A Systematic and Practical Approach to the Definition of Data

C. R. Symons* and P. Tijsma
N. V. Philips Gloeilampenfabrieken, Eindhoven, Netherlands

In methods and standards for the exchange of information, whether via transmissions over data communications networks or via shared databases, relatively little attention has been given to methods for exactly defining the meaning of the information being exchanged. In complex information systems, it is important to define the unambiguous meaning of data elements out of the context of particular messages, database records, or applications in which they currently appear.

To help solve these problems a methodology for formal definition of data elements using standardized terminology was developed within N. V. Philips Gloeilampenfabrieken, Eindhoven. The method is complementary to present-day naming of data items, is based on sound theory of data analysis and can be used to define data elements both in intercompany messages as well as in local systems and databases.

1. BACKGROUND

In recent years, despite the enormous attention given to data analysis, methods for solving the practical problem of how to define the basic data elements of information systems and databases have been approached mainly from only one direction, termed the ‘outside-in’ approach.

In this approach, various authors2–5 present a way of describing the ‘real world’ (or ‘outside’) and a method for translating that ‘inwards’ to a well-structured information system model. Figure 1, reproduced from Crawford’s paper, points out the alternative ‘inside-out’ approach. Crawford’s observations about the use of the two approaches need only a change of emphasis still to apply, namely: (1) ‘Outside-in’ approaches are gaining in popularity, but their practical application in the EDP world as a whole is still in its infancy. (2) On the other hand, whenever any significant automated information system or package is developed, or any data element standardization program is undertaken, records and fields are documented, and efforts are made to define and control data elements within and between systems. The ‘inside-out’ approach therefore still predominates, and such work is performed almost entirely along intuitive lines. This is a remarkable contrast. In most areas of study analytic and synthetic approaches are normally thought of as parallel and complementary.

In this paper, we start with the practical need to be able to define data elements. In the course of our work we came across the various ‘outside-in’ approaches and the confrontation with our ‘inside-out’ approach, we feel, gives interesting results.

The need to define and standardize data elements arises, for example, when making information-exchange agreements. These must be made regardless of the media or systems which will be used to exchange the data. Likewise a Data Administration Function when controlling the use of an enterprise’s data, will define data elements independently of their use in particular data-bases, messages or systems. For such activities it is a great help to have objective rules for defining data elements. Such rules must answer questions such as ‘How do I know my definition is complete?’, ‘How can I be sure that my definition has only one interpretation’.

Martin6 wrote that ‘over the next ten years many corporations will be carrying out the lengthy job of defining the thousands of data-item types they use, and constructing, step-by-step, conceptual schemas from which their databases will be built. The description of this large quantity of data will be an arduous task involving much argument among different interested parties’. The ‘arduous task’ and ‘much argument’ referred to can be reduced by a practical, objective definition method such as described in this paper.

*Present address: Rank Xerox Ltd, 338 Euston Road, London NW1 3BH, UK.

Figure 1. ‘Inside-out’ and ‘Outside-in’ approaches.1
As far as the authors are aware, there is only one published method for defining data elements, known as the 'Of-language'.

This aimed to produce unique, structured 'designators' for each data element so as to be able to store and retrieve them in a data dictionary. Use of the 'Of-language' is certainly better than an arbitrary approach to each definition, but the language has only a few rules for definition content, and it can certainly be improved.

Given the job of controlling a common interpretation and use of the data elements of a large system, one normally sets up a data element library. Such a library is set up either stand-alone, or as part of a general dictionary. An international study of the contents and use of several data element libraries in various EDP departments of N. V. Philips Gloeilampenfabrieken, and from external sources revealed many weaknesses.

Above all, incomplete, misleading and inaccurate 'definitions' of data elements were repeatedly found. Frequently, a data element library was used to control data elements during the analysis, design and development phases of a project, but fell into disuse after the system went live.

Against this unsatisfactory background this paper has three main objectives: (i) to examine how data elements are defined in practice and conclude what is wrong, (ii) to describe a practical method for the systematic definition of data elements which is in use in the business EDP environment of Philips, and (iii) to relate the practical method to certain theoretical ideas thereby shedding some new light on some of those ideas. The method to be described has been developed and tested over a five-year period. The principles of the method will be described, and illustrated with examples from the Philips business environment. Although the goal was to find rules for correctly defining data elements, in retrospect the method is now seen to be an integral part of data and system analysis and design at the level of abstraction commonly known as that of the conceptual schema or information model of an EDP System.

2. RESULTS AND LESSONS FROM DATA ELEMENT CONTROL IN PRACTICE HITHERTO

The study of the contents of 'data element libraries' referred to above showed the following problems.

(a) Confusion of fields and data elements. Generally fields of programs and records are documented, e.g. by symbolic name, description, format, etc. and this information is copied across to the 'data element library'. Fields and data elements are almost always taken to be synonymous. We feel, however, that it is essential to make a practical distinction.

A data element is defined as a basic unit of data which has a name, a definition, and a set of values for representing particular facts.

A field may be defined as merely a named space reserved in a specific location of a program or record for data values.

The Data Administration function, in controlling the use of the significant data elements of the enterprise, needs to know in which fields they appear. Perhaps in most cases there is a one-to-one correspondence between a data element in a controlled library, and a particular field. But, as shown in Fig. 2, there is often a one-to-many, or many-to-one, correspondence.

When an analyst defines a field, he instinctively relates the field to its immediate context or environment. In principle, however, the definition of a data element should not include any system contextual or usage information. (The data element is only bounded by the context of the enterprise.) Data element definitions may be valid for centuries in a particular enterprise, but field definitions may change with each system re-write.

The authors accept that a field and a data element may be viewed as different levels of abstraction or generalization of the same basic idea (a set of data). But practical experience has convinced them of the value of drawing a distinction. Figure 3 shows a list of the distinctions which can usefully be made.

(b) Poor definition quality. The second, and worst, error is to emphasize naming standards whilst omitting any objective rules for the definition of a data element.

Naming standards are, of course, important. Usually a standard symbolic, perhaps mnemonic name, and a 'user-' or 'long-name', are drawn up, a format and perhaps a 'description' are added. The 'description' may contain some sort of definition, references to related data elements (fields), an explanation of some of the terms used in the description, in short anything the author thinks might be useful. The lack of any objective rules for data element descriptions means that they are often incomplete, or mean little to anyone except the author. In data element libraries built up over many years, the terminology used for the same concepts changes with time and author. Hence, relations or differences between data elements which ought to be clear from the library, are usually obscure.

To correlate descriptions from different sources is often impossible without asking experts in the systems

---

Figure 2. Field/data element relations.

---

THE COMPUTER JOURNAL, VOL. 25, NO. 4, 1982  411

---

411
**Table:**

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Field (−TYPE)</th>
<th>Data Element (−TYPE)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identification</strong></td>
<td>- A significant context-dependent “User Name” (e.g. think of pro-forma field headings)</td>
<td>- “User Name” has little use.</td>
</tr>
<tr>
<td></td>
<td>- A symbolic name (code) for programming, unique within program.</td>
<td>- Coded identification, unique within enterprise.</td>
</tr>
<tr>
<td><strong>Definition</strong></td>
<td>- Determines scope and meaning of contents, and all “spatial characteristics” (see below)</td>
<td>- Determines scope and meaning of contents, and “own spatial characteristics” (see below)</td>
</tr>
<tr>
<td></td>
<td>- Always relates contents to other fields or records and the subjects they represent.</td>
<td>- May relate subject-concept to one or more other concepts, regardless of how those other concepts are represented or stored.</td>
</tr>
<tr>
<td><strong>Spatial Characteristics</strong></td>
<td>- Defined length.</td>
<td>- Length can only be determined if the representation is agreed.</td>
</tr>
<tr>
<td>- Own</td>
<td>- Defined format, appropriate to programming language.</td>
<td>- Character-type implied from representation.</td>
</tr>
<tr>
<td></td>
<td>- Defined position in record, program, etc.</td>
<td>- Positions are “where-used” information.</td>
</tr>
<tr>
<td>- Relative</td>
<td></td>
<td>- The set of values according to the definition.</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>- Empty, or a part, or one or more values of one or more data elements per field occurrence.</td>
<td>- Conceptual schema, information modelling.</td>
</tr>
<tr>
<td><strong>EDP Usage</strong></td>
<td>- External schemas, record layouts.</td>
<td></td>
</tr>
</tbody>
</table>
A SYSTEMATIC AND PRACTICAL APPROACH TO THE DEFINITION OF DATA

The method of analysing, defining and controlling enterprise data elements has three components: (a) a type classification system for enterprise data elements, (b) syntax rules for the structure and completeness of formal definitions, and (c) use of a controlled vocabulary of permitted terms for formal definitions.

(a) The type classification method. Classifying enterprise data elements is an aid to control and definition tasks, not an end in itself. When they are sorted into smaller classes the task of searching for a needed data element is easier. Also, analysing the definitions of all data elements in a given class brings out the common features, and thus helps the definition process. The standard type classification method used for Philips enterprise data elements is based on the principle of sorting them according to the type of information provided by their values. (Compare the ‘Of-language’ classification which is based on the type of value, i.e. there are classes for data elements whose values are codes, names, counts, amounts etc.)

In the Philips system one of the type classes is for data elements whose values identify geographical areas (continents, countries, districts etc.) regardless of whether the identification is via a code, name, number etc. and regardless of the roles or references to those areas in the information systems. Similarly, there is a class for data elements whose values give prices and tariffs. For a fuller description of the Philips classification system, see Section 4.

(b) Formal data element definitions. In 1976, E. M. Dozy and W. C. M. Swanenburg were engaged in a major study of the information systems of a Philips Product Division. This involved collating some hundreds of data elements representing quantitative and financial data from many sources. Needing to define each of these data elements in some standard way for the purposes of comparison, they realized that for this very large population of data elements there are certain independent standard aspects which must be represented in all definitions of the population, and for each aspect there is a limited number of terms, known as aspect-terms which are needed to represent the aspect in the data element definitions of the population. These ideas spurred a program of study of business data elements in many type-classes so that a pattern of aspects and aspect-terms has emerged. It means that for each type-class we may construct a standard table known as a List of Permitted Aspect-Terms, or LOPA. Figure 5 shows a simplified LOPA which helps solve the problem of defining many of the various delivery dates shown in Fig. 4.

To define any particular data elements, we first find its type-class, and hence the appropriate LOPA. Its formal definition is then made by taking the one appropriate aspect-term for each relevant aspect from the LOPA. This string of terms is a definition expressed in a syntax which is independent of natural languages. (The semantic
relations between the terms is found in the definitions of the aspects.) If LOPAs and formal definitions are made, say, in English for all data elements of interest, then it is possible (i) to derive a user ‘description’ definition in ordinary English, understandable to non-experts in formal definitions. The user description should be precise due to the firm basis of the formal definition. (ii) to translate automatically the English LOPAs and formal definitions into any other language for which there is a one-to-one translation table of the aspect-terms. User descriptions in the new language can of course then be derived; (iii) to store the LOPAs and formal definitions in a computer, e.g. in a data dictionary so that the uniqueness of each formal definition can be checked. When a new formal definition is required, it can be checked whether it already exists, or whether new aspect-terms have to be added to the LOPA, etc. A LOPA is therefore a sort of open-ended, data element standard definition generator for a given type-class. Standard LOPAs have been published internally within Philips. They allow selection of the formal definitions of data elements of interest out of the millions of data elements which could be theoretically generated.

(c) Controlled vocabulary. Aspect-terms must be controlled not only per LOPA, but also across different LOPAs. The aspect-terms must themselves be defined and a controlled vocabulary of permitted aspect-terms established. It is no use having a few controlled data elements using the term ‘cost-price’ in their formal definitions if this term is used by two groups to mean different things (‘homonyms’) and a third group uses the term ‘factory-price’ for the intended concept (‘synonyms’). Concepts expressed by homonyms must be carefully distinguished, and synonyms must be recognized in a controlled way. Building up this standard vocabulary of terms for the enterprise is pure systems analysis work which cannot be automated. It will normally be done by the data element controller (or data administrator) in consultation with the end-users and analysts of the systems concerned.

It may be argued that the problem of data element definitions has been simply shifted to a problem of defining terms. This is partly true, but this problem was always present before the idea of formal definitions arose. The advantage now is that terms which may be related, or should be distinguished will now be grouped mostly under the same aspect in the LOPAs of one or more type-classes. The distinctions and relations which must be made are now easier to find. Techniques for defining aspect-terms are summarized in Section 6.

4. THE TYPE CLASSIFICATION METHOD: DETAILS

We recall that the standard type classification method is simply an aid to control, definition and retrieval of enterprise data elements in a library or dictionary. Ideally it should be wide enough in scope and open-ended to accommodate all the data elements of the enterprise which will eventually need to be controlled, have a simple hierarchical structure where at the lowest level each type-class contains a ‘reasonable’ population of standard data elements, say 10 to 100, be totally independent of any organizational or system usage structure, and be simple and teachable; the potential users must be able to work out the type-class of a required data element with at least a 90–95% first-time success-rate.

The Philips classification system has grown empirically into a three-level system of Main Classes, Classes and Types; experience in training workshops and in practice shows that it meets the above retrieval success-rate objective.

The classification system will first be described at length, after which some questions, e.g. about its relation to other possible classification approaches will be discussed.

At the highest level, enterprise data elements are divided into those whose values give qualitative data and those which give quantitative data.

Let us examine the first group in detail, leaving aside the latter (also referred to as Measures) for the time being.

The Main Class of data elements whose values give qualitative data, is known as ‘Main Class A Identifiers’.

---

<table>
<thead>
<tr>
<th>ASPECTS</th>
<th>KIND OF VALUE</th>
<th>SUBJECT CONCEPT</th>
<th>VALUE FORM</th>
<th>STATUS</th>
<th>ROLE</th>
<th>RELATED CONCEPT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CODE</td>
<td>DESCRIPTION</td>
<td>DATE</td>
<td>ABBREVIATED</td>
<td>ORDERED</td>
<td>ISSUE</td>
</tr>
<tr>
<td></td>
<td>DATE</td>
<td>DESCRIPTION</td>
<td>WEEK</td>
<td>FULL</td>
<td>CONFIRMED</td>
<td>SHIPMENT</td>
</tr>
<tr>
<td></td>
<td>ASPECT-TERMS</td>
<td>CODE</td>
<td>DATE</td>
<td>ABBREVIATED</td>
<td>ORDERED</td>
<td>ISSUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DESCRIPTION</td>
<td>DATE</td>
<td>YWWD</td>
<td>YWW</td>
<td>YYMMD</td>
</tr>
</tbody>
</table>

Example "Formal Definition"

Code, Date, YWWD, Ordered, Receipt, Product.

Corresponding possible "User Definition"

Date, coded in the form YWWD. Stated in an order, at which the customer wants to receive the goods.

Figure 5. 'Delivery-date' definition problem. Simplified 'List of Permitted Aspect-Terms' (LOPA).
The name for this Main Class is not ideal because it has a different meaning in conventional database jargon, but no more suitable name could be found.

Identifiers

Identifiers are data elements whose values uniquely identify or help identify certain objects or subject-concepts, or classes thereof regardless of the roles played by those concepts.

(The authors strongly prefer 'subject-concept' to 'object'. 'Object' is suggestive of something tangible whereas 'concept' clearly includes abstract notions. Also 'subject' is necessary to distinguish the concept which is identified, or whose property is indicated, or which is measured, in the quasi-sentence of a data element definition. 'Object' has the wrong grammatical connotation. We use 'concept' in the widest sense. ('A concept is the meaning of a word'—Langefors.)

The Main Class for Identifiers is broken down into nine Classes and further into Types, depending on sub-groupings of the subject-concepts. The subject-concepts referred to are grouped at the Class Level as shown below:

1. Points or areas in geographical 'space' (countries, districts, ports, addresses, etc.)
2. Points or areas in organizational 'space' (organizations, departments, functionaries, networks, etc.)
3. Points or extents of 'time' (periods, dates, times of day).
4. Individual Persons (employees, pensioners, etc.)
5. Products, and the 'resources' that are necessary for the manufacture (machines, tools), transport (lorries, containers, ships) and packaging of products, etc.
6. Information units (transactions, orders, files) and the 'resources' that are necessary for the storage of information (volumes, media).
7. Units of Measure and Currencies.
8. Accounts, projects, activities and such-like.
9. Miscellaneous e.g. purely abstract concepts (colour, language, etc.), which are not exclusive properties of any of the other subject-concepts.

We must emphasize that the above list of subject-concepts happens to apply for Philips' business, i.e. manufacturing, commercial and administrative activities. It would need expansion if research and development data elements should also be included.

The following illustrates some 'classes thereof' (see Identifier definition) given by the values of Identifier data elements:

4. Sex, or grade, (of persons)
5. Make-or-buy-class, inspection-procedure, stock-booking-procedure (of products)
6. Urgency-class, processing-status (of orders).

Such classes of subject-concepts are formed by selection on either (a) permanent or temporary properties of the subject-concepts concerned, or (b) rules related to the handling of the subject-concepts, or to processing information about them in some way. Such properties or rules must apply exclusively to the object-types of the type-class concerned. For example 'make-or-buy-class' is a property of object-types such as 'product', but not of object-types such as 'person' or 'order'.

Measures (quantitative data elements)

Next we turn to data elements whose values give quantitative data about the subject-concepts referred to by Identifiers. They are also known as Measures, and are divided at the highest level into five main classes:

- M Amounts (i.e. Financial amounts)
- P Prices and Tariffs
- Q Quantities (including lengths, weights, etc.)
- R Ratios, Percentages, Indices, Factors, etc.
- T Times

Each of the above main classes may be further split, if relevant, into four classes. (1) Measures of 'situations' (quantities, amounts etc. of stocks, assets etc. which are debited and credited); (2) Measures in or related to single transactions, movements, etc.; (3) Measures in or related to aggregations of transactions or movements over periods of time (these are always set to zero at the beginning of the time-period); and (4) Measures in hierarchical and other structures e.g. parts-lists, and constants. The above distinction can, if relevant, always be reliably made; it also helps in understanding the aspects needed for the formal definitions.

Further division of measures to a third level can be made, if relevant, on the basis of the type of Object Measured, i.e. the subject-concepts of Identifiers as described above.

Example: a quantitative sum of movements of employees over a period of time would be classified in type (-class) Q34, of products in Q35, and of orders in Q36. Similarly, the financial value of a 'stock' of buildings would be in Type (-class) M11, and of products, machines, vehicles, etc. in M15.

Not all of the measure type classes from the above hierarchy are of practical importance within Philips (There are no data elements which give prices of organizations, and we unfortunately do not have stocks of times!). Of the roughly 300 types which could be defined, only about 60 appear to be of practical importance.

Evaluation of the type classification system

The Type Classification system may seem highly empirical at first reading. It evolved in that way, but as it now stands, it has a number of fundamentally significant features.

The information systems of Philips' business, like those of many other businesses, are mainly concerned with supplying quantitative performance data to the end-user i.e. the Measures of the five Main Classes.

To interpret the value of any of these Measures they must always be considered in relation to the values of Identifier data element types, i.e. (a) the point or extent in geographical/organizational 'space', for which the Measure applies; (b) the point or extent in 'time' for which the Measure applies; (c) the 'thing' or class of 'things' measured (in Philips business i.e. people, products, machines, vehicles, etc., information units, accounts, projects); (d) the Units of Measure.

The Type Classification system therefore helps towards understanding the data element definition problem, as
well as forcing data elements into groups of the same 'Type' as an aid to retrieval and definition.

Practical experience has shown the danger of trying to define a single data element in isolation. Looking at a group of data elements of the same Type helps one discover their common definition structure and terminology similarities and differences. Once these factors are understood, the formal definitions of all members of the group can be rapidly written and with much less of the argument referred to by Martin.6

By now anyone familiar with the concepts of the entity-attribute model7 for data analysis may be trying to see parallels with the concepts of the data element Type classification system.

In the terminology of our system, the terms 'entity' and 'attribute' have been carefully avoided. It has been said that decisions about entities and attributes are difficult because 'one man's entity is another man's attribute'. The type classification system sets out to be quite independent of any existing or planned system view of the real-world concepts of the enterprise. Nevertheless, it is perhaps helpful to show where parallels do and do not exist with the entity-attribute model.

A Qualitative Type (-class) contains all data elements whose values 'identify' the subject-concepts of that Type. Thus the Identifier Type (-class) for Organizations contains all data elements whose values identify Organizations or classes thereof. (a) whatever the type of Organization (customers, customer-groups, suppliers, divisions, departments, stores, plants, etc.), (b) irrespective of the various coding or naming systems used to identify the Organizations, (c) irrespective of the rôles of the Organizations-types in relation to other concepts (e.g. a department may be the holder of stock, the issuer of an order, the cost-centre for an employee, etc. at various points in the information systems).

By contrast, in an entity-attribute model only certain subject-concepts will be viewed as entities (things about which we want to hold data in the systems under consideration). The other subject-concepts will be viewed as subjects of data elements which define attributes of entities.

It is worth emphasizing and repeating that Identifier data elements as meant in this paper may or may not uniquely identify their subject-concept. (In the sense of Chen8 an identifying attribute is one which uniquely identifies an entity). 'Unique' means something only in a specified context, e.g. an employee name may be unique within a department but not in the enterprise. Data processing systems usually need coding systems as 'entity-types' identifiers, whereas the equivalent naming or description systems are merely other attribute-types of the entity-type. The end-user regards this as a nuisance of data processing. He cannot work just with codes; he needs names. In controlling data elements at the enterprise level we must therefore pay equal attention to controlling coding and naming systems, and any other systems of identifying descriptions or text, and to controlling their equivalences.

Many Identifier data elements give attribute-types which are classes to which particular entity-types belong. For example, 'make-or-buy' could be an attribute-type of the entity-type 'product'. It is a fact of life that in complex information systems there are often hundreds of these classes which are used for analysis, trigger-procedures, etc. (They are often known as 'Indicators'). It is our experience that in practice such 'Indicator' data elements are often very loosely defined.

To begin with, there are two ways in which the information given by such data elements can be expressed. A 'Make-or-Buy' data element in a product record with values 0, 1 could for example be defined in either of the two following ways

(a) 0—indicates the product belongs to the class of products which is made in-house;
(b) 0—indicates the product is made in-house

1—indicates the product is bought-in

1—indicates the product belongs to the class of products which is bought-in.

These are two alternative expressions of the same data element since one meaning is automatically derivable from the other. This intrinsic ambivalence is also discussed by Kent.10

However, the same data element which gives an attribute-type of the entity-type 'product' may be used in practice to accumulate information about 'make' or 'buy' products. In theory, however, we are now really talking about a different data element with the definition

(c) 0—indicates the class of products which is made in-house;
1—indicates the class of products which is bought-in.

This data element represents the entity-types 'make-products' or 'buy-products'. The distinction between (a) and (b) on the one hand and (c) on the other is rarely clear in data element libraries in practice.

Measure data elements, that is data elements whose values give quantitative data, are sometimes regarded as 'attributes of relationship-types' (or 'intersection entities') in the Chen entity-attribute model. However, physical constants, e.g. weight of a container, or length of a route, are regarded as normal attributes of their respective entity-types.

Note: There are many variations on the entity-attribute model. We refer to the Chen version here because it is widely known. Within Philips a variation is used which is in line with developments of the International Standardization Organization (ISO/TC97/WG3).

Summarizing the relation between the Type Classification System and the Chen entity-attribute model concepts:

(1) There is no one-to-one correlation between Identifier data elements and any concept of the Chen entity-attribute model. On the other hand all qualitative entity-type-identifiers are Identifiers.

(2) Measures are usually regarded as attributes of relationship-types in the Chen model, or as attributes of entity-types.

5. FORMAL DEFINITIONS: UNDERLYING THEORETICAL IDEAS

The conclusions of Section 2(b) on constructing formal definitions were arrived at by observation and analysis following the 'inside-out' approach. The purpose of this paragraph is to confront these conclusions with certain theoretical ideas mainly associated with the 'outside-in'
A SYSTEMATIC AND PRACTICAL APPROACH TO THE DEFINITION OF DATA

approach. These ideas which will be treated as simply as possible for the purposes of this paper consist of: (i) considering a data element as a set and (ii) the rôle and sort/sub-sort relations between concepts, as explained by Nijssen and Sowa.5

The data element and sets

Very simply, a data element has a set of data which share a common definition (and probably a common name and other characteristics as well). Three very important rules for any data element definition—formal or informal—arise from this idea.

The definition of a data element must (i) give the common meaning of all members of the set, (ii) enable the user of the definition to distinguish and interpret as far as is possible within the context of the enterprise and its activities, the meanings of all individual members of the set, and (iii) correspond precisely with the population of the set concerned. These rules help us judge the usefulness and completeness of data element definitions. The following examples use informal definitions. Consider the set which has the definition 'code of a country'. This definition satisfies rule (i) but not (ii) since the definition does not tell us how to interpret the codes and therefore the meaning of the individual code values. In fact as a data element it is quite useless, because this set includes all codes which identify countries from all coding systems in the universe. It does not satisfy any reasonable requirement of rule (iii).

A reasonable, useful set, by comparison might have the definition 'alpha code of a country according to ISO standard ISO-3166-1974.' This definition refers to a specific coding system (of the International Standardization Organization) for a set of named countries. (We must be satisfied that the general knowledge of the user of the definition is enough to understand the names).

A further example is the set with the definition 'Name of the country of legal-origin of the product'. This is a perfectly good set provided we can find out from our Vocabulary what is meant by 'name', 'country', 'legal-origin' and 'product'. We should be clear that the set includes all names of all countries in all languages, with all possible spellings, full and abbreviated. It is a perfectly good set for humans, but is only usable for computers if we 'teach' them all synonyms of country names in different languages. (The data element occurs on many forms).

From thinking about a data element as defining a set, we conclude that each Aspect-term added to a formal definition narrows down the meaning of the set, and its population. Conversely omission of a possible Aspect-term implies that the Aspect concerned is irrelevant for the data element being defined; there are no omission defaults on the set interpretation.

Role relations between concepts

Nijssen has written that 'each communication between a human being and an information system, as well as each communication between any two information systems can be considered to consist of a set of natural language sentences' and 'Without loss of generality we can define that a natural language sentence ... consists of a set of atomic sentences'.

An atomic sentence is of the form

subject verb object

or expressed in another way

subject-concept rôle related-concept.

The above general statements of Nijssen can be used to provide rules for the underlying structure of data element formal definitions, which can also be reduced to atomic sentences.

<table>
<thead>
<tr>
<th>Name</th>
<th>Birth-Date</th>
<th>Birth-Place</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>51 01 01</td>
<td>London</td>
</tr>
<tr>
<td>Jones</td>
<td>52 02 02</td>
<td>Cardiff</td>
</tr>
<tr>
<td>Stewart</td>
<td>53 03 03</td>
<td>Glasgow</td>
</tr>
<tr>
<td>Riley</td>
<td>54 04 04</td>
<td>Belfast</td>
</tr>
</tbody>
</table>

Figure 6. Employee table.

Consider the simple table of Fig. 6. This table may be defined by means of the three data element formal definitions:

Name, employee.

Code, date, YYMMDD, birth, employee.

Name, place, birth, employee.

With this trivial example we may now explain and illustrate many points about data elements and their formal definitions.

(a) The fundamental Aspects of Identifier data element formal definitions are:

<table>
<thead>
<tr>
<th>Kind of Subject-Value</th>
<th>Rôle</th>
<th>Related-Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>02</td>
<td>15 25</td>
</tr>
</tbody>
</table>

Term Terms
e.g. Name Place Birth Employee

(Each of the Aspects has been given a two-digit number which indicates the sequence of Aspect-Terms in formal definitions).

Here we see that (1) Any Identifier formal definition must have at least Aspects 00 and 02 (There is an implicit rôle 'identifier' between the Kind of Value and Subject-Concept). (2) Any Aspect-Term which can represent the 'Subject-Concept' Aspect can equally well represent a 'Related-Concept' Aspect. (3) The 'Rôle' Aspect gives the 'view' of the 'Subject-Concept' from the 'Related-Concept'. (in the above example 'place' has the rôle 'birthplace' as 'seen' from 'employee'.)

(b) Formal definitions may be extended to define the data element required, essentially by adding a rôle/related-concept pair (Aspects 15, 25) to the initial Aspects 00, 02. Thus if we wish to distinguish the two dates of signature of an expense claim this can be done with the following two formally defined data elements

<table>
<thead>
<tr>
<th>Descrip- date, sign-of, claim, sign-by, employee.</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 02 15 25 15 25</td>
</tr>
<tr>
<td>Description, sign-of, claim, sign-by, employee.</td>
</tr>
<tr>
<td>00 02 15 25 15 25</td>
</tr>
</tbody>
</table>
(Here the term 'description' is defined in our Vocabulary for formal definitions, as 'any informal identifying character string intelligible to humans'). Figure 7 shows these two data element definitions using a diagramming convention for a ternary association as used within Philips.

![Diagram](https://example.com/diagram)

**Figure 7. Diagrams for claims-signature dates (ternary association).**

(c) The Aspect '00: Kind of Value' expresses the principal way in which the value may be expressed; it may also be considered as a 'naming class'. Common permitted Aspect-terms for this Aspect are:

- 'Name', 'Code', 'Description', 'Reference', 'Number'.
- 'Numerical-value' (for Measures).

In many cases other Aspects are necessary to qualify the 'Kind of Value'. For example, if the 'Kind of Value' is a code, then we may need three Aspects to determine which coding system is intended, viz:

- 03: 'Naming sub-class'
- 04: 'Standards Authority'
- 05: 'Standard Document'.

Thus a data element which was informally defined above as '(Alpha) code of a country according to ISO standard ISO-3166-1974', has a formal definition 'code, country, alpha, ISO, ISO-3166-1974' where the Aspect-terms 'alpha', 'ISO' and ISO-3166-1974' correspond to Aspects 03, 04, and 05 respectively.

If we wish to distinguish different versions of a Name (e.g. including 'full' or 'abbreviated' as Aspect-terms in the definition), then we can also use the 03: Naming sub-class' Aspect which qualifies 'Kind of Value'. Moreover, a certain 'naming version' may be represented in different languages for which we need 'Aspect 06: Language'.

In summary, therefore, at least four Aspects (03, 04, 05, 06) may be needed to qualify Aspect '00 Kind of Value' to completely define how the values of qualitative data elements are to be interpreted (for 'Measures' other Aspects apply).

(d) The data elements which we need to define in practice involve one 'subject-concept', which may be linked to one or more 'related-concepts'. At the risk of confusion with Nijssen's 'atomic sentence' terminology, it seems helpful to consider: (i) a data element whose definition has only a subject-concept and no related-concept is an **atomic element**, (ii) a data element whose definition has a subject-concept and one or more rôle-linked related-concepts is a **molecular element**.

With this analogy we therefore in practice define an atomic element for any subject-concept which has an autonomous existence in the enterprise. Definition of Atomic elements is necessary for control of name assignments. The subject-concept of an atomic element may (but need not) be a subject (or 'entity-type') for a record. Otherwise the data elements in which one is interested are molecular elements.

Examples of Identifier atomic elements:

- Name, employee, full.
- Code, employee, Personnel-Dept, PE-123.
- Code, date, YYMMDD.

Examples of Identifier molecular elements:

- Code, employee, Personnel-Dept, PE-123, manager, department.
- Code, date, YYMMDD, birth, employee.

(e) Having established the basic structure of data element formal definitions with the aid of simple example Identifier data elements we can now turn to those Identifier data elements which are often referred to as 'Indicators'. These of course have the same underlying structure, but given the nature of 'Indicators' we can simplify the rules for their formal definitions.

The 'subject-concept' of such Indicators is inevitably complex, being composed of a basic-subject-concept, and the property which determines the class of subject-concepts. We therefore separate the property to its own Aspect (01).

Indicator formal definitions therefore have the following minimum essential aspects:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>02</td>
</tr>
</tbody>
</table>

Example aspect-terms

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>make-or-buy</td>
<td>order</td>
</tr>
</tbody>
</table>

These minimum aspects should be extended with aspects 02-06 if required, as for Identifiers. Implicit in the above formal representation is that the code determines either the 'property' ('make-or-buy') or the class of 'subject-concepts' ('product') selected on the property ('make products' or 'buy products'). The two interpretations are equally legitimate. The 'property' is of course itself a concept, and there is an implicit role relating the 'property' to the 'subject-concept'; this rôle-term is always 'Property-of' for this type of data element (using aspect 01), and is therefore left unstated in their formal definitions.

(f) Measure data elements mostly have more complex formal definitions, although still based on the same underlying structure already described. Only rarely in business information systems do we come across meaningful atomic elements which are Measures. Examples would be sets of pure physical constants.

The 'Kind of Value' Aspect has virtually only one possible Aspect-term namely 'numerical-value'. In our Vocabulary this is defined as 'any real number in the decimal numbering system'. (The only practical alternative to 'numerical-value' could be 'description' which is sometimes used when amounts of money are written on cheques and in contracts, etc.) Use of
the term 'numerical-value' makes it clear that we are only referring to the 'number' and not including any data about the unit-of-measure. The latter is a separate concept from the Measure-concept.

A formal definition of the form numerical-value, stock-quantity is quite useless, remembering the arguments in the paragraph on considering the data element as a set.

In order that the formal definition gives the meaning of each individual value of the set we must examine the rôle-relations between the Measure ('stock-quantity') and various other related concepts.

In practice the majority of Measure concepts of business data elements need a minimum of three or four, and less frequently, two related-concepts. When defining a Measure data element, we should always ask five questions. (i) What kind of Measure is it in the functional sense? (think of the first two levels of the Measure data element type classification system). (ii) What is the 'object measured'? (Think of the third level of the Measure type classification system.) (iii) In what units is the 'numerical-value' expressed? (iv) Where is the Measure in geographical or organizational 'space'? (v) When is the Measure in time? Over what period and/or at what point in time does the Measure apply?

The answers to these five questions show which concepts are rôle-related to the Measure. Figure 8 shows a Nijssen-like diagram for a Measure molecular element in which the Measure concept has four related-concepts. Some of the questions may however be irrelevant to the data element to be defined. Some of the questions may yield several answers. The following examples illustrate various cases.

(1) The formal definition of a data element whose values give the weight of a product (which is invariant in space and time) may only involve three variable concepts, namely 'weight' (the Measure naming-class), 'product' (object-measured related-concept) and 'unit-of-measure' (another related-concept).

(2) The 'object-measured' may not exist as a specifically recognizable concept in the information system. For example if the sales-quantity Measure of Fig. 8 were not of a specific product, but instead of 'all goods' or 'everything', then this can be accommodated in the formal definition by simply omitting any reference to an object-measured. On the set-interpretation, omission implies 'irrelevance' or 'of whatever', and therefore 'of everything or anything'. (For presentation reasons we prefer to include the term 'goods' in formal definitions in such circumstances, to make it clear that there is somewhere an 'object-measured'.)

(3) In an automated information system in which stock quantities are maintained in real-time, the related time concept may not be explicitly recognized within the system, but time must be recognized in an out-of-context formal definition.

(4) A Measure occurring in a single transaction (such as would belong to a Type-class which had '2' in the second position of the Type-class code) needs special treatment. At first sight, questions (iv) and (v) show that a 'single order quantity' was ordered from one organization, for supply by another, for delivery by another, for delivery to another location, ordered on a certain date, for delivery on another date, etc. This apparently complex structure is avoided by recording the rôles-relations of the order-quantity with the goods-item identified via an order-line, and order. This should be no surprise; in normal business activities each order has an identifying reference number which is unique and fixed in space and time for the supplier or for the customer, or both. The order 'carries' all space and time information wherever it is recorded. We therefore answer questions (iv) and (v) indirectly in such cases by including 'order-item' and 'order' in the formal definition, instead of the specific space and time aspects.
Additionally there are usually many terminology problems to be overcome in separating the various concepts. Practical solutions to some of the latter problems will be dealt with in para 6.

Sort/sub-sort considerations

Concepts may be related not only by rôle-relations, but by sort/sub-sort considerations. Subsorts of concepts should not be confused with sub-sets of values. Sowa offers a formalized definition of a sort/sub-sort relation.\(^5\)

Sort/sub-sort relations give the degree of 'overlap' between concepts. They may exist between concepts which are permitted for any one Aspect of any LOPA for any Type-class, and should be recognized in the controlled Vocabulary, just as importantly as synonyms.

Examples

(1) ‘Order’ and ‘Invoice’ are of different sort. An order is never an invoice, and vice versa. They do not overlap. However, both order and invoice are sub-sorts of document.

(2) Similarly both ‘customer’ and ‘supplier’ are sub-sorts of ‘organization’. But they may well be overlapping sub-sorts, if some organizations are both customers and suppliers.

(3) Multiple level sort/sub-sort relations may exist. ‘Manager’ is a sub-sort of ‘employee’, which is a sub-sort of ‘person’.

(4) Sort/sub-sort relations may also be considered to exist when an Indicator data element is used to classify a subject-concept by its property. This produces a sub-sort of that subject-concept (The entity-attribute model refers to ‘sub-entity types’).

Recognizing sort/sub-sort relations in the analysis of data elements is very important for understanding and controlling complex information systems. Again examples will illustrate.

(5) In information system design, data elements should always be chosen which involve the highest sort concepts. Thus one coding system for ‘organizations’ will generally be more flexible than separate coding systems for customers, and suppliers, if the latter overlap substantially. This then requires that permitted rôles are recognized for each organization such as seller, buyer, delivery-point etc.

(6) Adoption of a term such as ‘reference’ which is defined in our Vocabulary as a super-sort of ‘code’, ‘name’, ‘description’ etc. enabes formal definitions such as

Reference, order, customer.

(‘A customer’s identification of an order’)

This accurately defines a data element that exists in ordering transactions. It cannot be defined without this super-sort term.

(7) Recognition that ‘order’, ‘invoice’, ‘shipment-advice’ and the dozens of other document-types are all sub-sorts of ‘document’ permits the definition of one data element

code, date, YYMMDD, issue, document

which may save defining and controlling the dozens of ‘issue-date’ data elements for each document-type.

6. SOME TECHNIQUES FOR PRACTICAL SOLUTIONS TO TERMINOLOGY PROBLEMS

The ideal structure of formal definitions which has been described is difficult to maintain in the face of normal, established language usage. Here we find two sorts of problems:

(a) Some concepts, especially Measures, cannot be expressed with single terms. In practice several Aspects may be needed to unravel the concepts which exist in the enterprise.

(b) In some cases certain Aspects, e.g. Rôle Aspects, are implicit and may be ignored. In other cases it seems advantageous to ‘blur’ or simplify the real Aspect structure. If these steps are not taken then writing formal definitions can in some cases become a tiresome business.

An example of (a) has already been seen, namely to express the ‘Kind of Value’ Aspect fully, we need to be able to qualify by several other Aspects (‘Naming subclass’, ‘Standard Authority’, ‘Standard Document’, ‘Language’, etc.)

Measure subject-concepts are very often quite complex, e.g. ‘Budgetted—exponentially—smoothed—total—gross sales—amount’. Several Aspects must be defined to help break down such complex expressions. The set of Aspect-terms to be established in the LOPA’s and controlled Vocabulary must as far as possible be established normal terms.

We have also already seen examples of the reverse problem (b), e.g. in the suppression of the rôle-term ‘property-of’ which links the Aspects ‘Property’ and ‘Subject-concept’ of Indicators. Similarly the rôle-term which links ‘Budgeted Sales Amount’ and ‘Year’ is ‘sales-year’. It seems superfluous to include this in formal definitions. There are a number of cases where this is true for Measure Aspects, although suppression of Aspects must be done with great care.

Turning to the problem of controlling the Aspect-terms within an Aspect, there are many checks to help arrive at a ‘clean’ Vocabulary and set of LOPA’s.

(c) Any Aspect-term which represents a Subject-concept must also be able to represent a Related-Concept and vice versa.

(d) To avoid confusion, Aspect-terms which are chosen for ‘Rôles’ must not also be used for either Subject- or Related-concept Aspect-terms (and vice versa).

(e) When choosing Aspect-terms, care must be taken to distinguish terms which represent an individual, type, or class of a concept, and populations thereof.

(f) Super-sort/sub-sort/sub-sort relations must be unravelled and recorded.

(g) Synonyms must be uncovered and probably recorded.

(h) Homonyms must be eliminated.

(i) Aspect-terms must be defined in the Vocabulary down to the level of normal non-jargon language. Diagrams are sometimes useful to show relations between Aspect-terms and to help avoid circular definitions.

(j) Attention must be paid to the grammatical form of the terms used (unless one has the sort of facilities obtainable with information retrieval systems). Sub-
A SYSTEMATIC AND PRACTICAL APPROACH TO THE DEFINITION OF DATA

7. CONCLUSIONS

The data element analysis method described in this paper has been developed and used over five years in N. V. Philips. Its use and acceptance is steadily growing. Where the method has become established it is seen by system analysts as changing the task of defining data elements from an art to a science. We now find that analysts persevere to obtain a correct formal definition since it gives them greater insight and certainty in their analysis and specifications. Documenting data elements is not now a dull side-issue, but an integral part of the analysts' job.

For those concerned with standardization of data elements, setting up LOPA's has resulted in a ten-fold increase in productivity, and certainty in the quality of output. As the number of standardized aspect-terms grows, so can the output of accurately defined data elements grow more rapidly. The vocabulary and the LOPA's will need constant maintenance as business information requirements and language evolve. The LOPA's have been used successfully for multi-lingual definition comparisons and translations.

The method is teachable, although following a one to three day workshop, guidance is necessary from an experienced practitioner for some period. The method is also well suited to automation, having been successfully implemented on both an in-house and a commercially available data dictionary.

A relation has clearly been laid between natural language and data analysis. The preparation of LOPA's, the vocabulary, and formal definitions for the data of an enterprise seem to us to have great potential benefit for data analysis, data base administration, query languages and certain desirable office automation/information retrieval facilities.

A relation has also been laid between formal data element definition and Information modelling techniques, where a data element is considered to represent an Attribute which has an association with an Entity.

As Sowa has said in the context of a discussion of the basic structures for a natural language query system: 'Since few database designers are trained linguists, a practical system would have to be primed with a basic set of concepts for English words, a set of conceptual relations for linguistic cases and mathematical relations, and a set of tools and questionnaires for automating the task of defining conceptual graphs' (conceptual graphs are roughly speaking Sowa's equivalent of formal definitions). 'Much work remains to be done before the definition of a language can be reduced to filling out a questionnaire'. We believe that the approach of drawing up LOPA's is a practical step towards Sowa's questionnaire.

Returning finally to Crawford, the quality of the 'know-what' is catching up with the 'know-how'.

Acknowledgements

This paper could not have been written but for two essential factors, firstly the original inspiration of E. M. Dozy and W. C. M. Swanenburg which convinced us that a method could be developed, and secondly the painstaking effort and enthusiasm of many people who contributed to working out all the details of the classification system, LOPA's, Vocabulary, etc.

Particular thanks are due to C. Baijens, W. v. d. Bragt, D. Hemelrijk, W. de Jong, H. Vollinga and H. Vrieze with for their enthusiastic contributions over a long period. We would also like to thank J. v. Griethuizen and F. Knoet for their valuable comments on this paper during its preparation.
Applications

Computer tools and techniques required for system for a sound introduction to the work of systems in systems analysis. Part 1, entitled Principles, appendix illustrating the use of documentation which is valuable although more use could the text. The chapter in Part 2 dealing with a list of references which are effectively used in fact finding and user participation in rather

D. MILLINGTON
Systems Analysis and Design for Computer Applications

The stated aim of the book is to meet the need for a sound introduction to the work of systems development with an intended readership including all students of computer science, business studies, management courses and computer programmers considering a career in systems analysis. Part 1, entitled Principles, is a 'general review of the tasks and nature of systems analysis'. Part 2 deals with a range of tools and techniques required for system development, and covers some topics such as fact finding and user participation in rather more detail than Part 1.

The book gives a useful introduction to conventional and established approaches to systems analysis. Each chapter has a useful list of references which are effectively used in the text. The chapter in Part 2 dealing with a hierarchically structured charting technique is illustrated by reference to an example system which is valuable although more use could have been made of this idea perhaps as an appendix illustrating the use of documentation at all stages of the project life cycle.

A major criticism of this, as of most books on system development, is that the section on planning assumes the linear development strategy and makes no mention of any alternatives such as the prototype strategy. Furthermore, while it is always a difficult balance to strike in an introductory work, I think that this book is a little short on the newer approaches to this subject (many of which are well established in other fields where development is important) such as quality assurance in its widest sense.

There is no glossary of terms, which one would normally expect in an introductory book, although to be fair the author takes care to explain terms when they occur in the text. On balance I believe that this book achieves its stated purpose but at £16.50 it is rather expensive.

J. BRITTAN
Chertsey

JAMES MARTIN
Telematic Society

I found this a book full of facts, figures and lists of possibilities/alternatives but lacking any interesting anecdotes and no attempt to place the technological data in any personal, historical or philosophical perspective. If we are to be able to choose whether we want a Telematic Society or not, then the options and the consequences have to be presented in a more attractive way than this indigestible lump. Incidentally we are informed, on the dust cover, that the book is 'mandatory reading for all Civil Servants' working with the Thatcher Government—serves them right.

The theme of the book is that cheap energy is running out and the way that we might prevail (let alone endure) is to create a 'wired society' in which we can all (and that seems to include computers) play, get educated, earn a living, make a fortune, etc. via links through television and satellites—all this without moving out of our houses. Just imagine how wonderful it would be—10 tons of sand in the living room, turn on the sun lamps and the holographic wall display and you're on holiday by the Mediterranean. And there is no pollution—nobody can afford to actually travel there anymore.

The book has no such ironic imagination. Another example—we are told that cars are obsolescent (which seems true enough) and that we will be able to work shorter, more efficient hours at home (via remote visual links) and retire earlier. Yet we are told that one use of our home-based visual link will be to start a second-hand car and one thing we might buy with our early retirement/redundancy payment is a motel. Mr Martin, like computers, deals with facts not contradictions.

However, back to our Civil Servants and their advisors. In the New Scientist (1 April 1982) there was a report that the Japanese intend to launch a manned communications satellite to replace their earth-bound telephone network—this was described as a blow particularly to the British (civil-service approved) cable network proposal—'Once again the Japanese have leapfrogged the rest of the world' said an expert on the Cabinet's Advisory Panel (who has probably read the book as well). However, does this expert really believe that the Japanese are going to put all their communication eggs in one vulnerable basket? 'I'm sorry sir, we can't ask Japan to surrender because we've just destroyed all their satellites.'

A. G. BELL
Sheffield

P. CALINGAERT
Operating System Elements—A User Perspective

The author emphasizes resource management as the central function, and takes pains to distinguish management policies from the mechanisms used to implement them. The focus is on principles rather than on example systems, the approach is descriptive rather than analytic and practical rather than formal. This book is designed for third and fourth year undergraduates who wish to know what an operating system does rather than how to design one. Any other reader who knows something of hardware structure and is familiar with queues and stacks, and with reading ALGOL, will find this book helpful. Queueing theory is left entirely to the short description of books for further study that follows each chapter. These cover European and American sources up to 1980. Most chapters are followed by a set of exercises to stimulate deeper thought on the material. The book does not cover computer networks because communication subsystems need to be studied first. Their influence on control programs makes this essential.

The book is well produced and clearly laid out and is strongly recommended. Unfortunately its price will tend to discourage students from purchasing their own copy.

P. GILES
Stirling

C. R. SYMONS AND P. TIJSMAN

REFERENCES


Received October 1980

Book Reviews

Continued from p. 492