Some Thoughts About a Metadata Management System

Jean-Pierre Kent and Maarten Schuerhoff - Statistics Netherlands

1. What is Meta-Information ?

In the past quarter century the use of computers has radically changed the data management landscape. Both the volume of collected and analysed data and the speed with which these operations are performed have risen dramatically. New dissemination media, such as CD-ROM and the Internet, have also contributed to making data available to a broader public.

Under the challenge set by the new technical possibilities, the concept of metadata and meta-information has grown in importance in a spectacular way. Congresses and work groups on various aspects of data processing pay attention to it, and it has become the subject of conferences of its own. Software for data management refers to metadata, and there is software specifically designed for metadata management.

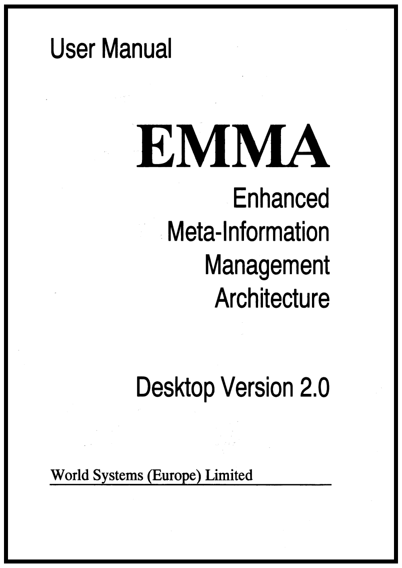
It is however increasingly difficult to formulate a definition that covers all the different aspects of the concept. It often happens that two specialists involved in a conversation about meta-information have difficulty in understanding each other because of different backgrounds and different aims. We ourselves happen to be the victims of this phenomenon, both inside Statistics Netherlands and outside. The only sure way to avoid misunderstandings is to start every dialogue about meta-information with a series of agreements about basic concepts.

In the first part of this paper we will try to identify the semantic area covered by the words *metadata* and *meta-information* in most contexts. We will not attempt a distinction between the two. In this exploratory phase we will use them as loose equivalents.

1.1 Definition of metadata

The words metadata and meta-information were coined on the model of *meta-philosophy* (the philosophy of philosophy), *meta-language* (a language to talk about language), and the such, in which the prefix *meta-* expresses reflexive application of a concept to itself. This construction is still productive, as shown by the recent words *meta-network* (a definition of the Internet) and *meta-search engine* (a search engine devised to search search engines). This provides the simplest, but most widely accepted definition of these words : *metadata* are data about data, and *meta-information* is information about information.

Beyond this thin definition, however, it is often difficult to agree upon what



User Manual

EMMA

Enhanced
Meta-Information
Management
Architecture

Desktop Version 2.0

World Systems (Europe) Limited

information about information should be captured and managed, and how to use it.

1.2 EMMA vs. Blaise

One of the ways to discover differences in meaning is to confront packages that explicitly state metadata management as one of their functions. This is the case of both EMMA and Blaise.

The letters of EMMA stand for **E**nhanced **M**eta-Information **M**anagement **A**rchitecture. The package is developed in Luxembourg by World Systems Europe Ltd. We will refer here to EMMA Desktop version 2.0.

EMMA provides a data base framework for defining and managing meta-information. It approaches meta-information from a hierarchical point of view and distinguishes three levels :

- *context* meta-information, such as special information, subject matter concepts, language, nomenclature, etc.,

- *resource* meta-information, such as documents, files, authors, etc.,
- *content* meta-information, including surveys, populations, variables and measurement units.

EMMA is a data base management system targeted at meta-information, and supports capturing meta-information, searching and browsing of meta-information, and the production of documentation texts. EMMA is specifically targeted at the management of metadata. It treats metadata in the same way as data are normally treated by a data base management system. Metadata are the data of EMMA.

blaise
A SURVEY PROCESSING SYSTEM

Blaise is a metadata-driven integrated survey management system produced by Statistics Netherlands. We will refer here to Blaise III, version 1.12.

Defining data models is one of the main tasks one can perform with Blaise. The system offers a hierarchical approach, in which variables can be clustered in blocks, and blocks can be nested. In Blaise, the metadata consist of all the information that can be specified within the system about the variables and their relationships, including the data type of specific variables, their value ranges, classification categories, with a variety of free format texts that can be used for interview questions, variable labels, value labels, documentation texts, etc.

Here are a few characteristic differences between the two packages :

- EMMA focuses on metadata: beyond the fact that metadata refer by definition to data, EMMA has nothing to do with data. In contrast, Blaise focuses on data. The metadata in Blaise are put to work in order to define data, access and manipulate them, and finally transfer them to other packages.
- Blaise metadata are primarily used by software. The system actively uses meta-information during all processes, such as data entry and editing, interviewing, data manipulation and conversion. It can also convert metadata to the format required by other programs, allowing any third-party package to access whatever meta-information is relevant to it. In contrast, EMMA is a user-oriented metadata management system. All the normal user activities on a data base are supported (entering, correcting, searching, browsing, extracting), and the main output of the system is printed text.

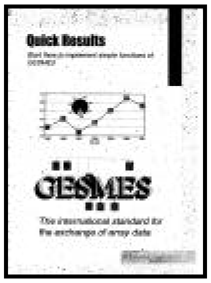
- In EMMA metadata are passive bits of information. They can be stored and retrieved. In Blaise, metadata are active. The user enters the metadata, producing a collection of objects that can be put to work for entering, validating, transforming and transferring data.

- EMMA is open to all sorts of information about data. It allows you to define whatever you need as long as it fits in one of the three categories of context, resource and content metadata. In contrast, Blaise supports only those metadata elements that can be put to work by the system.

This comparison shows how different the two packages are in their view of metadata. Their strong points are very different. EMMA is about defining and standardising concepts, co-ordinating data definitions, and managing the meta-information needed by users. Blaise is about automating the survey process, about managing the metadata needed by the software. EMMA is about concepts, and Blaise is about operations. We would like to establish here our first distinction : there is *conceptual meta-information*, and there is *operational meta-information*.

In our opinion this is not a natural contrast. There is no reason why well standardised and co-ordinated concepts should not become operational, or why operational metadata should not be standardised and co-ordinated. The two should merge, and this should be one of the long-term aims of any metadata development project. We will come back to this idea in the second part of this paper.

1.3 Gesmes



Quick Results

[illegible]

GESMES

The international standard for the exchange of array data

We will now extend our comparison to GESMES, a standard that also refers to metadata in its definition. GESMES is defined by the Statistical Office of the European Communities, Eurostat. It is a standard derived from EDIFACT, and it specifies the format in which statistical data and metadata should be transferred electronically from one institution to another. GESMES tells you how to format the information before transferring it.

Here is an example of a data set cast in the GESMES mould :

```
UNA:+.?'UNB+UNOC:3+STATINSTITUTE+EUROSTAT+950123:1400+
+REF001++GESMES'UNH+001+GESMES:D:95A:UN'NAD+MS+BE001++
+++++BE'DTM+137:950511:101'CTA++:Henri de Bakenbourg'
COM+3222567980:TE'DSI+QPROD123'DTM+Z02:1991119923:708'
ARR++BE:101:FR:911:c11+BE:101:FR:912:c12+BE:101:FR:913
:c13+BE:101:FR:914:c14+BE:101:FR:921:c15+BE:101:FR:922
:c16+BE:101:FR:923:c17+BE:101:ES:911:c21+BE:101:ES:912
:c22+BE:101:ES:913:c23+BE:101:ES:914:c24+BE:101:ES:921
:c25+BE:101:ES:922:c26+BE:101:ES:923:c27+BE:201:FR:911
:C31 etc.IDE+5+DFORMAT123'UNT+12+001'UNZ+1+REF001'
```

Before you can use GESMES three conditions have to be met :

- the statistical concepts must be known to both sender and recipient,
- the order of the dimensions and cells must be fixed and known in advance,
- the coding system used for the statistical values must be known to both sender and recipient.

So GESMES is clearly not meant for modelling data or defining concepts. Most of what is called *metadata* in Blaise must already be known before GESMES can be put to use, and GESMES cannot be used to transfer this type of meta-information. However, GESMES allows you to attach a footnote to an individual cell or row, or to the whole data set. Footnotes are free text, so they can convey any human-readable information. This covers most of the domain covered by the word *metadata* as it is used by EMMA.

The GESMES standard itself is all about data, so it also falls within the scope of metadata. It specifies information that cannot be missed in the process of transmitting data: who is the sender, what is the content of the message, and the specific format in which the content should be transferred. This is not about concepts or about data structures, it is about packing data and metadata in such a way as to identify them to the user and to the software. It is about files, storage media and transfer protocols. Let us call this category of metadata logistic metadata.

1.4 Metadata standards

In comparing Blaise and EMMA we established a distinction based on the intended use of metadata : the point was whether the metadata were meant primarily for use by humans or by software. Human users in this context are mainly people involved in designing, managing and executing the operations leading the data through all the phases of collecting, editing, processing, analysing, publishing and disseminating. This picture misses out the most important user of all : the end-user, the person who uses the data in published form. Whether they browse the book, search their way through the CD-ROM or surf the Net, they must be able to find the data they need, decide whether the data they find are relevant to their questions, and whether the quality of the selected data meets their needs. Finally they must be able to understand the meaning of the data and to use them correctly. We will call the information supporting this type of activities *documentary metadata.* One of the prerequisites for successful communication with the end-user world-wide is international standardisation of metadata. The need for metadata standards has become increasingly acute with the exponential growth of the Internet. The main standard organisations, both national and international, are involved in the definition of metadata standards. For an example of such a standard we will refer to the Federal Geographic Data Committee's "Content Standards for Digital Geo-spatial Metadata", FGDC standard for short. This is a handy example because it is available on the WWW. This makes it easily accessible for our readers, and because it is available in electronic form, it is easy to use parts of the document for quotation purposes.

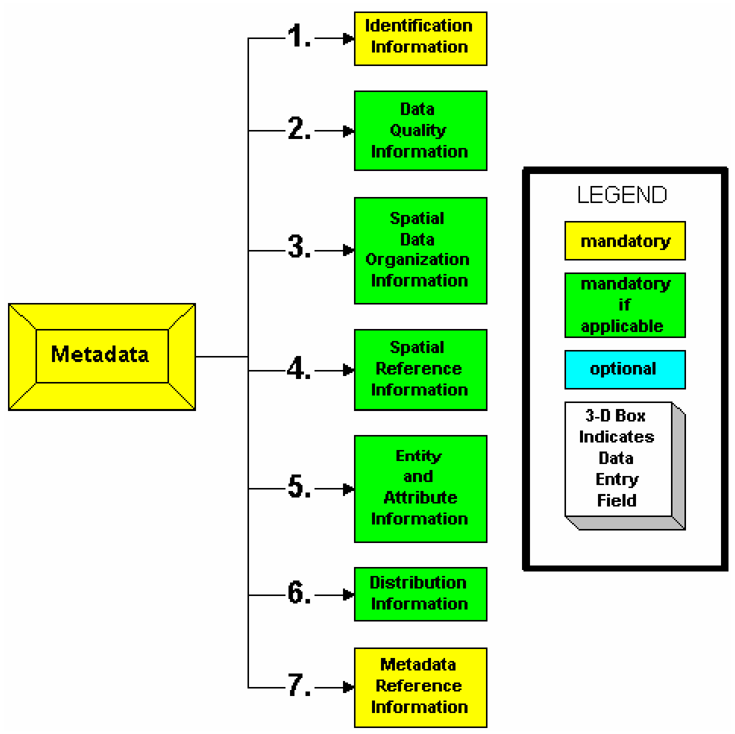

Metadata
1.
Identification Information
2.
Data Quality Information
3.
Spatial Data Organization Information
4.
Spatial Reference Information
5.
Entity and Attribute Information
6.
Distribution Information
7.
Metadata Reference Information
LEGEND
mandatory
mandatory if applicable
optional
3-D Box Indicates Data Entry Field

Figure 1 : Content Standards for Digital Geo-spatial Metadata (FGDC)

This figure is a sketch of the structure of a document containing metadata that conforms to the FGDC standard. It can be found on the WWW at http://www.its.nbs.gov/nbs/meta/meta.htm. Each box refers to a similar diagram, which in turn contains a number of linked items. At the lowest level the standard specifies elementary fields, each having a name and a data type. This standard prescribes down to the most specific details the information that has to be supplied in order to identify, access, interpret and use a document. It is, in fact, a data model of the documentary metadata. This is far away from the approach to metadata that we have seen in the previous paragraphs. Blaise and EMMA allow you to specify, manage and use your own definitions, GESMES tells you how to format data belonging to a data model specified elsewhere. Metadata standards, on the other hand, prescribe both the form and the content of the information to be supplied. They provide one specific data model, not a general framework for data models.

They are not tools, but applications. To illustrate what this difference means, one could use EMMA to record and manage the definition of the concepts to be captured according to the FGDC standard, one could use Blaise to build a tool to help in entering and validating documents conforming to the FGDC standard, and one could use GESMES to transfer such documents from one computer to another.

1.5 The whole picture

The areas of metadata covered by the preceding examples do not, of course, have the sharp rectangular edges suggested by figure 2.

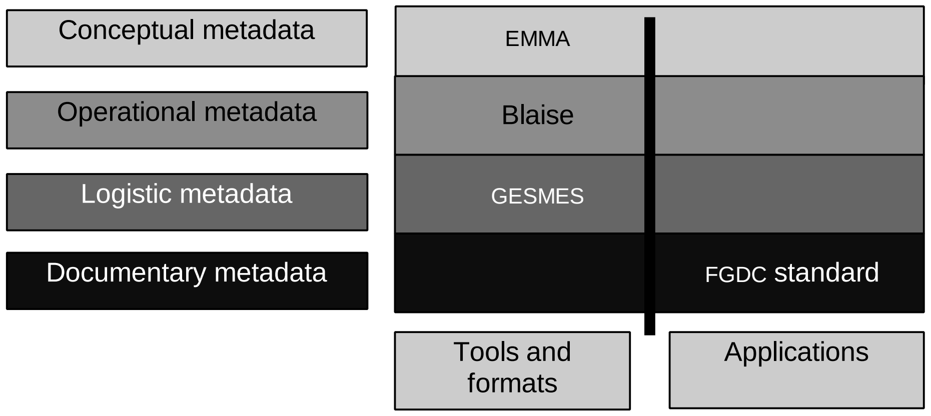


Figure 2 : Semantic area of metadata and meta-information, with a few examples

There is some overlap between them, and wide areas of the metadata concept are not at all covered by any available package. What is missing in this picture is the dynamic aspect: from survey specification to document publication, data and metadata undergo a complex process of specifying, collecting, editing, aggregating and documenting that needs to be analysed to be understood properly, needs to be formalised in order to be managed properly, and eventually needs to be automated. We shall refer to the metadata covering this aspect as *process metadata*.

1.6 Process metadata

In order to illustrate the phenomena that should be controlled by process metadata, let us use the example of a stripped-down survey designed to compute income information. The results have to be published in a table of average income by sex and age class. The size of the population is already known, as well as its distribution over age class and sex. The income information still has to be collected.

This situation implies that a number of concepts have already been provided with a definition, and that both data and metadata are available. Drawing a sample could be one of the first steps of this survey. This process needs to know the population to draw the sample from, and the size of the sample. It produces the sample as its result :

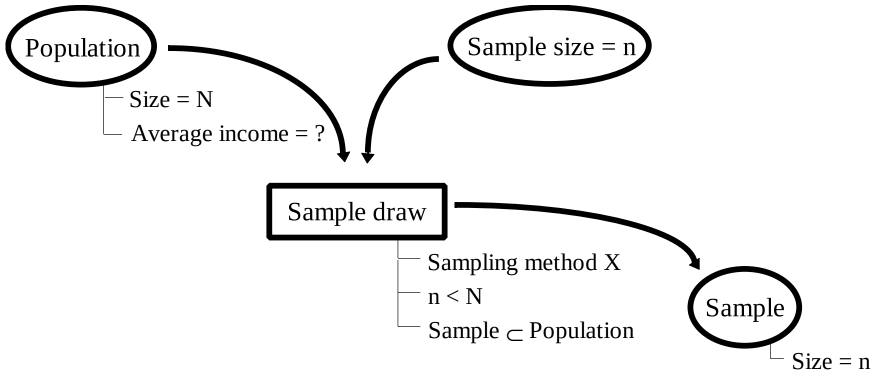


Figure 3 : a sample draw process, defined by its input and output parameters

This diagram represents a process, called Sample draw, which takes two input parameters, Population and Sample size, and produces one output parameter: a Sample of size n. This definition says nothing about the implementation of the process. It could be a manual process performed on a physical card file, an electronic process taking place on computer files, or a mix of the two. The electronic variant could use a relational data base management system or any other type of DBMS, a tailored application, or, again, a mix.

This platform independence of metadata is a very important point. An automated meta-information system must be able to integrate the management of non-automated processes with the management of automated processes. The automated management can trigger a manual process by putting a message on the screen or sending an e-mail message to the person in charge of performing the action. It can then wait for a message telling it the action has taken place. Some processes resist formalisation. The art of carrying them out is passed from old to young, along with the attitude that they cannot be analysed and automated. Such processes will remain manual longer than others, and it is of strategic importance to be able to formalise them in terms of their input and output even if their internal implementation has to wait.

1.7 Modularity of process metadata

An important prerequisite for a concept to become formalised and automated is the possibility of describing it in a modular way. This possibility is present in structured programming languages, where a procedure can be designed as a series of calls to sub-procedures ; in relational databases, where data is analysed in tables through normalisation ; in object oriented design, where an object can be defined as a compound of several smaller objects. It is easy to show that processes, as outlined in the previous paragraph, can also be analysed in smaller processes.

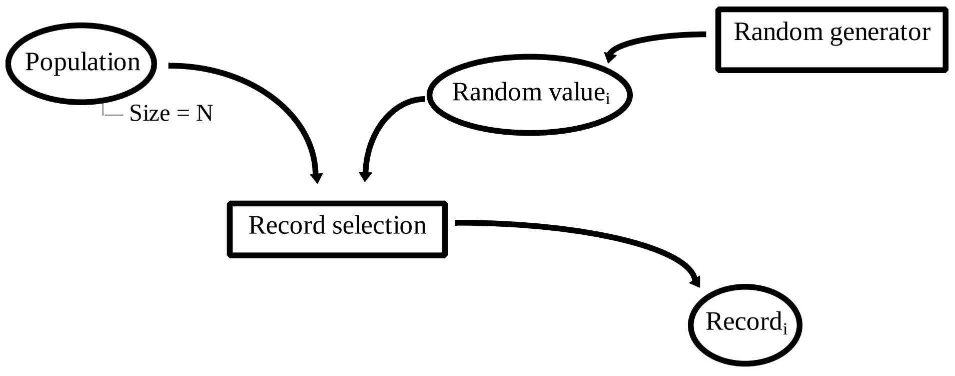


Figure 4 : The record selection process

In the analysis of the sample draw process illustrated in figure 3, one could split the process into n executions of a record selection process. This process would, in turn, need to input a pseudo-random value produced by a generator (see fig 4 and 5).

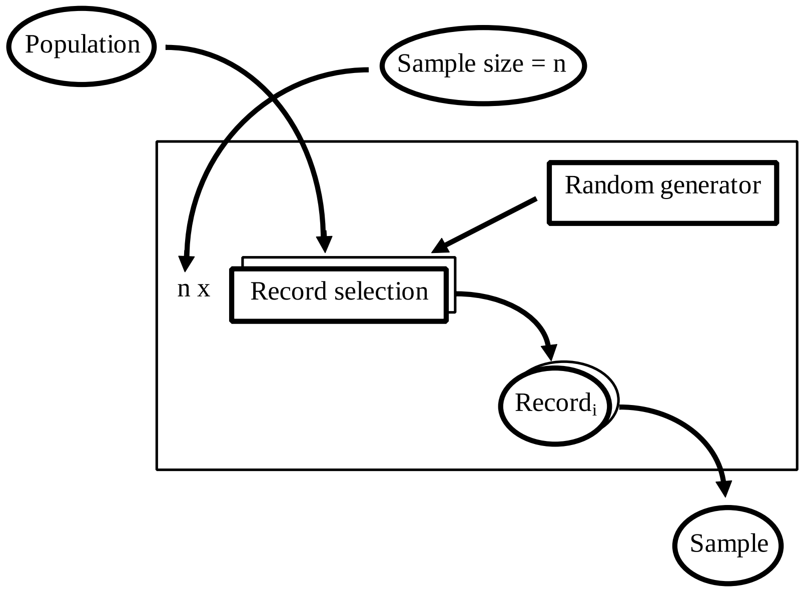


Figure 5 : The sample draw process analysed in n record selections and random generation.

This approach allows us to see a process from two points of view, and concentrate on one of them at a time. Internally, a process has its own structure, which can be designed, understood and maintained without reference to its function within the main process. Externally, a process takes a specific place in the web of sub-processes that contribute to the definition of the whole. While working on one side of the picture, it is safe to forget the other side. At the highest level of the definition, a whole survey can be seen as one compound process.

2. An Integrated Metadata System

In the long term, we wish to have a system that integrates all aspects of metadata (conceptual, operational, logistic and documentary) and covers both data and processes. Specifying the main lines of this system at an abstract level is the first step to take. This general specification can then be used as reference material for short and medium term projects. The second part of this paper states a few points that in our view should guide the general abstract specification of such a long term project.

2.1 A corporate data base ?

Most statistical offices are involved nowadays in the construction of central data bases to manage all their data and metadata. We would like to stress the fact that such a data base is not what we have in mind here. A central data base is an application. It covers the right hand side of the area presented in figure 2. What we are looking at is an integrated system for building, maintaining and using such applications. A tool covering the left hand side of the same area.

2.2 The implementation platform

Considering the high level of abstraction of the system in the first phase of this long term project, implementation aspects should not be taken into consideration. We wish to specify the system independently of the functionality made available by today's data base management systems. We do not know how long the relational model will remain the main standard, whether the object oriented model will take its place, or what implementation platforms will look like in the future. Data base management systems also impose limits on what is feasible. We want to develop our abstract system without taking such limits into account. This should be done in such a way that implementation is possible in any data base management system present or future, or in a general purpose programming language. This platform-independent approach could lead to the conclusion that some parts of the system should use a DBMS, while other parts should be developed in a general purpose programming language.

2.3 Language-based or data-driven ?

As far as specification facilities are concerned, today's systems generally fall into one of two categories: they either offer a specification language allowing to input descriptions and definitions by means of a text file which is processed by a parser, or they offer a series of interactive screens enabling the designer to enter his information by filling in forms. There is, however, no contradiction between the two approaches : a system for managing and retrieving (meta)data can easily have two input channels: a batch-oriented one with a parser, and an interactive one with input forms.

It is an implementation choice whether to supply either or both input methods.

2.4 Active metadata

In the context of operational metadata (1.2), we said metadata should be active. Meta-information should not only be available to the user. It must also actively control the whole statistical process. The following example will illustrate what we mean, and reveal the advantages of such a feature. The table shows journey data in terms of a journey identifier, distance, time and speed.

Journey id	Distance	Time	Speed
	miles	hours	m/s
J1	100	1:00	44.7
J2	45	0:15	80.5
J3	127	1:30	37.8

The speed, of course, is computed from distance and time, but the units of measurement are different. In a language-based system, a formula like the following could be specified :

Speed = (Distance * 1609.29) / (Time * 3600)

There are two problems with this approach : 1) the system does not know what the constants 1609.29 and 3600 are all about, 2) the system does not know why distance has to be divided by time. It is all in the designer's head. Maybe it has been stated somewhere in a formal design document, or in a meta-information data base, but there is no operational link between those specifications and the use of the formula. If, for example, it is decided that the distance will be presented in kilometres instead of miles, it is the responsibility of the designer to change not only the unit label of the Distance column and its data, but also the first constant of the formula for Speed. On the other hand, it would be difficult to support interactive visual definition and maintenance of this computation.

The approach we suggest allows the system to do the job of defining and maintaining this computation. Measurement units and scaling factors must be part of the metadata. The system must be aware of a measurable concept such as Distance, and it must know all about the units that can be used to express a value of a given concept (meters, kilometres, miles, inches, etc.). If told to convert a value from one unit to the other, such as inches to kilometres, the system must be able to access the definition of the conversion formula. This can involve more than just a scaling factor, as is the case for conversion of a temperature value from Fahrenheit to Celsius. Derived concepts such as Speed are defined by reference to two or more other concepts : Speed is defined as the quotient of Distance by Time, and the measurement units of Speed are defined by reference to the measurement units of Distance and Time. Finally, each numeric field is specified by reference to a unit belonging to a predefined measurable concept.

Now it is possible to replace the previous formula by a much simpler one :

Speed = Distance / Time

The system knows all about the relationship between the three concepts referred to by these fields, so it can do the following things :

- check that the quotient of values in Distance and Time units yields a value in a Speed unit: this can lead the system to reject a formula that is not consistent with the available unit definitions,
- select the scaling factors to apply to the computation, in relation to the units used in the three fields,
- automatically convert the values of a column if the unit scale is changed,
- automatically adapt the scaling factors in the computation if the unit scale is changed in one of the columns.

This also makes it easy to support visual interactive specification of the Speed column. The following scenario becomes possible, supposing there is a table containing the first three columns of our example: Journey id, Distance and Time :

- use the menu to add a column,
- select columns 2 and 3 and drag them onto the new column,
- the system looks up its list of derived concepts for one that is defined in terms of Distance and Time. It finds Speed, which is specified as the quotient of Distance by Time,
- the system opens a list box with all defined units of speed,
- select m/s,
- the system fills the new column with computed values,
- the system looks up the list of labels available as column titles for the Speed concept and presents a list box,
- choose the label Speed.

This example illustrates the fact that the language and interactive specification approaches are not contradictory, and shows that a good operational metadata concept can help simplify work in either case.

2.5 Events

Processes take place at a certain point in time, as soon as predefined conditions are met. Such a condition can simply be the availability of all the required input. Reaching a certain date can also trigger a process. Other possibilities are a human decision to launch the process, a state reached by another process, etc. If process meta-information is to be automated, the system needs to know about events triggering processes. These are an integral part of the metadata.

2.6 From concept to documentation

The same concept can play various roles in different parts of a survey. A publication table, for instance, could be supplied with information referring to the non-response rate and the weighting method used to compensate for it. At this stage, it is documentary meta-information. However, both concepts have been present in an earlier phase : non-response has been captured as refused or unknown information and was aggregated to a non-response rate indicator, and the weighting method was implemented in a process that computed the weights. This has to be captured in operational meta-information. But first of all one needs to have a clear definition of the concepts of non-response, weighting and the available algorithms. This is conceptual meta-information.

The system has to know how to supply the individual cases of non-response to the weighting process, how to merge them into a non-response rate, and how to make this available for documentation. The information about the weighting algorithm must be stored only once, and accessed both by the weighting process and the documentation.

3. System Design

As stated in the first part of this paper, the first step will be an abstract specification of the system to be built. The design will have to meet the following four requirements : implementation independence, unlimited coverage of all aspects of meta-information, unity of definition, and object orientation. In the third part we will now concentrate on these four points and conclude with some examples.

3.1 Implementation independence

We have already mentioned that some processes are implementation-resistant. This is partly due to the fact that they have never yet been analysed formally. It is also a question of culture. There are statistics that are generally recognised to be dependent on the specialist’s know-how, something that “could never, ever be replaced by software”, a handy alibi for the status-quo. So we want to integrate these processes into the system by describing them as black boxes. The system only knows they exist, what input they need, what output they produce, what the relationship is between input and input, and what events trigger them.

Taking the human factor into account in this way implies building a system that runs on two platforms at a time: the computer and the human brain.

Another argument for implementation independence is the fact that some aspects of metadata management already have been implemented in systems that are available on the market. EMMA is one of them. The domain of metadata is so vast that it is not possible to cover it all with one universal package. However, by integrating third-party software into the system we can easily extend its functionality to what is already available. Blaise III has shown that this approach is very powerful. Blaise does not carry out tasks that are made available by other packages. One example is statistical analysis. Quite a few good packages do the job, SPSS is but an example. A Blaise application can offer statistical analysis by integrating SPSS. For instance, there can be a call to SPSS in the menu, which triggers a data selection process, creation of an SPSS set-up on the fly, after which control is transferred to SPSS to create the system file and interact with the user directly. When the user exits SPSS, the application goes on to whatever has been defined as the next step. Any third-party software can be used in this manner to process data managed by the Blaise system. This works for data, and we are convinced that it will also work for metadata. So our choice of platforms is not limited to data base management systems and general purpose programming languages: we also have third-party software to choose from.

Our own software is also a candidate for integration in the metadata system. In the case of Blaise, this could happen in various ways. The system could transfer operational metadata to Blaise and call upon the data management functionality of Blaise for such things as interviewing and data editing. Or we might end up building such a general tool for designing metadata management subsystems that Blaise could be rewritten in terms of the new system. Alternatively, the functionality of Blaise could be extended in such a way that Blaise itself would become the general metadata management system. We do not know yet. And we are not in a hurry to decide.

Presently we are witnessing a rapid evolution of the concept of inter-program communication. In the past, programs could communicate through ASCII files ;later came the dBase format; today, they access each other's data through ODBC drivers, or even communicate with each other in real time through OLE. Software is no longer by definition an executable program. There are a host of formats for software that does not run autonomously : there is DLL, VBX and OCX, and there are more to come. The day will come when those formats will be preferred to the heavy monolithical executables of today. The end-user will build his own private application by clicking diverse modules into place. A word processing module from Microsoft, a spelling checker from WordPerfect, a data management module from Borland, and maybe a metadata management module from Statistics Netherlands. So the system could be implemented as a loose collection of modules selectable by the user.

The system we have in mind might very well never be implemented completely. The first prototypes will be limited subsystems, such as a module for classification definition and maintenance or a tool for co-ordinating and standardising the legacy meta-information scattered over all the surveys and publications of an institution. The implementation of such loosely related applications could vary greatly, and this is another reason to wish to keep implementation aspects out of the general design.

Finally, the implementation landscape is so dynamic that switching from one implementation form to another between releases needs to be an easy and fast operation. This is not possible if implementation aspects are part of the design.

3.2 Unlimited coverage of all aspects

It is not easy to give a well delimited definition of what metadata will be considered relevant, and what should be left out of the system. Metadata are used to manage data. What do we use to manage metadata ? If they are to be operational, they need to be described, they need to fit their own model. The metadata of the metadata. The meta-metadata. This is also meta-information. Here are a couple of examples that will show that this is also relevant to our system.

The process of data editing has two objects : its first function is to identify and correct errors in the data set. The input of the editing process is a dirty data set, and the output is a clean (or less dirty) data set. The user of the data set considers this to be the primary function of data editing. But the designer has something else in mind : systematic errors can be symptomatic of design mistakes that have to be corrected before the same data are collected again. So the designer also wants data editing to produce an edit trail. This edit trail (which belongs to the output of the data editing process) will be used as input by the maintenance process. The output of the maintenance process is a new data entry process that produces better data and needs less editing. The new entry process cannot take place before the maintenance process has executed.

In this example, the survey design process has become an integral part of the designed survey process. There is no a-priori reason to exclude this type of reflexivity. This, of course, does not mean that we have the ambition of automating every single sub-process that we identify. But we want to see the whole picture before we start implementing some parts of it, and we do not want to exclude any relevant aspect because it might turn out to be overkill.

We would like to draw our next example from the problem of multi-lingualism. Some countries have two or more official languages, and maybe also one or more widely used non-official ones. In such cases good communication requires the use of more than one language, and sometimes the use of multilingual documents is imposed by law. This applies to every bit of natural-language meta-information : question texts of an interview, titles and footnotes of a publication table, etc. This poses a heavy maintenance problem if the task of warning all translators is the responsibility of any one entering new texts or updating old ones, or if it is the responsibility of translators to check whether there are new or updated meta-information texts. The whole operation of maintaining texts in multiple languages is much easier if the system is designed to keep multilingual texts synchronised by sending a warning to translators at every change of the original set of texts. Here again, we have something that is marginal to the survey process and belongs to the design process, something, nevertheless, that we do not wish to exclude from the possible functionality of our future system.

Another point that falls under this paragraph is the range of applications of a meta-information system. We want to be able to use it for all activities involving data and/or metadata. It must be able to control the co-ordination and standardisation of concepts, to run and monitor processes, to manage resources, to produce documentation, etc.

3.3 Unity of definition

A concept can be present at many different points of a project. Most concepts have a definition (conceptual meta-information), a set of operations (operational meta-information), and user-oriented documentation (documentary meta-information). We do not want these different aspects of one concept to be scattered all over the system. One must be able to access all aspects of a given concept through the unique object representing it. The rule is : one concept, one polyvalent definition.

3.4 Object orientation

In this paper we have expressed the idea that meta-information should not only be stored and retrieved, managed and maintained, but that it should have tasks to perform and know how and when to perform them. In the previous section we have also demanded that every concept should concentrate all elements of its definition. This combination of data and behaviour is a typical result of an object oriented analysis. It is our conviction that object orientation can be a significant contribution to a powerful and intuitive metadata management system. This does not mean that we intend to use an object oriented language or object oriented data base management system: these implementation decisions will have to be taken later. By an object oriented system we mean a system in which resources, data and processes are represented as objects combining information and behaviour.

The importance given here to objects may come as a surprise to the reader : object orientation is often seen as a programming style, an attitude towards implementation techniques, something that does not concern the user. It is also often considered to be something hard that needs a considerable time investment in order to be applied successfully. We do not agree with this point of view. Object orientation is first of all a design attitude, which in our view contributes to better control of system complexity. It is the next logical step after structured design. A system whose users are involved in designing surveys should help them to apply the most efficient design methodology.

By managing all information and behaviour of one concept, a unique object representing that concept is a contribution to the overall unity and coherence of the system. In this view of things there are no barriers between conceptual, operational and documentary metadata.

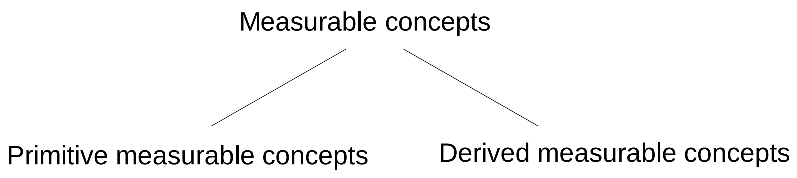
What is the behaviour of a metadata object ? It is anything the object can do to supply information or to contribute to one or more processes. An object should be able to supply its name or identifying key, its definition, a list of all the objects it is related to ; it should be able to write itself in Pascal or C code or in an SPSS set-up, to store itself in a relational data base, detect that its documentation in its original English form is more recent than its French translation and trigger a translation process (in which the human black box will be involved, of course), perform computations and selections, take decisions, and carry out any other relevant tasks.

3.5 Examples

To illustrate these ideas, let us follow the life cycle of the concept of Speed as represented by an object in the system. We will refer to the table represented in section 2.4.

Before we can use Speed in a survey, we need a definition of the concept of speed. This definition must be common to all surveys : we want to be able to relate speed information over surveys. So we need an object to represent the concept of speed. Speed is a concept that can be measured, it can have values expressed in units belonging exclusively to speed. The system offers a predefined class of objects, the Measurable Concepts, which is designed to model things like distances, times and speeds.

There are two sorts of measurable concepts: primitives, like distance and time, which are defined independently of any other measurable concept, and derived, which are defined in terms of one or more other measurable concepts. The two have in common the possibility of defining and using a list of units in which to express values. The different units belonging to one concept can be interrelated in order to provide both their definition and their conversion formulae.



Primitive and derived measurable concepts can be defined in terms of what they share and in terms of their differences. We model them with two different classes of objects, but when we are interested in their common features, we can pretend they belong to the same class and forget their differences. They are interrelated in the following way:

Any Primitive measurable concept is a Measurable concept, and so is any Derived measurable concept. The relationship, however, does not hold the other way round. The class of Measurable concepts defines all features common to both: a natural-language definition, a set of units, and an operational definition of the relationship between the available units.

Derived measurable concepts add to this a list of related measurable concepts and an operational definition linking them in a formula. In the case of Speed, the operational definition is Distance/Time. The Distance object, which, let us assume, has already been defined, provides the following units: *mile, yard, inch, km, m,* and *cm*. One of them, e.g. *m*, is defined as the primary unit. All others are defined by reference to the primary unit and a multiplication factor. The Time object is very similar. Its primary unit could be chosen as being the *hour*, which can be expressed in *minutes, seconds, days* and *weeks* (months and years address the issue of variable scaling factors, which we will not discuss here). Theoretically all combinations of Distance/Time units could yield a Speed unit. But in practice only a few are used. Only three of them will be taken in the definition of Speed : *mph* (miles/hours), *kmh* (km/hours) and *m/s* (m/seconds). Note that the symbol of a derived unit is not always identical to its operational definition : see *mph* vs *m/s*. A derived measurable concept like Speed has no primary unit : all units are derived from those of the concepts referred to.

In order to collect and publish the information shown in the table of section 2.4, we need to define a data model containing fields for a journey identifier, and for Distance, Time and Speed. The mechanism for defining fields is the same as the one for measurable concepts : the system provides a predefined class to model field objects. For the definition of the field object Distance, there is no need to rewrite or copy the definition of the concept object Distance and its units : the Field class provides a reference to a concept for its definition, and knows how to access the functionality of its concept. If the Distance field object is asked to convert its value from miles to kilometres, it will pass the request over to the Distance concept object along with the value to be converted. This mechanism, by which an object draws upon the functionality of another object, is called delegation, and is one of the most powerful features of object orientation. Note that the Measurable concepts have no values : they have only definitions and operations. Field objects have values, and they can use the Concept objects they refer to in order to perform the required operations.

The functionality of the Speed object is a good illustration of the power of the delegation concept. This is what allows us to design the Speed column by dragging the other two onto it (see 2.4). Adding a column to the table creates a new field, but without any definition. Dropping the other two fields onto it triggers the automatic definition process, which works as follows : the new field asks the other two to supply information about what they mean together. They send the question back to their respective concepts. The two find out that they only collaborate in the definition of Speed. A reference to the Speed concept is sent back to the new field. The field inserts the reference to the Speed concept in its definition. From now on this is the Speed field. It asks the Speed concept for a list of available units and asks the user to choose. It passes the user's choice, *m/s*, back to the Speed concept and asks it to return the Distance and Time units that support *m/s*. The Speed concept returns *m* for Distance and *seconds* for Time. The Speed field asks the Distance field to supply its value in *m*. Because the Distance field knows it is using the *miles* unit, it sends its value to the Distance concept and asks it to convert from *miles* to *m*. On receiving the result it passes it back to the Speed field. The same happens to the Time value, which the Time field will send to the Speed field after having it converted to seconds by the Time concept. Having the two required values, the Speed field can send them to the Speed concept and request the computation of a Speed value. It is the responsibility of the Speed concept to perform the division and send back the result.

At first view this looks complicated, and it would be if it had to be implemented in a procedural way. Objects, however, allow the user to define this interaction as a series of messages and responses exchanged by the objects. Most of this behaviour can be implemented in the abstract objects available in the system. So the preceding scenario turns out to be quite simple to design. When the developer comes to defining specific measurable concepts and their units, the abstract classes defining measurable concepts and their features, including their interaction with field objects, are already present in the system, with all their functionality. Fields and concepts already know how to communicate with each other. All the designer has to do is supply definitions for new concepts. The task of defining Distance, Time and Speed will take only a few minutes, after which the objects are operational to define Distance, Time and Speed fields throughout the whole survey, or rather for all surveys.

What we want to offer our users is all the advantages of object orientation without the burden of object oriented theory and rules. Blaise has shown that this was possible for structured design: Blaise offers a framework in which developers work in a structured way without ever having learned to do so the hard way. One of our users told me one day about a software specialist who was looking at his work and said he was doing structured programming: being told such a thing made him feel like Mr Jourdain who had been producing prose for his whole life without knowing so. We are convinced that this can work for object oriented design too. The only way to prove it is to build the system and look what happens. It is a bet, but we are confident that the odds are on our side.

Notice the definition of fields by reference to concepts such as Distance and Time, and the definition of the concepts themselves are two different design sub-processes. Keeping them separate promotes both modularity and standardisation : it makes the definition of concepts independent of their use. This approach enables to separate tasks that are normally performed by different persons: defining concepts and maintaining classifications is usually not the responsibility of people in charge of defining and maintaining questionnaires or publishing tables. By integrating those tasks in different sub-processes we ensure that they do not interfere with each other.

Conversion of Age to Age class will offer another example of the possible contribution of object orientation to making metadata more readily available both to software and to the user. Age information is usually present in microdata in the form of an age field. In cross tables, age information is often present in the form of age classes defining the rows or the columns. This could be modelled by the definition of an Age concept and an Age class concept, which are interrelated. The Age class concept should know what the relationship is between an Age value and an Age class value. It would be expected to know the answer to questions such as : what age class does the age value 45 belong to ? Is the age value 45 greater than (all the values belonging to) the age class 31-40 ? What is the highest age value belonging to age class 61-69? In other design methodologies, where processes are defined far away from the data, such questions would be implemented in separate functions. An object oriented system will allow to manage all these aspects as part of one object : the Age class object.

4. Conclusion

In this paper we have tried to present some very general ideas about metadata. The aim is not yet to provide solutions, but to define the field for further research. It has become clear that the concept of metadata is very diverse, and that a general approach is needed in order to provide generally applicable solutions.

We have learned some basic principles about metadata that will help us in further research :

- Meta-information is interpreted in almost as many ways as there are implementations of meta-information concepts.
- In order to talk about a system covering all meta-information aspects, one must observe meta-information from a broad perspective.
- Meta-information should be defined in such a way as to become operational. That is : its usage should not be limited to one task, say documentation, but it should remain open for implementing other tasks, even some that may not yet be known at design time.
- Design principles such as object orientation should play an important part in the functionality of a meta-information system.
- The ultimate meta-information system will not be one system covering all meta-information aspects, but a loose combination of other systems and tools, sharing the same meta-information principles.

Guided by these principles we will go on searching for ways to handle metadata that will enable us to compare and, more importantly, combine existing metadata tools. Once we reach a metadata concept of a sufficient level of generality and abstraction, we can start designing a system that will contribute to easing metadata management.