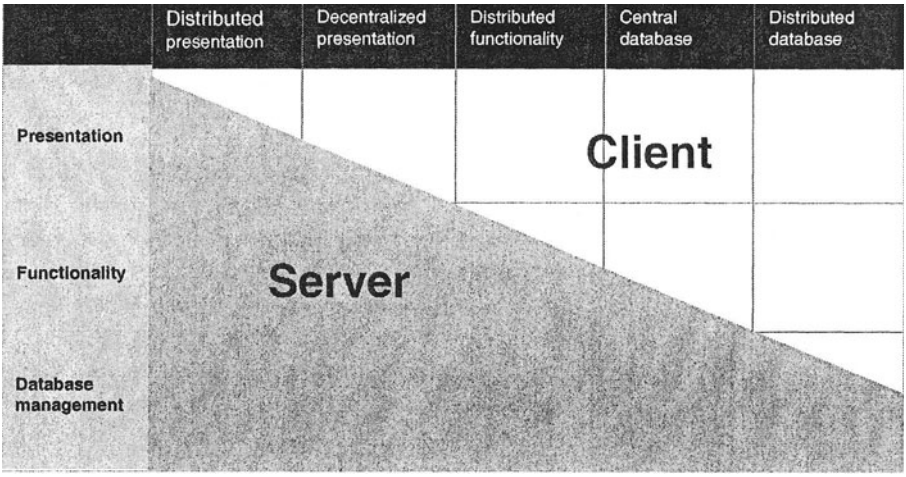

Figure 1. Client/server application model (based on Gartner model)

The client/server architecture will be characterized by several layers of software, such as:

- the presentation layer (Graphical User Interface or GUI)
- decentralized functionality (input and output applications)
- the communication layer (middleware)
- central functionality (central applications and stored procedures)
- the database management system (DBMS)

These software layers can be implemented on different hardware platforms, which are connected by a network. The client platforms can consist of Personal Computers (PCs) or Unix workstations. For the server platform microcomputers, mid-range computers or even mainframes can be used. These servers can be network servers, application servers or dedicated database servers.

The database management system in client/server architectures generally supports a relational database, which is implemented on one or more of the central servers. In the case of distributed databases the data can be stored on the client systems as well, although this is actually not in line with the client/server concept.

1.3 Control problems with the client/server model

The client/server architecture causes some control problems. One of the objectives of using client/server systems is to grant more autonomy to the users. End users can be able to use query tools and to adapt input and output screens to their needs. The control of the application software on the workstation is questionable. A proponent of the client/server architecture puts it this way: "In an application with a GUI the user is in control. With his mouse he selects functions in the menu bar on top of his screen. There are useful symbols (icons) that can be activated by the user....." [2].

In most cases the clients will be Personal Computers (PCs). Healy points out that PCs offer insufficient facilities for the control of applications for transaction processing. The PC is totally unprotected against the professional hacker, who can decode the application on the PC very easily and subsequently manipulate the central database. Healy recommends to restrict the application software on the client to just presentation functions and to install the applications for transaction processing on the central servers by means of so-called stored procedures. These servers can be protected using traditional methods, such as transaction monitors and security software [3].

The solution described by Healy resembles the traditional host/slave solution, that means a mainframe with "dumb" terminals. The traditional controls will hinder the flexibility and user friendliness of the applications. They conflict with the objectives of the client/server-architecture and will be considered by the users as an impediment. In consequence this solution will be doomed to fail. There is a need for solutions, that give the user enough freedom and at the same time protect the integrity of the corporate data.

1.4 A starting point for the control of C/S applications

The challenge is to combine the autonomy of the client/server applications with a tight integrity control of corporate data. The integrity of data can be undermined during the successive stages: input, processing, storage and output. Controls are necessary to prevent human error, especially in the case of manual input. Human involvement in the input stage tends to be replaced by automatic registration of the business processes, but manual input cannot be avoided in many cases.

In order to control manual input and update of data, authorization and integrity controls are needed. Only authorized users should be allowed to modify the data of the corporate database. Besides these modifications should be controlled by as many validation routines as efficiently possible.

The identification and authentication of users can be a function of the client operating system, the network operating system or the operating system of the server(s). The client operating system can just protect the client and no more than that. Generally the network software and the operating system of the servers will be used to protect the network and the data on the servers. Recent developments, such as the Kerberos authentication system [4], offer opportunities for a useful identification and authentication policy.

Traditionally the authorization of users will be implemented by granting users the authority to execute certain application programs. Consequently users should only be allowed to insert or update the data in the database by using these application programs. In this way it can be arranged that the validation routines in the application programs will be executed before the data in the database will be modified.

A usual division of the application controls is as follows [5]:

- Source data controls
- Input validation routines
- On-line data entry controls
- Data processing and file maintenance controls
- Output controls

Examples of source data controls are:

- Check-digit verification of account numbers and other code numbers entered into the system.
- Sequence tests to check the right numerical or alphabetical sequence of input data.
- Manual reconciliation of batch totals.

Examples of input validation routines are:

- Validity checks, which compare customer numbers or other codes with the data stored in the database.
- Limit checks to establish whether entered amounts do not exceed a predetermined upper or lower limit, like a test of the order amount with a credit limit.
- Reasonableness tests to determine whether the input data is logically acceptable, e.g. a test whether the size of ordered quantities is reasonable.

Examples of on-line data entry controls are:

- Prompting to enter data into fields on the screen.
- Completeness checks on the data to be entered for a transaction.

- Closed-loop verification, such as displaying the name of the client according to the client table when a client number has been entered.

In a client/server environment these application controls can be implemented either on the client level or on the server level. The controls relevant to the corporate data should be executed on the database level, that is to say as close to the database as possible. There are no safeguards that the application controls on the client level will always be executed. As a matter of fact the function of the controls on the client is just to support the users. One could call them: 'self-controls' for the user. They help the user at the data input stage, but they are hardly useful for the control of the integrity of the corporate data.

2. INFORMATION FLOWS IN A COMPANY

2.1 An information model of trading companies

Business information systems are supposed to give information about business processes. The management will need information about the successive stages in the business processes, especially for the control of these processes. This information should be consistent and auditable, that is to say there should be an audit trail into the underlying detailed data.

The model shown in figure 2 gives an overview of the information and physical flows in a trading company regarding the primary business processes. The financial information flow represents changes in the financial relations with vendors, customers and financial institutions, including banks. The physical flow (value chain) represents the flow of goods and services received from the suppliers and (generally after some kind of transformation) delivered to customers. The logistical information flow steers the physical flow by means of purchase orders, sales orders, shipment orders and so on.

This model is meant to show how the information and physical flows should be measured and recorded in the database. The flows can be split up into stages and each stage should be recorded by a dedicated application program and recorded in a dedicated table of the database. This means the basic design of the database should reflect the information flows and physical flows in the organization.

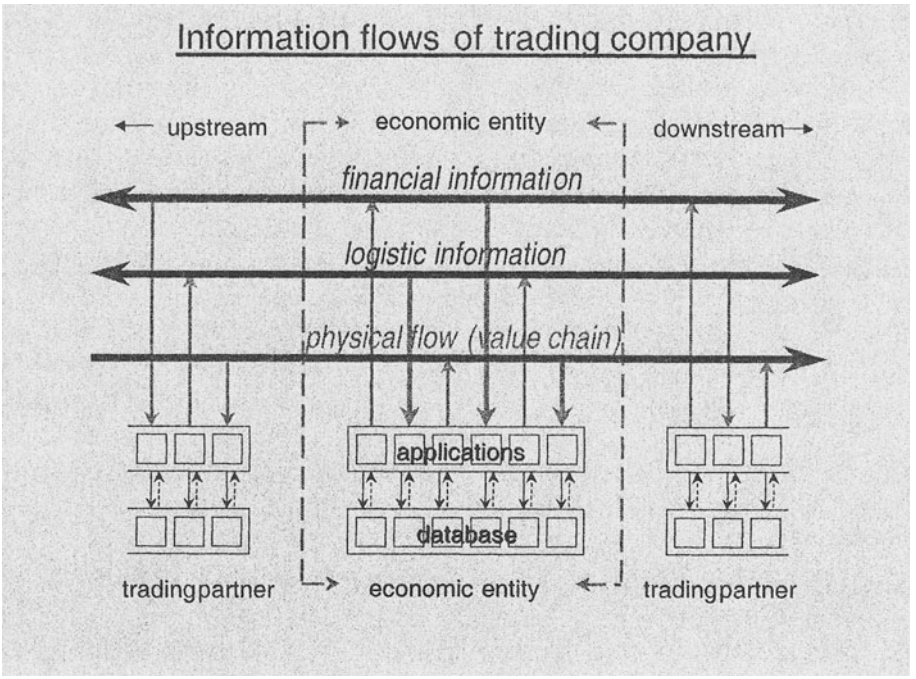

Figure 2. Information model of trading company

The model represented in figure 2 shows that an individual company is only a link in a chain of information flows and value streams from one company to another. The integration of the information systems with the systems of the preceding companies (upstream) or succeeding companies (downstream) in the chain is a challenge for many companies in the current business environment.

In order to retain an audit trail of transactions with trading partners it is essential to measure and record the information flows as soon as they cross the borders of the economic entity. E.g., incoming sales and shipping orders will have to be recorded as soon as these orders enter the network of the company. In the past the primary recording of transactions played an important role as a basis for the internal control and external audit. Nowadays the primary recording can be safeguarded by grasping all the transactions as soon as they are crossing the borders of the entity by means of data communication. The recording of these transactions asks for control and the files with the images of the transactions should be secured properly.

In order to obtain the necessary information about the value chain, the logistical flow and the financial flow, these flows should be measured at different stages. To secure the integrity of this information the (manual)

input of data has to be controlled and the consistency of the information about the business flows has to be reviewed.

2.2 Network of reconciliation totals

In figure 3 the interrelations which are recognizable in most trading companies are shown. These interrelations can be used to check the consistency of the financial representation of the value chain within the organization.

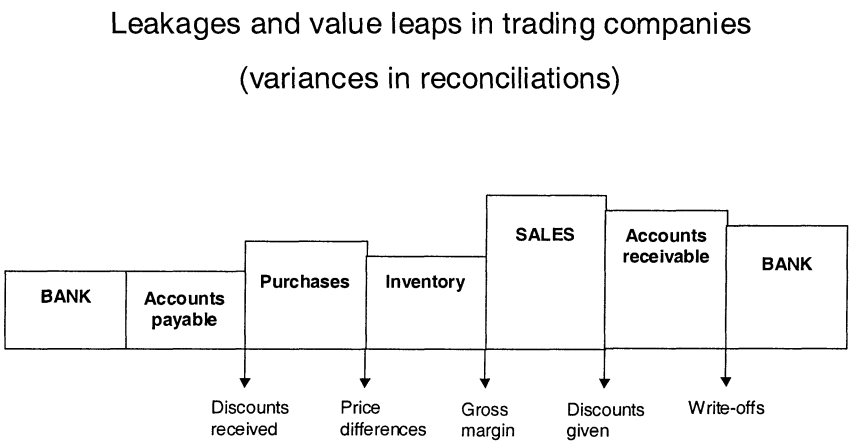

Figure 3. Value stream of trading companies

Note the following interrelations in this flow:

- payments to suppliers = debits on bank account
- recorded purchases = increase in accounts payable
- received goods = increase in inventories
- recorded sales = decrease in inventories
- recorded sales = increase in accounts receivable
- payments form customers = credits on bank account

However, these interrelations are never straightforward in practice. There are "leakages" in the value chain, like lost inventories, discounts given to costumers and non-payment by costumers, and "value leaps" like

discounts received from suppliers and the realized gross margin. These discrepancies in the value chain have to be reviewed and analyzed in order to find out their causes and to consider whether they are acceptable.

In service companies, aimed at delivering services charged per man-hour, the value stream could look like the representation in figure 4.

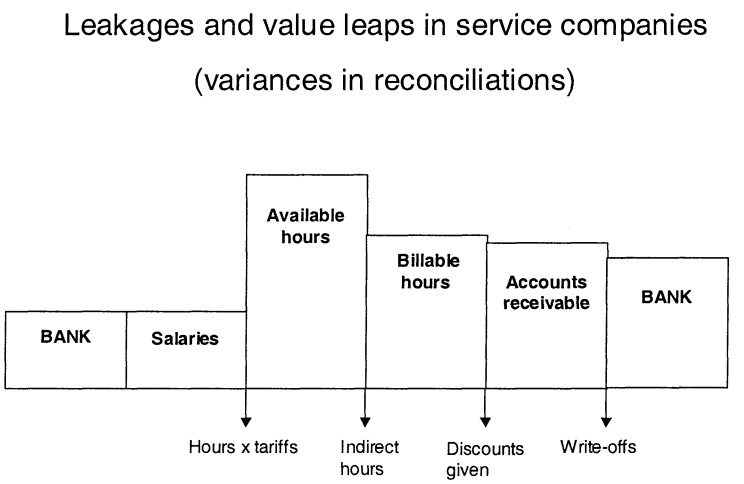

Figure 4. Value stream in service providing companies

The interrelations for this kind of companies can be:

- paid salaries = debits on bank account
- paid hours x rate = available man hours
- available man hours = billable hours + indirect hours
- billable hours = invoiced hours = increase in accounts receivable
- payments form customers = credits on bank account

These interrelations will be straightforward neither. The "leakages" in figure 4 cover discounts, non-billable hours and write-offs. The "value leaps" will cover gross margins, rating hours and so on. The information system should present enough information to analyze these leakages and value leaps. E.g., it must be possible to view the gross margin from different viewpoints, such as the gross margin per product group, the margin per customer group, region, sales person, branch, etc. The leakage of indirect

hours can be analyzed by presenting the leakage per category hours, per branch, level of employee, etc.

A check on consistency can also be a comparison with available standards such as budgets, production standards, sales targets, budgeted gross margin and so on. A discrepancy between budgeted margins and real margins is a form of inconsistency and should be analyzed and elucidated.
The checks on consistency of the distinguished flows should be primarily carried out as a standard function of the information system. The analytical information regarding the discrepancies should be presented to the controller or another functionary with controlling responsibilities, who has to investigate and explain the discrepancies. The management decides whether the discrepancies are acceptable.

In my thesis [6] I tried to combine systems theory with a reconciliation approach, which is rather common in Dutch auditing theory and basically described in a paper of the Dutch Institute of Certified Accountants [7] (the "network of reconciliation totals"[1]. Systems theory tells us that there are relations between input and output. These relations can be simple (e.g. in a trading company, as described above) or very complex (e.g. in some complicated industrial processes). In a industrial environment, the information systems will typically record usage of raw materials or components as the input to production processes and the delivery of parts or finished products as the output of the production processes. There must be a relationship between input and output. Generally this relationship can be described in the form of an arithmetic formula. From this arithmetic formula the standard usage of raw material and components can be derived.

To control these kinds of production processes, information must be available about the usage of raw materials or components and about the production of parts or finished goods. This information will typically be presented in the form of totals per production cycle or period. In order to judge the validity of these control totals, normative information should be presented as well. When there is a difference between the normative information (e.g. the standard usage of raw material) and the information about the real results, it should be possible to investigate the underlying basic data. As a consequence there is a need for storing data about the inputs and outputs of the production processes in dedicated tables of the database, in other words: the inputs and outputs form separate entities in the data model.

[1] see also [8], page 37-41

3. THE DATA MODEL

3.1 Modelling information flows

This paper stresses the importance of database design (in fact the development of the data model) for effective and reliable information regarding business processes. In consequence it is essential that at least controllers and accountants are involved in the development of the data model. A defective or incomplete data model can have far reaching consequences for management information and accounting information. Romney and Steinbart [5] put it this way: "Accountants can and should be involved in all stages of the database design process, although the level of their involvement in each stage is likely to vary. In the requirements analysis and design stages, accountants participate in identifying user information needs, developing the logical schemas and designing the data dictionary, and specifying controls."

The data model should reflect the successive stages of the information and value flows in order to record the business processes in an adequate way. Figure 5 represents a simple data model for a trading company.

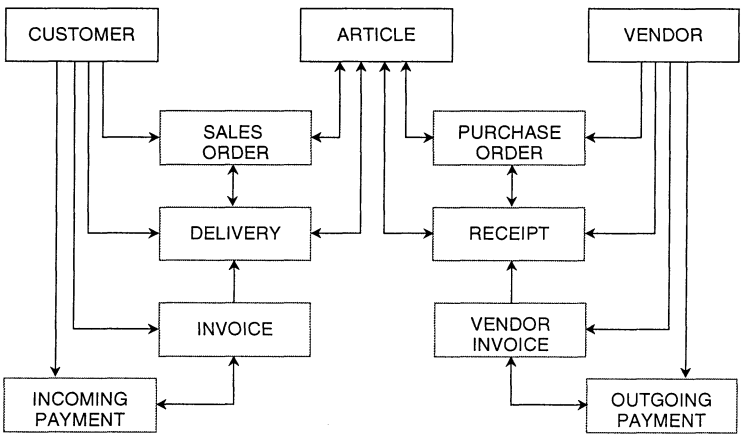

Figure 5. Simple data model for trading company

In this data model there are entities which are visible in the real world, such as "customer", "article" and "vendor". Besides these there are entities which reflect an action or transaction, such as "order", "delivery", "invoice", "payment". These entities are not tangible in the real world (apart from documents recording these transactions). They resemble the "events" of the REA-model of McCarthy [9], the model that makes a distinction between Resources, Events and Agents as entities in the data model.

For the control of business processes the management will need accumulated information about the "events", being the successive stages in the transaction flows. There must be an opportunity to analyze this accumulated information in case reconciliation of control totals will show differences or inconsistencies. Therefore the information system will need "drill down" facilities, that is to say facilities to recover the basic elements of accumulated information.

3.2 Example of design flaw in data model

In this paragraph an analysis will be made of a data model presented in a textbook for graduate students [10] with the object to illustrate the need of a good database design.

The following objects (entities) for a library system were identified:

- Library user (client)
- Library item (book or paper)
- Library staff member

Library users are recorded in the user database before they can borrow library items. Library items are recorded in the library items database. Library users are allowed to borrow library items. Library items are checked in and out by the library staff.

These notions can be described in the (simplified) data model, shown in figure 7.

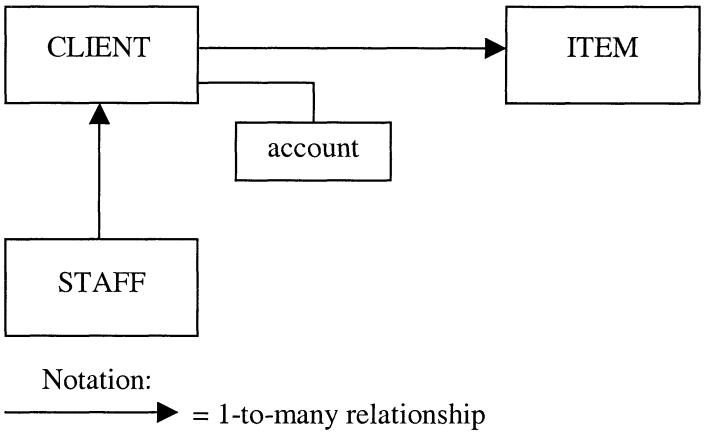

Figure 6. Simplified data model library system

In this data model three separate entities are defined: CLIENT (Library user), ITEM (Library item) and STAFF (Library staff member). Only the relationships relevant for this example have been drawn. A staff member can serve several clients. A client can only be served by one staff member. A client can borrow several library items. A library item can only be borrowed by one client (at the same time). The rectangular mentioning "account" should not be regarded as an entity, but as an attribute of the entity client. This data model could be used as the basis for a library system. The entities chosen are entities that are visible in the real world, resources and agents according to the REA-model of McCarthy. Apparently in this case there are no events defined as entity.

The data model presented is like a snapshot, a picture of a situation at just one moment. As soon as the book borrowed by the client has been returned the relationship between the client and the item should be deleted; otherwise it will not be possible to lend the same book to another client. The historical information has been discarded at that moment. However, it can be expected that later on the management will need management information, e.g. how many books have been borrowed over a period of time, or how many clients have been served by a staff member. This information cannot be derived easily from an information system built up according to this data model. Maybe (by chance) it can be derived from the aggregation of accounts, though these attributes of the client are not meant to give that kind of information. They are only meant to control the borrowing process. If the decision were made to define an entity for the event "transaction", the

problems mentioned above could be solved easily. In that case the data model could be drawn as shown in figure 8.

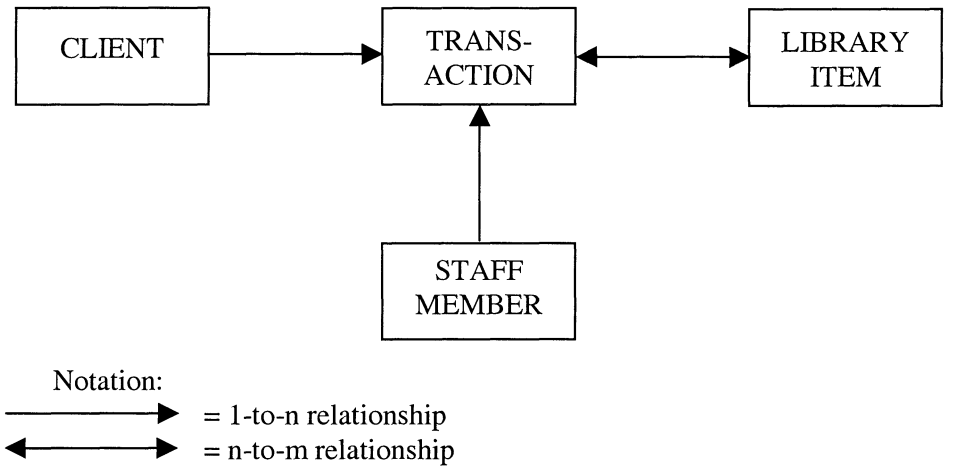

Figure 7. Adapted data model for library system

A client can be engaged in more than one borrowing transaction. A borrowing transaction can conceive one or more library items. A library item can be included in zero, one or more borrowing transactions. A staff member will handle more than one borrowing transaction. This adapted model can be the basis for recording historic transactions in order to present control information over time and to offer the possibility of drilling down into detail records of borrowing transactions.

This example illustrates the need for the involvement of controllers and accountants in the data-modeling phase of systems development. In this phase important decisions are made for the information system to be built, decisions that can have far reaching consequences for the information to be gathered from the system and for auditing possibilities as well.

3.3 Separation of duties

Another reason for identifying the successive stages in a transaction flow could be the wish to implement a suitable form of separation of duties. Separation of duties forms an important measure of internal control, especially with a view to the reliability of information about business processes.

Traditionally the following separation of duties has been prescribed [11]:

- Authorization of transactions

- Execution of transactions
- Custody of assets
- Recording of transactions and assets

In addition to these "classic" separation of duties, independent performance of various phases of accounting, such as separated recording cash receipts and cash disbursements, has been recommended [12].

As Romney and Steinbart [5] point out, the REA-model (the data model that distinguishes resources, events and agents) can be useful in evaluating the extent to which incompatible duties are segregated since this model shows which internal agents participate in each event.

Clark and Wilson proposed to implement the desired separation of duties in a IT-environment by means of authorization controls [13]. This includes assigning application programs to users. These application programs should be what they call "well-formed transactions". The data in the database should be changed exclusively by means of these application programs.

One of the problems of this approach is the integrity of the application programs. In practice the well-formed transactions are often not as "well formed" as one would desire. Application programs are prone to change and error, and therefore less reliable in many instances. Besides, application programs can be manipulated or circumvented by the user, especially when they are installed on client systems.

A more reliable solution can be derived by implementing the authorization controls on the database level, that is to say by assigning users or groups of users the authorization to insert, update or delete data in tables (or in columns in tables). Such a solution is particularly suitable for audit purposes. In case the authorization controls are implemented on the level of applications, a good knowledge of the functionality of the applications is necessary to be able to evaluate authorizations and consequently separation of duties. Usually it is very hard to get insight in the functionality of application programs, due to a lack of good and actual documentation.

The authorizations on a database level, however, are typically straightforward and easy to analyze. Leenaars stated in his thesis [14] that – in a highly computerized environment - the "recording of transactions and assets" as a separate function will be replaced by automatic recording within the information system. In such an environment many business transactions (such as the execution of sales orders) can only be carried out by using the computer system, i.e. by entering the order into the computer system. The information system will check whether the order remains within the credit limit of the client and whether the level of stock is sufficient. After these checks the order will be displayed in the warehouse, where it can be

compiled, packed and shipped, and subsequently be entered into the system as delivered.

If these kinds of procedures are followed and it is not possible to circumvent these procedures (e.g. by passing sales orders to the warehouse without using the computer system), then the completeness and accuracy of the registration of the sales transactions is sufficiently safeguarded. In this case the recording function has been transferred from a manual function to an automatic procedure.

Besides the example given above illustrates that even the function "authorization of transactions" can be replaced by automatic procedures, such as the checks against the credit limit and the stock quantity. The result will be that there is only a separation of duties between the "execution of transactions" and "custody of assets". In many companies the custody of assets is not an issue, especially in the service related industry. The conclusion must be that in many cases the traditional model for separation of duties, advocated by Romney and Steinbart [5], Vaassen [8] Gray and Manson [11], Horngren [12] and many others, has become outdated.

The aim of the internal control measure "separation of duties" was primarily to ensure the reliability of information about business events. When more people were involved in a business event and each of these persons kept a separate registration of the event, then the reliability of the information about the event could be checked afterwards by comparing the separate registrations.

However, when transactions are automatically registered, there is no need for separate registrations of the same event anymore. What is needed is a system that safeguards the reliability of the representation of the real world transactions in the information systems. A solution could be to observe (and record) the transaction flow at different stages and to compare the results of these recordings.

There could, e.g., be a separation of duties related to:

- the recording of a sales order
- the recording of the shipment
- the recording of the incoming payment

These recordings should be consistent with each other within the constraints of the "reconciliation network" described before. This approach can be implemented by defining different entities and different tables of the databases for successive stages in the transactions. Each table can then be allocated to one user or a group of users in order to realize what might be called "separation of phases"[2].

[2] Term introduced by Hartman [15].

4. DATABASE CONTROLS

4.1 Control facilities of a modern DBMS

Traditionally application controls are implemented by means of application programs. It has been argued before that application controls on the client level are less reliable, since the user may influence or circumvent the applications on the client. But even the application programs on the server level might not be very reliable, since they are prone to change and error too. Many authors (e.g. [16] ,[17], [18]) point out, that data structures tend to be more stable than procedures. That is an important argument to investigate whether the application controls can be implemented on the database level. In this section I will investigate the facilities of a modern database management system from that point of view. These facilities[3] include:

- referential integrity
- the data dictionary
- integrity constraints
- stored procedures
- database triggers

These facilities offer interesting opportunities to transfer (part of) the integrity rules from the application programs to the database environment.

Referential integrity

Modern relational database systems offer facilities for controlling referential integrity. A foreign key in a table should always point at a primary (or unique) key in another table. E.g., when an order is recorded in the order table, the customer will be identified by a foreign key referring to a primary key in the customer table. The foreign key cannot be entered into the order table when there is no corresponding primary key in the costumer table. Besides the primary key cannot be removed from the customer table, when there is still a reference to this key. This is called referential integrity, an important concept regarding accuracy of data.

Data dictionary

The data dictionary will automatically be created when a database is created. When the structure of the database is altered, the data dictionary will be automatically updated.

[3] Terminology from "Database Security in Oracle*8i*™, An Oracle Technical White Paper, November 1999".

The data dictionary describes the logical and physical structure of the database. Besides the data dictionary can present the following types of information:

- definitions of all schema objects in the database (tables, views, indexes, synonyms, procedures, triggers, and so on)
- default values and integrity constraints for columns in the tables
- the users of the database with their privileges and roles
- historical information, e.g. updates of objects by users

The data dictionary is particularly interesting from the audit point of view, presenting up to date and reliable information about application and authorization controls.

Integrity constraints

The integrity constraints declare rules for the values stored in the columns of a table. If an integrity constraint is declared for a table and some existing data in the table does not satisfy the constraint, the constraint cannot be enforced. Once a constraint is defined, all the updates in the column will be checked against the integrity constraint. If a DML statement violates any constraint, the statement is rolled back and an error message is returned.

Examples of integrity constraints are:

NOT NULL	disallows nulls in a column of a table.
UNIQUE	disallows duplicate values in a column or set of columns.
PRIMARY KEY	disallows duplicate values and nulls in a column or set of columns.
FOREIGN KEY	requires that each value in a column or set of columns match a value in a related table's UNIQUE or PRIMARY KEY
CHECK	checks on a logical expression.

Stored procedures

Stored procedures are small programs, written in languages such as PL/SQL or Java, that can be used to limit the way a user can insert, update or delete the data in the database tables. The validation rules in (front-end) applications can be bypassed by the user if the user has direct privileges in the database. Stored procedures, however, can not be bypassed during modification of data.

A procedure can be defined that performs a specific business function. The user can be given authority to execute this procedure without being granted any access to the objects and operations that the stored procedure

uses. This prevents users from modifying data outside the context of the predefined procedure.

Database triggers

Database triggers are small programs like stored procedures. While stored procedures are explicitly executed by users, database triggers are automatically executed or "fired" in the case of predefined events. A trigger can be executed either before or after an insert, update, or delete. When such an operation is performed on that table, the trigger automatically fires. Four types of triggers can be defined on a table: BEFORE statement, BEFORE row, AFTER statement, and AFTER row.

Database triggers offer the opportunity to record operations by users. E.g., a BEFORE UPDATE trigger can be defined on a table that automatically records the existing values in the table before they are updated by the user. In that way both the old and new values can be recorded in the rows updated.

5. CONCLUSIONS

Good business information is information that gives a true and fair view of the real world activities. In order to get good information, the real world should be observed and recorded at different stages of the information and value flows in a company. The data model that forms the basis of an business information system should reflect these different stages. Such a data model (a type of REA-model) offers opportunities for effective and reliable information about the business processes. This information is needed for management and control purposes. Besides, the auditability of the information will be improved by a data model reflecting the stages in the business processes.

The consistency of information about business processes requires reconciliation of control totals. Variances and inconsistencies between successive control totals, or between control totals and related standards, should be presented to the controller or another user with a controlling task for further analysis. This analysis can be supported by the information system in the form of analytical information about variances and by offering drill down facilities. These facilities require the retention of the basic data, which must be foreseen in the data model.

The implementation of separation of duties requires authorization controls. Traditional separation of duties tend to be replaced by a separation of duties following the stages of the business processes. When the data model reflects these stages, the separation of duties can be implemented

using authorizations for the different tables in the database. Separation of duties implemented at the database level is more reliable and auditable than separation of duties implemented at the application level.

Validation controls are important to check the accuracy of input, especially in case of manual input. Validation routines can be implemented in application software. In client/server systems the front-end applications are difficult to control. The validation routines at that level can be considered to be more important for the user himself (self-control) than for the corporate data. Truly reliable validation routines should be implemented at the server level and preferably by means of the DBMS. With referential integrity, integrity constraints, stored procedures and database triggers more stable and reliable validation routines can be arranged. The data dictionary facilitates the insight into the implemented controls.

REFERENCES

[1] Woolfe, R., *Managing the Move to Client-Server*, Wentworth Research Program, 1995.

[2] Noordam, P.G. and A. van der Vlist, *Trends in Informatietechnologie*, Deventer, NL, 1995.

[3] Healy, M., *Client/server-omgevingen beveiligen*, in: Computable, September 8th, 1995, page 29.

[4] Schiller, J.I., *Secure Distributing Computing*, in: Scientific American, November 1994, pages 54-58.

[5] Romney, M.B. and Steinbart, P.J., *Accounting Information Systems*, 8th edition, Upper Saddle River, NJ, 2000.

[6] Koning, W.F. de, *Informatie voor de beheersing van bedrijfsprocessen*, thesis, Rotterdam, 1997.

[7] NIVRA, NIVRA-geschrift nr 13, *Automatisering en controle*, Amsterdam, 1975.

[8] Vaassen, E.H.J., *Accounting Information Systems*, Chichester, UK, 2002.

[9] McCarthy, W.E., *The REA Accounting-model*, in: The Accounting Review, July 1982.

[10] Kotonya, G. en Sommerville, I., *Requirements Engineering*, Chichester, UK, 1998.

[11] Gray, I. and S. Manson, *The Audit Process*, London, 1989.

[12] Horngren, C.T., W.T. Harrison and M.A. Robinson, *Accounting*, 3rd edition, Englewood Cliffs, NJ, 1996.

[13] Clark, D.D. and D.R. Wilson, *A Comparison of Commercial and Military Security Policies*, Proceedings of the 1987 IEEE Symposium on Security and Privacy, page 184-194.

[14] Leenaars, J.J.A., *Functiescheidingen in hooggeautomatiseerde omgevingen*, thesis, Alphen aan den Rijn, NL, 1993.

[15] Hartman, W., *Organisatie van de Informatieverzorging*, The Hague, 1995.

[16] Veldhuizen, E., Ham, H.W.F. van den, Keijzer, C., Kielen, E.M. and Koning, W.F. de, *Rapport van de Werkgroep Informatietechnologie en Interne Controle*, Limperg Instituut, Amsterdam, 1994.

[17] Yourdon, E., Whitehead, K., Thomann, J., Oppel, K. and Nevermann, P., *Mainstream Objects: An Analysis and Design Approach for Businesses*, Upper Saddle River, NJ, 1995

[18] Curits, G. and Cobham, D., *Business Information Systems*, 4th edition, Harlow, UK, 2002