December 2000

Linear Algebra and Sets Theory (LAST)-based Formal Modeling for IS Client-Specification

Luis Gonzalez
Universidad de la Rioja

Cesar Ruiz
Universidad Complutense de Madrid

Follow this and additional works at: http://aisel.aisnet.org/pacis2000

Recommended Citation
http://aisel.aisnet.org/pacis2000/60

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 2000 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Linear Algebra and Sets Theory (LAST)-based 
Formal Modeling for IS Client-specification

Luis Gonzalez, Cesar Ruiz
Department of Business and Economics of the Universidad de La Rioja,
Department of Mathematical Analysis of the Universidad Complutense de Madrid

Acknowledgments
This article has benefited from the valuable suggestions of Jesus Almendros (Universidad de Almería) and the kind assistance of Pilar Vargas (Universidad de La Rioja)

Abstract
It seems clear that there is plenty of room for improvement in the area of IS client-developer working relationship. On the client side, it is general to find professionals with a background in the fields of Management and Accounting (M&A). Two disciplines involving the development of rich and precise abstract models of their domain of interest. Consequently, this paper assumes that it is desirable to establish a (suitable) common language for communication between the M&A and IS communities. To that end, this paper describes a LAST-based formal method for IS client-specification. The use of standard mathematical language guarantees precision, accessibility, and comprehensibility by both IS and Management specialists. The resulting client-specification may be translated by the IS designer into a system-specification proper (that can be verified and implemented). Another possible use of LAST-based modeling, consistent with the stated objective and tested by the authors over the course of several years, is as an educational tool. For illustration purposes, a Financial Accounting Information Subsystem is chosen because it is highly standardized and of widespread use.

Keywords: IS development; Specification method(s); Formal method(s).

“The gross division of system development into the two stages of specification and implementation is widely accepted. We first determine the ‘what’ and then determine the ‘how’.” (Jackson 1983, 25).

“There appears to be general consensus within the Information Systems literature that formal specification of software systems is an inappropriate response to the perceived general failure of information systems to meet user requirements. Such views would seem to be based primarily on the difficulty of constructing formal specifications -- and on the difficulty of understanding such specifications once constructed.” (Swatman, 1992)

1. Introduction

Despite the continuous improvement in both IS underlying technologies and system-development methods, IS-project failures still abound, which have a negative impact on organizational performance. Recent studies (Preiser-Houy, 1999) point to insufficient quality in the client-developer relationship as a major cause of this and the “culture-gap” between IT and business people has been identified as a major cause of system development failures (Taylor Cummings, 1998). Taking into account that the client side usually has an M&A background, implying
modeling capabilities which are not sufficiently appropriated during IS-development, this paper contends that improving the client-developer communication will lead to a dramatic improvement in the requirement analysis. This may be achieved through a pre-processing on the client side, using a formal (specification) method that would use a standard language, common to the different project-stakeholders.

Formal methods are mathematically-based languages, techniques, and tools for specifying and verifying computer systems (Clarke and Wing 1996). There are two ways in which a formal methods user may cross the boundary between the mathematical world and the real world (Wing, 1990, 19-20). The first one takes place when he or she codifies informal requirements stated by the customer (system-user):

\[ \text{Informal requirements} \rightarrow \text{codifying} \rightarrow \text{Formal specification} \]

When mapping from the real world to an abstract representation of it, a second boundary is crossed. The formal method then serves the purpose of encoding the resulting abstract model:

\[ \text{Real world} \rightarrow \text{abstraction} \rightarrow \text{Abstract model} \rightarrow \text{encoding} \rightarrow \text{Formal specification} \]

Given the fact that Management and Accounting are fairly good at modeling the reality of business organizations, it seems fair to say that, if provided with the necessary tools, specialists from these two fields may very well deliver an accurate abstract model of the desired IS. LAST-expressed specifications would have the advantages usually attributed to formal methods: (1) designs thus specified are completely independent of the supporting hardware and software, therefore they are more likely to survive technological change (Cohen et al. 1986, 95); (2) because of their mathematical basis, formal specifications are more precise and usually more concise than informal ones (Wing 1990, 10); and (3) an abstract specification would favor gains in cost and efficiency in the subsequent product design and implementation (Hoare 1990); it could serve as a sort of contract between client, specifier and implementer (Wing 1990, 12).

A method allowing for accurate specification on the client side is of utmost interest for several reasons. For clients because they have full control of the system’s features that are relevant to them (what the system can do and how you get it to do it) and, at the same time, they are released from the specifics of system specification (how the system does what it does). On their side, system developers will doubtlessly benefit from a clear-cut order. That should lead, in turn, to more robust (and better-documented) systems.

The usual formal methods, like Z or VDM, are arguably too complex for this objective (and, possibly, for the specification of IS itself). In addition, although they share a common philosophy and a common method (Hoare 1990, preface), they seem to be growing apart as regards style and notation. Together with the fact that they use a newly coined (and thus, not established), ad-hoc (and therefore, non-standard) notation, this implies that they lack both the time-resilience, and the worldwide acceptance, that mainstream Mathematics enjoys. On these grounds, the technique herein developed uses standard mathematical language, from the fields of the Sets Theory and Linear Algebra, a worldwide standard notation that is common to the M&A and IS communities.

1.1 LAST-based Specification

The technique that will be formalized herein is supported by just a few mathematical tools:


3. Regular mathematical notation: conjunction (and), disjunction (or), conditional (if, also noted by \( \Rightarrow \)), and biconditional (if and only if, noted with \( \iff \)), quantifiers like \( \forall \) and \( \exists \), order symbols (\(<\), less than, \(>\) greater than, etc), intervals \((a, b]\), the set of elements between \(a\) and \(b\), \(x \in \{a, b\} \iff a \leq x \leq b\), and simple operations with numbers: addition, subtraction, multiplication and division.

Item 1 shall serve the purposes of data storage and retrieval, while items 2 and 3 will be used to represent data-manipulation.

Data are represented by vectors (grouped in universal sets), and the use of coordinate functions allows for retrieval of either whole vectors, or any “string” of vector-entries. This makes it possible to specify any possible operation with the stored data. Note that there never is, in any of the vectors that represent different types of data, a vector-entry that acts as a “key field”. A relational DBMS may be adequate for the implementation of the specification herein developed, but the latter is not a relational model. LAST-modeling differs from Relational Algebra (Codd, 1990) in several aspects: (1) it does not use keys, nor tables (notice that vectors used herein have variable lengths); and (2) it does not “coin” special non-standard mathematical concepts (like field, attribute, or attribute value).

1.2 Chosen Specificand

The specification of a fully-fledged IS would require a very lengthy discussion. Therefore, a basic Financial Accounting Information Subsystem (FAIS) has been chosen for illustration purposes. This choice is motivated by two considerations: (1) the convenience of focusing on a subsystem that is highly standardized, and that continues to be the most commonly used in practice (Seddon et al. 1992, 94); and (2) the other subsystems in the IS are much easier to specify, once an abstract model has been developed. Besides restricting the scope of the specification, it is necessary to choose a particular accounting abstract model that may be encoded into a formal specification. This paper chooses to specify a FAIS that relies on the standard accounting data manipulation formalism (DMF). This has, again, the advantage of its widespread use, plus the added one of being a formalism, not an abstract model, and thus easier to specify in very precise terms. It should nonetheless be clear that the method that will be described can be applied to the specification of any component of the FAIS not dealt with hereunder, of the other subsystems in the IS, and of any components providing integration between subsystems.

1.3 Resulting Specification

In addition to the mathematical tools mentioned above, two specific tools will be necessary: a basic data structure and a signs convention that substitutes for the Debit and Credit convention for data processing within the system. Note that the data structure will be different for the other subsystems of the IS and that the sign convention is exclusive to the FAIS. These specific tools are described in Section 2. Section 3 describes the input, storage and retrieval of data. The
following sections develop the formal specification of operations leading to different types of reports. Reports specified in Sections 4 and 5 correspond to what is commonly referred to as online inquiry (journals, statements and trial balances); these are usually highly standardized and may well be predefined in the specification. What accountants refer to as (accounting or financial) reports are called here, for the sake of identification, highly aggregated reports (Section 6). These are not so precisely standardized and in any case, report design seems to be another common feature of systems (Seddon et al. 1992, 95). A specification procedure, rather than an accurate specific design, therefore seems more appropriate.

2. Specific Tools

Before describing the basic components of the FAIS that will be specified (henceforth referred to as “the system”), it is necessary to establish two particular features of it. These can be modified to accommodate the user’s requirements.

1. Transaction data input (journalizing) is carried out through one single general journal interface (Figure 4). This means that there are no special journals, nor subsidiary ledgers, and that all entries, irrespective of their financial nature, have the same configuration and are jointly processed.

2. Posting has been eliminated to avoid the storage of redundant data. For the same reason, the ledger is broken up: the initial account balance has to be retrieved from one source of data, and the account movements from another one.

These features will be built into the LAST-expressed model in the following sections.

2.1 Data Structure

Figure 1 gives the basic data structure implicit in the system:
It has two important features: (1) aggregation always goes from the bottom to the top of the diagram; and (2) it involves two types of relationships: constant and variable. Constant relations cannot be changed by the user once they have been established. Variable relations will be defined by the user, who may, at any time, redefine them.

As regards the components of this data structure, the following rule applies: the user must establish the components in the bottom half of the structure in the reverse order to the aggregation sequence: first the accounts and then the entries. Definition of items and highly aggregated reports is addressed in Section 6.

2.2 Transformation of the Debit and Credit Convention into a Signs Convention

Accounting magnitudes have in common a specific characteristic: they all are bidimensional magnitudes. For each and every one of them, definitions must be made for both the amount $q$, expressed in currency units, and what could be called the “direction”, $\delta$, which can only have one of two alternative values (Debit or Credit):

$$
(q, \delta) \\
q \geq 0 \\
\delta = \{Db, Cr\}
$$

except when the amount is nil (a zero balance), in which case there is no direction: $q = 0 \Rightarrow \exists \delta$.

It is possible to formulate the operation of accounting addition as follows. Let $(q_1, \delta_1), (q_2, \delta_2), ..., (q_n, \delta_n)$ be the $n$ accounting magnitudes to be added up, and $(Q, \Delta)$, the result of the operation. The two resulting dimensions will be:

$$Q = \left\{ \begin{array}{ll}
\text{SUM} & \\
\text{SUM} < 0, & \text{Db} \\
\text{SUM} = 0, & \text{Cr} \\
\text{SUM} > 0, & \text{Db} \\
\end{array} \right.
$$

$$\Delta = \left\{ \begin{array}{ll}
\text{SUM} < 0, & \text{Cr} \\
\text{SUM} = 0, & \exists \Delta \\
\text{SUM} > 0, & \text{Db} \\
\end{array} \right.
$$

where: $SUM = \sum_{i=1}^{n} q_i \{ \text{if } \delta_i = \text{Cr}, \text{then } -1, \text{otherwise } 1 \}$

For the preparation of more complex financial statements, e.g. balance sheet and income statement, a certain “rule of signs” applies, as illustrated by the table in Figure 2.

<table>
<thead>
<tr>
<th>Reports Components</th>
<th>Sign for aggregation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assets on the balance sheet</td>
<td>Debit balances +</td>
</tr>
<tr>
<td></td>
<td>Credit balances -</td>
</tr>
<tr>
<td>Equity and liabilities on the balance sheet</td>
<td>Debit balances -</td>
</tr>
<tr>
<td></td>
<td>Credit balances +</td>
</tr>
<tr>
<td>Income statement</td>
<td>Debit balances -</td>
</tr>
<tr>
<td></td>
<td>Credit balances +</td>
</tr>
</tbody>
</table>

Figure 2: Highly Aggregated Reports Components
It is apparent that for the data processing, it is much easier to handle only one (conventional) rule of signs. The idea is quite simple. It works by converting debit magnitudes (debits and debit balances) into positive values, and credit magnitudes (credits and credit balances) into negative values. The system will thus handle values that may be added algebraically, making aggregation easier. For the communication between the system and the user, the opposite conversion is sometimes required. For the issuance (by the system) of more aggregated statements (e.g. balance sheet and income statement) it is enough to specify whether it is necessary to change signs or not. As an instance of this, for the reports listed in the table in Figure 2, changing signs is required for the following components: (1) owner’s equity and liabilities on the balance sheet and (2) income statement account (complete).

3. Data Input, Storage and Retrieval

3.1 Transaction Data (Accounting Entries) Input

Figure 3 represents a relatively complex interface for the input of entries.

The data input may be either a semiautomatic, or an automatic process⁶. As shown in Figure 3, the fields: type, branch, title of account and project are always preceded by their corresponding code. For entering transaction data manually, only codes need be typed in⁷. The precise contents of each field will now be described. To do this, entries will be treated as being formed by two basic building blocks: the heading (first line in Figure 3), and the entry-lines (the remainder).

**Date**: to be assigned by the user, it may or may not be coincident with the date of recording. **Number**: automatically assigned by the system⁸. **Reference**: identification of the supporting document. **Type of transaction**: brief description of the one being recorded. **Branch**: in companies with a decentralized information system, this field would identify the relevant business unit for the transaction being registered. These five values will be entered only once for each entry. They make up what will henceforth be called heading of the entry or just heading. Below the heading, there will be as many lines as there are accounts necessary for the recording of the transaction (no less than two, as a consequence of the double entry restriction: total debits must always equal total credits, therefore two accounts, at the least, are needed to make an entry).

**(Entry) line number**: within each entry, lines are correlative ly numbered. This number will also be assigned automatically by the system. **Account**: debited or credited account (an account represents a specific category within either assets, liabilities, owner-equity, income or expenses). **Amount**: monetary value for the entry-line. **Direction**: Debit or Credit. **Maturity (date of)**: to be
filled in only when the entry-line refers to receivables or payables. Project: this field allows for
the allocation of revenue and expenses to jobs, orders, etc.

The data configuration just described is unnecessarily complex for illustration purposes. Instead, the simplified general journal interface in Figure 4 will be used.

The reduction in the number of variables does not diminish the validity of the specification. Although the variety of reports that may be specified is thus lessened, a minimum effort allows for the introduction of new variables, and thereafter for the specification of new reports.

<table>
<thead>
<tr>
<th>Entry Date</th>
<th>Number</th>
<th>Type of transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line</th>
<th>Account</th>
<th>Amount</th>
<th>Db/Cr</th>
</tr>
</thead>
</table>

Figure 4: Simplified General Journal Interface

Another regular feature of actual FAIS (see note 7) that has been deleted is the assignment of a pair (code, title) to every category of each set of classes. It is arguable whether this level of specification requires such technical detail, it would unnecessarily complicate the specification herein developed (which, after all, is just an example of the application of a method) and, in any case, it is quite simple to modify the specification to include this feature.

3.2 Data Storage

This process may be broken down into three stages: conversion, verification and storage proper.

3.2.1 Conversion

The conversion is the process of substituting positive and negative values for debits and credits, respectively. Let \( q \) be the amount of the entry-line and \( \delta \) its direction (Db/Cr). The value of the line is a function of \( q \) and \( \delta \): \( \Phi(q, \delta) = q\Phi(\delta) \), where: \( \Phi: \{Db, Cr\} \rightarrow \{1, -1\} \), \( \Phi(Db) = 1 \) and \( \Phi(Cr) = -1 \).

3.2.2 Verification

This involves verifying that all the accounts included in the entry exist in the system and testing that the entry is balanced. If the entry includes an account not belonging to \( U_A \) (defined in 3.4 Definition of the Sets of Classes), or totals on the debit and credit sides are not equal, the system should reject the entry. This can be included in the specification by imposing the following two restrictions:

\[
\forall E \in U_E \quad \exists A(E) \in U_A \quad \text{and} \quad \sum_{r=1}^{\hat{R}(E)} \hat{V}_r(E) = 0
\]

where:

\( \hat{R}(E) = \text{number of lines in the entry} \)

\( \hat{V}_r(E) = \text{value of line } r \text{ of the entry} \)

\( U_E = \text{set of all the entries stored in the system} \)

\( U_A = \text{set of all the accounts} \)
3.2.3 Storage

Each entry will be represented as one element of the set, $U_E$, of all the entries in the system:

$$U_E = \{ E_i: i = 1, \ldots, n \} = \{ (D_i, N_i, R_i, A_{1,e}, V_{1,e}, A_{2,e}, V_{2,e}, \ldots, A_{r,e}, V_{r,e}, \ldots, A_{R,e}, V_{R,e}) \in \mathbb{R}^{4 + 2R_i} \}$$

where:
- **Heading**
  - $D_i$ = entry date
  - $N_i$ = entry number
  - $O_i$ = type of transaction
  - $R_i$ = number of lines making up the entry
- **Line $r$**
  - $A_{r,e}$ = account in line $r$
  - $V_{r,e}$ = value of line $r$

Every element in $U_E$ contains all the components of an entry and, for each entry, there is an element in $U_E$, whose minimum number of vector-entries is eight. The reason is that compliance with the double (accounting) entry restriction means that two is the minimum number of lines for an entry. On the other hand, there is no limit, in principle, to the number of lines.

3.3 Data Retrieval

3.3.1 Definition of Coordinate Function

A coordinate function $\tilde{x}_j$ acts on a vector $V$, returning the vector-entry $v_j$. That is:

$$\tilde{x}_j(V) = v_j \quad \text{where} \quad V = (v_1, v_2, \ldots, v_n) \in \mathbb{R}^n$$

3.3.2 Definition of Search Functions over Vector $E$

It is possible to define the following coordinate functions:

$$\forall E \in U_E$$

- **Heading**
  - $\tilde{D}(E) = e_1$ = entry date
  - $\tilde{N}(E) = e_2$ = entry number
  - $\tilde{O}(E) = e_3$ = type of transaction
  - $\tilde{R}(E) = e_4$ = number of lines
- **Line $r$**
  - $\tilde{A}_r(E) = e_{4 + 2r - 1}$ = account in line $r$
  - $\tilde{V}_r(E) = e_{4 + 2r}$ = value of line $r$
  - $1 \leq r \leq \tilde{R}(E)$

805
3.4 Definition of the Sets of Classes (Categories for Classification)

Some of the variables in one entry are meaningful for classification purposes, and some of them are not. For the configuration shown in Figure 4, transaction type and account belong to the first group. Therefore, two sets of classes must be defined:

1. Set of transaction-types: \( U_o = \{O_i; i=1,2,...,m\} \), where \( O_i \) stands for the transaction-type \( i \), and \( m \) is the number of different types defined by the user.

2. Set of accounts: \( U_A = \{A_j, V(A_j, \tau); j=1,2,...,n\} \), where \( A_j \) stands for account \( j \) and \( V(A_j, \tau) \) for its initial value (on date \( \tau \)), being \( n \) the number of different accounts defined by the user. The initial value will be zero when the account has just been opened, and will then be updated (when necessary, see under 3.6 Limits to Transaction Data Writing and Retrieval) with the following formula:

\[
V(A_j, \tau) = \sum_{\beta \in U \beta} \tilde{V}(E) \quad (\text{for } \beta \in A_j \cap K(\beta))
\]

Processes involving aggregation may be substantially simplified if the vector representing an account (any element of the set \( U_A \)) includes its values on certain dates. The criterion to select the dates is simple: frequency of use\(^{11}\).

3.5 Date Format and Chronological Order Arrangement of Entries

If the dates of the entries are stored in the format YYYY/MM/DD, then the order resulting from arranging entries according to their number is chronological. In addition, if entries are ordered within the matrix \( U_E \) according to their date and number, the following holds:

\[
\begin{align*}
\bar{D}(E_i) < \bar{D}(E_j) \\
\text{or} \\
\bar{D}(E_i) = \bar{D}(E_j) \quad \text{and} \quad \bar{N}(E_i) < \bar{N}(E_j)
\end{align*}
\]

A double criterion (based on date and number) is used for the ordering of entries for two reasons:

1. It allows the manipulation and ordering of successive (daily, monthly, or yearly) series of entries.
2. Because of the imperfect relation between entries’ dates and numbers, that may be postulated\(^{12}\):

\[
\forall E, E' \in U_E \quad \text{it does not necessarily hold that } \bar{N}(E) > \bar{N}(E') \rightarrow \bar{D}(E) > \bar{D}(E'), \quad \text{nor that } \bar{N}(E) < \bar{N}(E') \rightarrow \bar{D}(E) < \bar{D}(E')
\]

3.6 Limits to Transaction Data Writing and Retrieval

Figure 5 shows a time diagram with four relevant dates which bound three different periods of
time:

\[
\begin{array}{c}
\omega \\
\downarrow \\
\rho \\
\downarrow \\
\tau \\
\downarrow \\
\theta
\end{array}
\]

\begin{itemize}
  \item deleted transaction data time interval
  \item data entry time interval
  \item data retrieval time interval
  \item \(0 \leq \tau < \rho\)
\end{itemize}

\textbf{Figure 5: Entry Retrieval Time-interval}

Where: (1) \(O\) stands for the system inception date, (2) \(\tau\) denotes the latest date for which there is no transaction data available in the system (if the system’s database is perpetual, \(\tau\) will equal \(O\)) and (3) \(\rho\) and \(\omega\) stand for the lower and upper bounds of the (closed) time interval for which transaction data are allowed to be input into the system.

The system database will thus contain transaction data only for the following period of time:

\[
[O, \omega] \text{ if } \tau = O \quad \text{or} \quad [\tau+1, \omega] \text{ if } \tau \neq O
\]

In the second case, the way to avoid loss of information is to keep it in an aggregated fashion. Prior to deletion of transaction data, the initial value for every account must be updated as of the end of the period to be deleted. That is, the values \(V(A, \tau)\) in the set \(U_A\), must always be the values as of the day before the first date for which transaction data are available in the system.

4. Non-aggregated Reports (Journals)

Journal is used herein meaning: a report disclosing the entries that verify the conditions set forth by the user in his/her request and ordered as established therein. Both criteria, for search and ordering, refer necessarily to one or more of the variables making up the entry heading. Two variants of this report are specified below. But first it is important to raise two preliminary issues:

1. There are limits for the retrieval of transaction data (see under 3.6 Limits to Transaction Data Writing and Retrieval). If the user’s request is bounded by two dates, \(\alpha\) and \(\beta\) (both inclusive), the system will reject the request unless \([\alpha, \beta] \subseteq [\tau+1, \omega]\).

2. Depending on how the entries are numbered (see 3.5 Date Format and Chronological Order Arrangement of Entries), it may not be possible to specify a type of journal where entry numbers bound the request.

Regarding the first issue, it is a constraint that will be included in the specification. As to the second, and taking into consideration that the objective of this paper is to describe a methodology rather than specifying a fully fledged system, two alternative journal types, both with chronological boundaries, will be specified, thus obviating this problem. However, it is easy to do otherwise once a certain entry-numbering sequence has been specified.

Apart from the data that the user must enter at the time of making his or her request, there are three stages of the process to be executed by the system that will be specified: search, ordering and conversion into interface (external) format.
4.1 Journal Bounded by Dates and Ordered by Entry Date and Entry Number

The user will enter the dates, $\alpha$ and $\beta$, bounding the journal. As stated above, in a valid request the following condition obtains: $[\alpha, \beta] \subset [\tau+1, \omega]$ (please note that $[\alpha, \beta]$ is a closed interval; entries dated $\alpha$ and $\beta$ will also be included). The mentioned three stages are specified below.

4.1.1 Search

The results of the search may be represented as a set: $J = \{E \in U_E | \delta(E) \in [\alpha, \beta]\}$

4.1.2 Ordering

The elements of this set have to be arranged by date, the number being the secondary criterion:

$$\text{Ord}(J) = \{E_i | i=1, 2, ..., \text{Card}(J)\}$$

such that:

$$\delta(E_i) < \delta(E_j) \quad \text{or} \quad \begin{cases} \delta(E_i) = \delta(E_j) \quad \text{and} \quad n(E_i) < n(E_j) \end{cases} \quad i < j$$

4.1.3 Conversion into Interface (External) Format of a Journal

Conversion is required for every entry included in the Journal. This operation involves substitution of debits and credits, as appropriate, for the values in entry-lines through application of the inverse of the function utilized during the conversion stage of the entry storage process:

$$\hat{\nu}_r(E_i) = |\nu_r(E_i)| \cdot \Phi^{-1}(\frac{\nu_r(E_i)}{|\nu_r(E_i)|})$$

$i=1, 2, 3, ..., \text{Card}(J)$

$r=1, 2, 3, ..., \text{Card}(E_i)$

$$\Phi^{-1}(1)=\text{Db}$$

$$\Phi^{-1}(-1)=\text{Cr}$$

4.2 Journal Bounded by Dates and Ordered by Year, Month and Entry Number

As for the previous type, the user will enter the dates, $[\alpha, \beta] \subset [\tau+1, \omega]$. Of the three stages specified in the foregoing section, only the second (ordering) is different for this second type. The elements of the set resulting from the search will be ordered according to «year and month», $N$ being the subsidiary criterion:

$$\text{Ord}(J) = \{E_i | i=1, 2, ..., \text{Card}(J)\}$$

where:
4.3 Introducing New Variables

As indicated in 3.1 Transaction Data (Accounting Entries) Input, it is easy to extend designs formally specified to more complex transaction data configurations. If working with the configuration matching the interface shown in Figure 3 the heading of an entry will then be:

\[
\begin{align*}
\text{Heading} \\
\bar{D}(E)=\text{entry date} \\
\bar{N}(E)=\text{entry number} \\
\bar{O}(E)=\text{type of transaction} \\
\bar{B}(E)=\text{branch} \\
\bar{R}(E)=\text{number of lines}
\end{align*}
\]

and it becomes possible to specify, for instance, journals containing only those entries corresponding to the branch established by the user in his or her request.

5. Account-level Aggregated Reports

In this section, some processes that require the sorting and aggregation of entry-lines are specified. The following relation is essential for this purpose:

\[
(r, \bar{A}, (E), \bar{V}, (E)) \text{ is related to } A \Rightarrow \bar{A}, (E) = A \\
\forall E \in U_E \\
1 \leq r \leq \bar{R}(E) \\
\forall A \in U_A
\]

5.1 Account-level Aggregates

For the composition of reports at this level, it is necessary to first define three aggregates.

5.1.1 Value of an Account on a Given Date

To compute the value of account \( A \), as of date \( \alpha \), it is necessary to do the following sum:

\[
V(A, \alpha) = \frac{\text{Cemd}(V_i)}{\sum_{i=1}^{\text{Cemd}(V_i)}} V_i
\]

\[
\{V_i\} = \{V(A, \tau)\} \cup \{\bar{V}, (E) \mid \bar{D}(E) \in (\tau, \alpha) \text{ and } \bar{A}, (E) = A \text{ where } 1 \leq r \leq \bar{R}(E)\}
\]

The first element of set \( /V_i/ \) can be retrieved from the set of accounts: \( \{V(A, \tau)\} \in U_A \), while the rest is to be chosen from the set of entries, \( U_E \).
5.1.2 Movement of an Account over a Certain Period

a) Movement on the debit side of an account over a given period.

Let $Db(A, [\alpha, \beta])$ be the total of debits to account $A$ over the period $[\alpha, \beta]$. This aggregate may be computed by adding up the elements of the following set (formed by elements retrieved from the set of entries):

$$\{V\} = \{\bar{V}(E) | \bar{E} \in [\alpha, \beta] \text{ and } \bar{A}(E) = A \text{ and } \bar{V}(E) > 0 \text{ where } 1 \leq r \leq \bar{E}(E)\}$$

The operation below will give the Debit movement:

$$Db(A, [\alpha, \beta]) = \sum_{i=1}^{\text{Card}([\alpha, \beta])} V_i$$

b) Movement on the credit side of an account over a given period.

The computation of the total of credits to account $A$ over the period $[\alpha, \beta]$, $Cr(A, [\alpha, \beta])$, works likewise, but leaves out the entry-lines with a positive value:

$$\{V\} = \{\bar{V}(E) | \bar{E} \in [\alpha, \beta] \text{ and } \bar{A}(E) = A \text{ and } \bar{V}(E) < 0 \text{ where } 1 \leq r \leq \bar{E}(E)\}$$

And it always holds that: $V(A, \beta) = V(A, \alpha - 1) + Db(A, [\alpha, \beta]) + Cr(A, [\alpha, \beta])$. That is, the value at the end of a period equals the initial value plus the movement over the period.

5.2 Account Statement

The normal statement gives the user, according to the terms of his/her request, the details of the movement of an account over a period of time. Together with the information available in the system about each debit or credit to the account, the statement discloses the accumulated balance after each movement. The user will enter the dates, $[\alpha, \beta] \subset [\tau + 1, \omega]$, and the account, $A$.

5.2.1 Search

For the composition of the statement, the following elements are necessary:

$$X(A, [\alpha, \beta]) = \{\bar{E}(E), \bar{N}(E), \bar{O}(E), r, \bar{V}(E) \in \mathbb{R}^5 | \bar{E}(E) \in [\alpha, \beta] \text{ and } \bar{A}(E) = A \text{ where } 1 \leq r \leq \bar{E}(E)\}$$

For simplicity, each vector in the set $X(A, [\alpha, \beta])$ is henceforth denoted by $X_i$; and the vector itself may be written:

$$X_i = (\bar{D}(X_i), \bar{N}(X_i), \bar{O}(X_i), r(X_i), \bar{V}(X_i)) \in \mathbb{R}^5$$

5.2.2 Ordering

For the arrangement, three criteria are applied: (1) as primary criterion, entry date; (2) as secondary criterion, entry number; and (3) as tertiary criterion, entry-line number. Thus, the
following set is obtained:

\[
\text{Ord}(\mathcal{X}(\mathcal{A},[\alpha,\beta])) = \{X_i; i=1,2,...,\text{Card}(\mathcal{X}(\mathcal{A},[\alpha,\beta]))\}
\]

where:

\[
\begin{cases}
\tilde{D}(X_i) < \tilde{D}(X_j) \\
[\tilde{D}(X_i) = \tilde{D}(X_j) \text{ and } \tilde{N}(X_i) < \tilde{N}(X_j)] \\
[\tilde{D}(X_i) = \tilde{D}(X_j) \text{ and } \tilde{N}(X_i) = \tilde{N}(X_j) \text{ and } \tilde{r}(X_i) < \tilde{r}(X_j)]
\end{cases} \Rightarrow i < j
\]

5.2.4 Conversion into Interface (External) Format

Account Statement Account From To

<table>
<thead>
<tr>
<th>Description</th>
<th>Movements</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Entry</td>
<td>Line</td>
</tr>
</tbody>
</table>

**Figure 6: Account Statement Interface Format**

For the user-interface format given in Figure 6, the first line of the statement, “initial balance”, is to be composed as follows:

\[
\begin{align*}
\text{Date} = \alpha - 1 \\
\text{Balance} = \begin{cases} 
\text{if } V(\mathcal{A},\alpha - 1) = 0 \text{ then } 0, \text{NOTHING}, \\
\text{otherwise } |V(\mathcal{A},\alpha - 1)|, \Phi^{-1}\left(\frac{V(\mathcal{A},\alpha - 1)}{|V(\mathcal{A},\alpha - 1)|}\right) 
\end{cases}
\end{align*}
\]

As to the remaining lines, the following conversion procedure is necessary:

\[
\begin{align*}
\tilde{v}(X_i) & \to |\tilde{v}(X_i)|, \Phi^{-1}\left(\frac{\tilde{v}(X_i)}{|\tilde{v}(X_i)|}\right) \\
\tilde{v}(X_i) & \to \begin{cases} 
\text{if } \tilde{v}(X_i) = 0 \text{ then } 0, \text{NOTHING}, \\
\text{otherwise } |\tilde{v}(X_i)|, \Phi^{-1}\left(\frac{\tilde{v}(X_i)}{|\tilde{v}(X_i)|}\right)
\end{cases} \\
i = 1,2,3,...,\text{Card}(\mathcal{X}(\mathcal{A},[\alpha,\beta]))
\end{align*}
\]
5.3 Trial Balance

This report is a list disclosing all the accounts in $U_A$, including the following information for each of them: initial balance as of the beginning of the period selected by the user, movement (debit and credit sides) over the period, and balance at the end of the period.

5.3.1 Search and Computation

The components of the trial balance will be represented by vectors, one for each account:

$$B[\alpha, \beta] = \{(A, V(A, \alpha-1), Db(A;\alpha, \beta), Cr(A;\alpha, \beta), V(A, \beta)) \in \mathbb{R}^5 \forall A \in U_A\}$$

Each vector includes five vector-entries: account, initial value, movement on the debit side, movement on the credit side and final value. For simplicity, each vector in the set $B[\alpha, \beta]$ will be denoted from now on by $B_i$, so that: $B_i = (\vec{A}(B), \vec{V}_{a-1}(B), \vec{Db}(B), \vec{Cr}(B), \vec{V}_{b-1}(B))$.

5.3.2 Ordering

The elements in the previous set will be ordered according to a pre-defined order-relation between accounts:

$$\text{Ord}(B[\alpha, \beta]) = \{B_i; i=1, 2, ..., \text{Card}(B[\alpha, \beta])\}$$

where: $\vec{A}(B) < \vec{A}(B_j) \Leftrightarrow i < j$. Note that $\text{Card}(B[\alpha, \beta]) = \text{Card}(U_A)$.

5.3.3 Conversion into Interface (External) Format

<table>
<thead>
<tr>
<th>Trial Balance</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beginning of period balances</td>
<td>Amount</td>
<td>Db/Cr</td>
</tr>
<tr>
<td>Movement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Credit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>End-of-period balances</td>
<td>Amount</td>
<td>Db/Cr</td>
</tr>
</tbody>
</table>

**Figure 7: Trial Balance**

Let Figure 7 be the desired format. The following conversion process is required:

$$\vec{V}_{a-1}(B) \rightarrow \{\text{if } \vec{V}_{a-1}(B) = 0 \text{ then } 0, \text{otherwise } |\vec{V}_{a-1}(B)|, \Phi^{-1}\left(\frac{\vec{V}_{a-1}(B)}{|\vec{V}_{a-1}(B)|}\right)\}$$

$$\vec{Cr}(B) \rightarrow (-1)\vec{Cr}(B)$$

$$\vec{V}_t(B) \rightarrow \{\text{if } \vec{V}_t(B) = 0 \text{ then } 0, \text{otherwise } |\vec{V}_t(B)|, \Phi^{-1}\left(\frac{\vec{V}_t(B)}{|\vec{V}_t(B)|}\right)\}$$

$t = 1, 2, 3, ..., \text{Card}(B[\alpha, \beta])$
6. (Item-level) Highly Aggregated Reports

A highly aggregated report is any report made up of items and operations with items. An item is a set of accounts. Relationships between accounts and items (Figure 1) are of the “one-to-many” type. A given item of a highly aggregated report may include one or more accounts, but each account can only belong to one item. On the other hand, these relations (between accounts and items) are variable; thus, the user may, at any time, redefine them (it would merely be a reclassification of the affected account within the report). With reference to the relations between items and reports (Figure 1), there are two (not mutually exclusive) alternatives: (1) in the case of reports predefined in the system design, these relations are constant (the user cannot change the composition of the report); and (2) in the case of those to be defined by the user (systems usually include a report-design facility) they are variable.

6.1 Specification Procedure

For the specification of a highly aggregated report, it is necessary to establish what follows.

1. The set of accounts over which the report will be defined. This set is a subset of the set of accounts existing in the system: $U_R \subseteq U_A$. For the balance sheet, it holds that $U_R = U_A$; for the income statement, $U_R = \{\text{Income and Expenses Accounts}\}$.

2. The items making up the report: $R = \{I_i; i = 1, 2, ..., m\}$ and the accounts belonging to each of them: $I_i = \{A_{j_i}; j_i = 1, 2, ..., n_i\}$, where $n_i$ stands for the number of accounts forming item $I_i$. The set $R$ is a set of classes, just like $U_O$ and $U_A$.

The value of an item, $I$, on a given date, $\alpha$, can be computed as: $V(I, \alpha) = \sum_{\forall A \in I} V(A, \alpha)$; and the covering condition stated below guarantees that the report is complete, not only at the time of its specification, but also later on, if new accounts are opened by the user. The condition implies:

a) The items are disjoint sets (of accounts):

$$I_i \cap I_j = \emptyset$$

$$\forall I_i, I_j \in R$$

b) The union of all defined items equals the set of accounts over which the report is defined:

$$\bigcup_{i=1}^{m} I_i = U_R$$

That is: $R$ is a partition of $U_R$ (a partition of a set $A$ is a collection of non-empty disjoint subsets of $A$ and whose union is $A$).

3. Operations with items to be disclosed in the report. The values of the items in a report are
subject to some (very simple) mathematical operations, such as

a) Adding the values (on a given date) of two or more items
b) Comparing the values of an item on two different dates: \( V(I, \alpha) - V(I, \beta) \)
c) Calculating the percentage of increase in the value of an item over a period of time:

\[
100 \frac{V(I, \alpha) - V(I, \beta)}{V(I, \beta)} \quad V(I, \beta) \neq 0
\]

4. The criterion for arranging items and operations in the desired order.14

5. The changes in sign to be made. These will differ between reports. For instance: (1) for the balance sheet, it is necessary to change the sign of every component of the liabilities and owner’s equity side; and (2) for the income statement (in the form of a list, whereby expenses are deducted from the revenue to calculate the profit or loss), the change of sign will affect every single component.

7. Summary and Future Research and Development

7.1. Summary

This paper has developed a formal method, based on elements of Linear Algebra and Sets Theory, for the client-specification of IS. The motivation is that IS-development may be greatly improved through a greater involvement of M&A specialists from the client’s side during the requirements analysis stage. To that end, it is desirable to establish a language for communication between them and the IS-designers that is both powerful and exact, and, at the same time, common to the M&A and IS communities.

LAST-based-modeling research was initiated by the authors in order to obtain an efficient alternative for teaching Computer Science students how IS work. To close the “culture gap” between IS and M&A specialists, the best time to start is at college. It would then be desirable to do likewise with Accounting and Management students.

LAST uses standard mathematical language, which is both precise and universal. In addition, the mathematical tools employed are few and fairly accessible (concepts of Linear Algebra for data storage and retrieval, and concepts of Sets Theory for data manipulation).

For illustration purposes, a FAIS that relies on the standard accounting DMF has been chosen. This choice is based on the widespread knowledge and use of the specificand, on the fact that it is a time-resilient and successful formalism, and on its relatively high complexity.

7.2. Future Research and Development (R&D)

Once a working methodology is established, it should be relatively easy to specify other IS’s subsystems (and alternative versions of the FAIS herein specified), and components providing integration of the different subsystems. In this regard, it could be useful to notice the existence of a pervasive isomorphism in the basic (raw) data; namely the structure: \{heading\} + a number of \{lines\}. Take, for one, what could be the formal specification, according to the methodology hereby discussed, of a sales invoice:

\[
Invoice = (D, N, C, O, T, R, L, Q_0, P_1, \ldots, L, Q_r, P_r, \ldots, L, Q_p, P_p) \in \mathbb{R}^{6*3}\]
Where:

\[ D = \text{date of the invoice}; N = \text{number of the invoice}; C = \text{customer ID number}; O = \text{sales order number}; T = \text{total, sum of } Q_r x P_r, \ r = 1, 2, ..., R; R = \text{number of items}; \]

\[ \text{and for every one of the } R \text{ items the three-tuple: } L_r (\text{item}), Q_r (\text{quantity}), \text{and } P_r (\text{unit price}). \]

It is possible to add as many variables as required: sales region, branch, salesclerk, etc; and this same structure would befit (among other transactions): a sales order, a purchase order, a store receiving report, or an employee’s payroll.

LAST modeling could be applied at the requirement-analysis stage of the life-cycle development process. LAST-expressed client-requirements would then be the input for the next stage of development: system-design. LAST-based specifications are consistent with: (1) basic principles of formal methods, as regards pre-conditions and post-conditions for input and output data (Hoare, 1983); and (2) traditional databases data-model integrity constraints. This implies that they can be translated into other constraint languages. For instance, in UML (Unified Modeling Language) methodology, the former would correspond to pre-post constraints written in OCL (Object Constraint Language), while integrity constraints would have to be considered in the entire design process.

It seems thus feasible to establish the following sequence for future R&D: (1) develop a fully-fledged LAST-based formal method for IS client-specification, (2) describe all the correspondences between LAST and UML categories (classifiers, relations, and diagrams of the UML-notation), (3) development of an application for LAST-specification (LAST-Interface) and for translation of LAST into UML (LAST-UML Translator), and (4) test the results of this application as an IS-design tool.

Footnotes

1. This paper consistently refers to LAST as a specification method. It does not work (nor is it intended to) for verification purposes.

2. Specially Accounting. In the academic field, since the formulation by Sorter (1969) of his events theory, most of the research aimed at better compatibility between Accounting IS and IT has focused on conceptual “multidimensional database” modeling of events theory-conforming AIS (for a concise review, see Murthy and Wiggins, Jr. 1993, 98-101).

3. “Specification of requirements should be formulated from the beginning at the highest possible level of abstraction, using all the available power of mathematics to describe the desirable, observable and testable properties of the product which is to be implemented.” (Hoare 1990).

4. To avoid any possible confusion between accounting entries and vector entries, the former will always be referred to as “entries” and the later as “vector-entries”.

5. Readers not familiar with transaction data processing for accounting purposes, may easily get a clear overview in Page and Hooper (1992).

6. Entries may be entered manually (semiautomatical entries), or they may be directly fed from other subsystems (automatical entries). This second possibility is a common feature of modular systems (Seddon et al. 1992, 106-107).
7. The codes and their related titles are previously entered through the user-interface and are stored. During the entry-storage process, the system only registers the codes, otherwise it would be storing redundant data. During manual journalizing, titles are displayed on the computer screen, purely to facilitate detection by the user of possible (basically typing) errors. Regarding the other possibility (automatic data input), whereby data are funneled from other subsystems, through so-called “feeder systems” (Seddon et al 1992, 95), it need not be noticeable in the specification of the FAIS. It should rather be addressed in the specification of the “source” subsystems; or, if dealt with as an “interpreter”, thus constituting a pathway between subsystems, when specifying the interpreter.

8. The numbering must follow a correlated sequence; the issue is to determine the frequency with which the sequence is restarted (beginning from 1). There are two alternatives that are very common in this respect: the yearly and the monthly basis; depending on which one applies, the resulting series of entries are yearly or monthly series.


10. For instance, in the case of the class transaction-type, by substituting the vector \([O_i, T(O_i)]\) for \(O_i\) as the elements of \(U_O\) and including, as part of the conversion into interface (external) format operation the change: \(\tilde{\Omega}(E) \rightarrow \tilde{\Omega}(E), \Pi[\tilde{\Omega}(E)]\), wherever necessary.

11. For instance, if the user very frequently requests account statements for the current month, the value as of the end of last month will satisfy the criterion of reference. If the user wants highly aggregated reports to include previous year-end figures, it is convenient to specify the set of accounts to include the value on that date.

12. Even within one series of entries, the order resulting from arranging entries according to their numbers is not necessarily chronological.

13. If accounts have been assigned numerical codes, arrangement following the codes will be satisfactory.

14. If numerical codes have been assigned to each component according to the desired order, this point is automatically solved; arrangement following the codes will be satisfactory.

15. This sequence will be followed in a R&D project, presently under way, to be executed by a group which (initially) includes the authors, J. Almendros, P. Vargas and A. Becerra (Universidad de Almería).

References


