Data Management

Jens Andresen
Torsten Madsen

IDEA – the Integrated Database for Excavation Analysis

1 Introduction

Archaeology has many diverse appearances. It can degrade to a mechanical manipulation of artefacts with seemingly no theoretical foundation, or it can escalate into wild theoretical digressions with virtually no reference to the archaeological record. One reason for this variability in archaeology is the almost paradoxical epistemological conditions of the discipline: on the one hand the subject matter of contemporary, physical objects, and on the other hand the aim to create a mental construction called the past (Madsen 1994: 31).

In the study of artefacts, during archaeological survey, and last but not least during archaeological excavation, the theories of the researcher are confronted with observations of the real world, and it is through this process that new knowledge is created. This is the reason why archaeological excavations *do not* constitute a mechanical unearthing and subsequent recording of objective facts, an opinion not uncommonly stated in archaeology.

Because of the destructive nature of archaeological excavations, attention is frequently focused on the recording of this activity. It has been customary in recent years to stress the subjective nature of excavation records, an attitude we fully share. However, in their eagerness to point out the biased nature of archaeological doings the critics tend to forget the status of excavation records in archaeology. Although coloured and filtered, these recordings are statements of observations of the real world: one may deliberately select and unwillingly overlook, but one cannot (by the code of the discipline) record something not seen. This is the reason why records from archaeological excavations are to be treated as historical documents and why it makes sense to establish a structured archive of recordings from archaeological excavations.

Still, a major problem in archaeology is to master the inherent complexity, diversity, and quantity of archaeological data. No wonder computer based recording systems for archaeological excavations have been and are continuously being created all over the world. Far the major part of these are either fairly simple flat file systems or hierarchically organised systems (Arroyo-Bishop/Zarzosa 1992; Rains 1995). Flat files and hierarchies, however, are too simple structures to provide a general basis for the description of archaeological reality in all its complexity. As a rule more or less well adjusted *ad hoc* solutions to particular situations become the result (Madsen 1993).

In the late eighties we began to discuss the potentials of relational data modelling applied to archaeological excavation recording, and at the CAA in Southampton in 1990 we presented a paper on the structure of information from archaeological excavations viewed in a relational framework. For various reasons the paper was only published two years later in another context (Andresen/Madsen 1992). Initial attempts to acquire money for realising our IDEA were not successful either (Madsen 1994: 27), but in late 1993, we finally succeeded. A three year project funded by the Danish National Research Foundation for the Humanities was established (Madsen/Andresen 1993).

The purpose of the project is to create a general system for recording, analysis and presentation of information from archaeological excavations. The system is intended to serve as an archive of recordings from all types of archaeological excavations, and it shall be able to automate the production of the archival report as a paper-copy. In a next phase of development, not covered by the current funding, the system should develop into an analytical tool for the processing of excavation information.

The core of the system is implemented in a Relational Data Base Management System (RDBMS), naturally. We have chosen Microsoft Access for this purpose because it combines sophistication as a RDBMS with a low cost availability within today's standard PC environment. The advantages are amongst others: low learning curves for the users, high potential for integration with other application software, and upgrading security. Furthermore, the National Museum of Denmark has also chosen Microsoft Access as their development tool for their interface to the Danish central SMR. Using identical development tools should help us to overcome some of the technical obstacles in attempts to integrate data from the excavation archive with SMR-recordings — at least on a national scale.

2 The conceptual model

The basic conceptual model for the system has already been presented and discussed in some detail (Andresen/Madsen 1992, 1994). We will not repeat the discussion here, but only summarise the structure and basic entities of the model.

Fundamental to our model is the acknowledgement of three universal entities into which we can categorize all excavation information. These entities we termed Layers, Objects and Constructs (fig. 1).

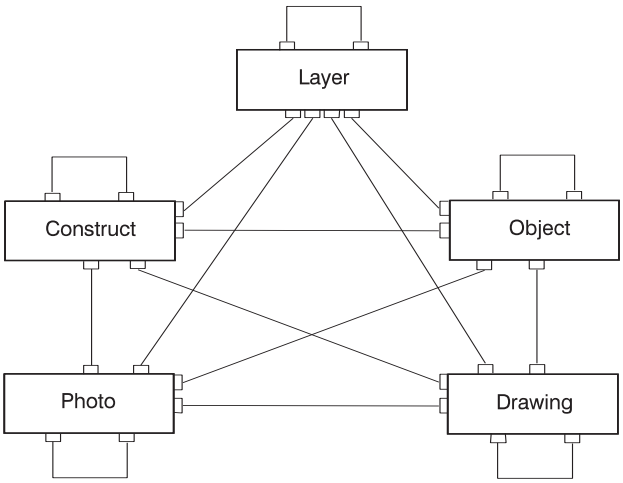

Figure 1. The core of IDEA: five basic entities – Layer, Object, Construct, Drawing and Photo – linked with many-to-many relationships between entities as well as internally between records of the same entity.

A Layer is an entity of deposition separated from its surroundings by its physical and chemical properties. In this sense a layer can be seen as a geological entity, even if often it owns its existence to human activities. Other names for a layer often seen are fill or context (Carver 1985).

Essentially an Object is a subpart of a layer, segregated from this by the actions of the archaeologist. Any part of a layer that an archaeologist sticks a label on, bags, and brings home from the excavation automatically becomes an object, or perhaps as it should be named more correctly, a find or component (Carver 1985).

The Construct is a slightly more controversial entity. In our definition it is any interpretation category that an archaeologist may impose on Layers or Objects, alone or in any combination.[1] In its simplest form an instance of the Construct entity could be something like 'pit', 'post-hole', etc. At the more complex level it could be 'activity area', 'village', 'chronological phase' etc. Traditionally this is what would be classified as cuts, features, structures and beyond (Carver 1985).

In addition to these three universal entities we also defined two documentary entities — Drawings and Photos. These two auxiliary entities are fully interlinked with the archaeological entities. One may question, if it is possible to draw or indeed take photographs of interpretations/Constructs. On the other hand it is customary in archaeological recording systems to refer to interpretation units in the documentary sources, so we have chosen to endorse this practice and allow links between Constructs on the one hand and Drawings and Photos on the other.

This conceptual model may at first glance seem fairly simple, but due to the many-to-many relationships between all five entities and internally within each entity, it is a fairly complex model to implement. Further, there are a number of additional concepts and features that modifies and qualifies the model, adding further complications to the implementation.

One such moderator is the concept of project. In order to be able to handle more than one excavation in the same database, and indeed to be able to handle excavations differently with respect to structure of recordings and classifications of content, we have defined the project as the primary separator of information. Everything within a project is by definition fully comparable. Information from different projects is only compatible and comparable to the extent that the projects share equivalent definitions and classifications of structure and content.

An important, but also a potentially very difficult qualifier to handle is the concept of event. On a fairly simple level it is the ability to record the who and when of a drawing being created by a number of draughtsmen over a period of time at a number of different 'drawing events', or the ability to record information according to a number of different excavation campaigns. On a much more complex level it could be the possibility of recording 'the history' of interpretations of a site. That is, instead of overwriting an interpretative Construct, a substitution takes place with the former Construct being kept as a historical, currently obsolete, piece of information concerning the interpretation. Historical information like this should be hidden, but not forgotten. That is, it should be possible to recall former interpretations on demand.

A full handling of events is truly complicated, and we have decided to take up only the simplest part of the problem relating to different recording events during excavations. Thus, at the moment we are not going to try to solve the problem of event recording in connection with the interpretation of an excavation.

Another very important qualification to the system is the possibility for users to customise the structure of recordings as well as differentiate classifications of their content. In a system where all entities are interlinked, the number of

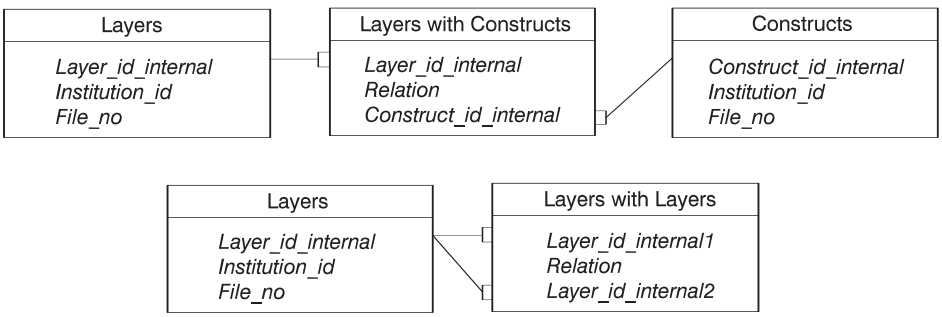

Figure 2. Implementation of many-to-many relationships using link tables. The upper part shows linking between entities, and the lower part linking within entities.

ways to structure the information (by choosing different channels through the web) is high. What particular structure should be used in relation to an excavation is not a programmer's decision. It must be a user decision. The same applies to the classification of items of information, for instance the artefacts. A Stone Age and an Iron Age excavation certainly cannot use the same classification and description system for artefacts. It is the user who should decide the classification system to be used, and the user must be able to tailor the system to fit his or her needs.

3 Implementation of the model

Even though each instance of the database is able to hold data from several excavations — and thus might be maintained by a central authority — the decentralised structure of Danish archaeology (and archaeology in most other countries) demands a decentralised solution. This is one of the reasons why our system is organised into two separate parts: the core data structure in one module and the user-interface in another. This division has several advantages during the phase of software-development. For the end user, one advantage is that by telecommunication channels it will be possible to interface different geographically dispersed data sets. At the same time new versions of the user-interface can be implemented without any side effects on the data part. Indeed, there is the possibility to create multiple user-interfaces to the same set of data, should this be needed.

Another consequence of this architecture is the possibility to separate archaeological from administrative information on the project level. Administrative information dependent on country and institution can be included as tables in the user-interface part. They cannot be separated from their entry forms anyway. Likewise, the structure and layout of reports are also country and institution dependent and thus should be kept entirely in the user-interface module as well.

A disadvantage of this two-level architecture is that uploading of data from one instance of the database to another has to be monitored by a module of specially written code. This we foresee will cause some head-scratching to write. The technical problem is that each entity instance in each instance of the database is given a unique sequential number (key) by the RDMBS. Thus entity instances are very likely to share the same identifiers throughout the various instances of the database. Because these numbers are used as pointers (foreign keys) in the link tables, it is obvious that uploading has to occur as a sequence of multiple, nested transactions in order to maintain integrity of the uploaded database.

The backbone of the implementation consists of five tables corresponding to the five basic entities of our conceptual model. Each of these contain basic information, which subsequently can be tied together using link tables. The link tables are of two kinds (fig. 2). One type consists of link tables interconnecting the five entity tables. There are ten of these, each having the primary keys of each of the two tables they connect as foreign keys, and in addition a field called 'Relation' to hold the type of relation between the two basic tables. Together the three fields of the link table constitute its primary key, and hence a unique entry. The other type of link table is that which connects an entity table to itself. Logically there are five of these, although we have not implemented Photos linked with Photos as we cannot imagine who would use it. This kind of link table is constructed like the former except that both foreign keys refer back to the same primary key in one single table.[2]

The field 'Relation' in link tables between entities is customarily filled in with a 'is linked to' string, but it is currently not used for any purpose, as we have seen no way in which we could make use of different types of relations. Should the need arise, however, the system is prepared for multiple relation types.

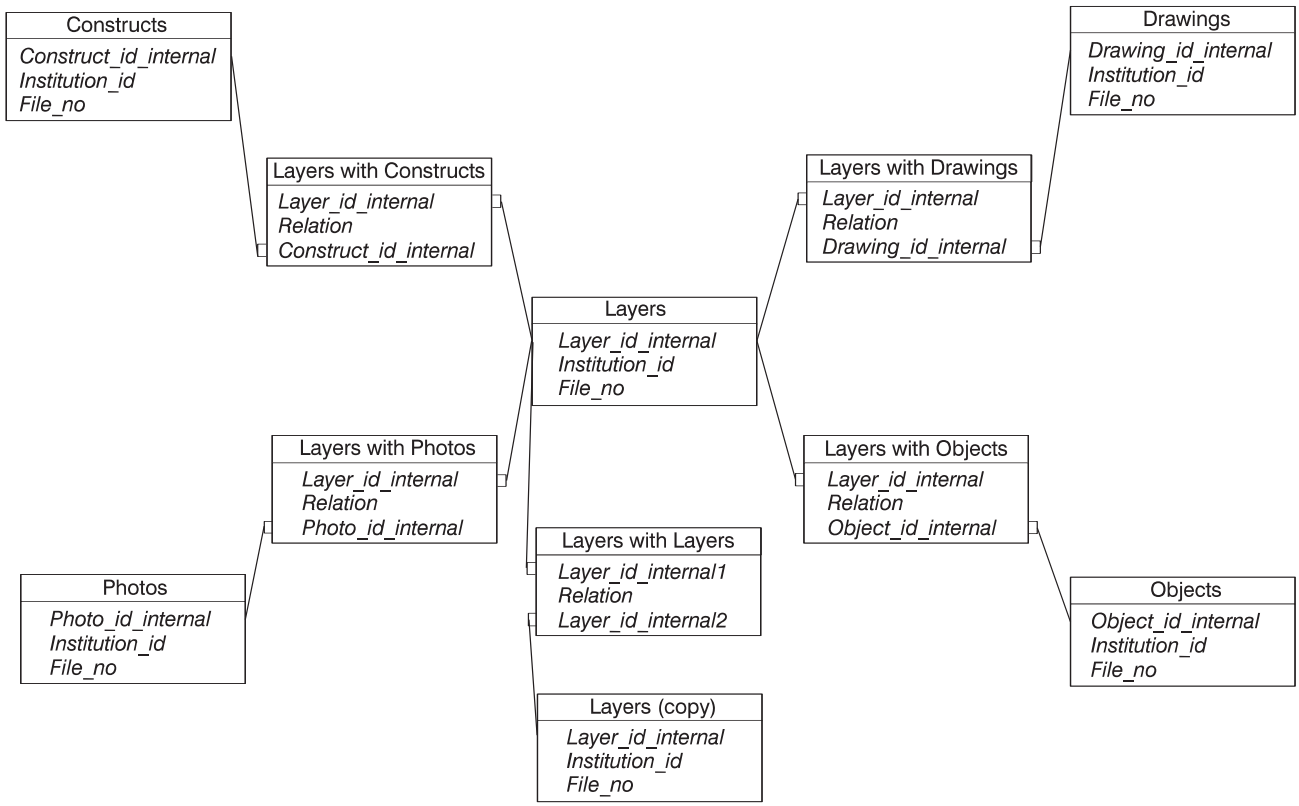

Figure 3. The core of IDEA as seen from one entity (Layers). No other entity is more than one link table away.

In link tables connecting entity tables internally, however, the field holding the relation type in the link table is very useful. Thus in the example shown in figure 2, where Layers are linked to Layers, the field can hold various types of stratigraphic relationships. Had the example been Objects linked with Objects, the relations could have been various types of information on refitting the objects in question.

The implementation of the conceptual model (fig. 1) leads to what looks like a spaghetti junction construction. However, when viewed from any particular entity the structure appears fairly simple. Any entity table is only one link table away, and we have what may be described as a five armed octopus (if that is not too much of a contradiction) (fig. 3). Four of the arms lead through link tables to the four other entities, while the fifth arm leads through a link table to the entity itself.

In figure 3 we view the structure of the system as seen from the Layers table, but a view from any of the other entity tables would look exactly the same. Any of our five basic entities is thus linked directly to the others as well as to itself. It is our claim that in using this structure we can map most if not all data models for archaeological excavations.

One pre-condition in our implementation of the conceptual model, is that any instance of an analytical or documentary entity has to be uniquely identified. Thus if we want to store information about a specific sherd from a bag find, then this sherd has to be identified separately. Because the user-supplied identification does not enter a key-field in the underlying table, the sherd identified need not be renumbered, that is double-numbering of user-supplied identification is allowed! The key-field for any record in the tables is assigned by the RDBMS automatically as a positive long integer unknown and hidden to the user.

In cases of double numbering the problem for the user of course remains as ever: the difficulty to separate instances with identical identification. The obvious solution is, as it has always been, to number every instance uniquely.

Provided that information has been loaded into the basic entity tables, setting links between the entries in these tables is a fairly simple matter. We have chosen to implement the linking through a form where we can pick any number of

items available from the entity you wish to link to, and connect them to the current record of the current entity table. Linking thus always takes its starting point from a specific entity instance, say a layer in the Layer entry form. If we wish to link objects to this layer we call up the linking form by pressing a button at the base of the Layers entry form. This provides us with a form containing two list boxes (fig. 4). The one on the right contains all those finds, if any, already linked to the layer, while the other box contains all recorded finds not linked to the layer. Setting links, or removing already established links is simply a matter of using the arrows between the two list boxes, or by just double clicking the item we want to move.

Figure 4 shows the form used to set links between entries in different entities. As mentioned we do not need to work with different relation types in this case, but if we move to the internal relations (say Layers with Layers) we need to be able to set the type of relation as well. This we do in a form much resembling the one in figure 4, but with the addition of a combo box, where we can choose the type of relation we wish to view and set.

The implementation of the 'project' concept is rather simple. We have created two tables, *Institution* and *File*. An institution may be an archaeological unit, a university, a museum etc. Each institution may house many excavation-projects, each uniquely numbered in the *File*-table. The key-fields of the two tables are included in the five primary tables of the core database and in some auxiliary tables which hold other relevant information.

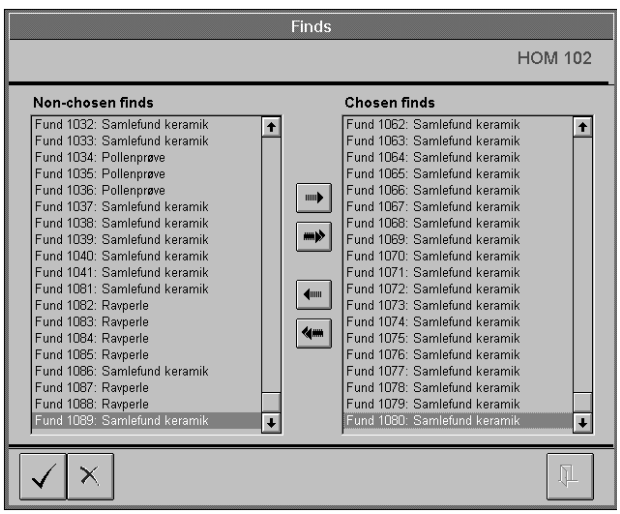

Figure 4. The form used to set relationships between the current record (could in this case be a construct, layer, drawing or photo) and any number of available objects. The list box marked Chosen finds contains those finds already linked to the current record, while those listed in the Non-chosen list box are those finds still available for linking.

Further subdivisions within a project may be implemented utilising numbering conventions in the user-supplied identifiers. I.e. area codes and calendar year could be used as suffixes to the identifier of the entity instances. Therefore we can foresee a future demand for a customisation module to take care of project subdivisions and numbering conventions.

4 Customising the system

4.1 Implementing a datamodel

At the bottom of each of the entity entry forms (fig. 5) a number of buttons with arrows across opens up link forms to other entities of the type shown in figure 4. However, if we allow users to link as they please we seriously risk that the entries entered will cause inconsistency in the database. Very different structuring of the information can result from unconstrained linking of entity instances.

One way of controlling input is to disallow users to input data to a particular entity, say Objects, unless it is controlled by another entity, say Layer. That is, the only way you will be able to enter Object data is through an input form activated by the Objects button on the Layer entity form. In this case you will not get a link form in response to a press of the button, but an input form setting a pre-defined link as part of the data entry process. At the

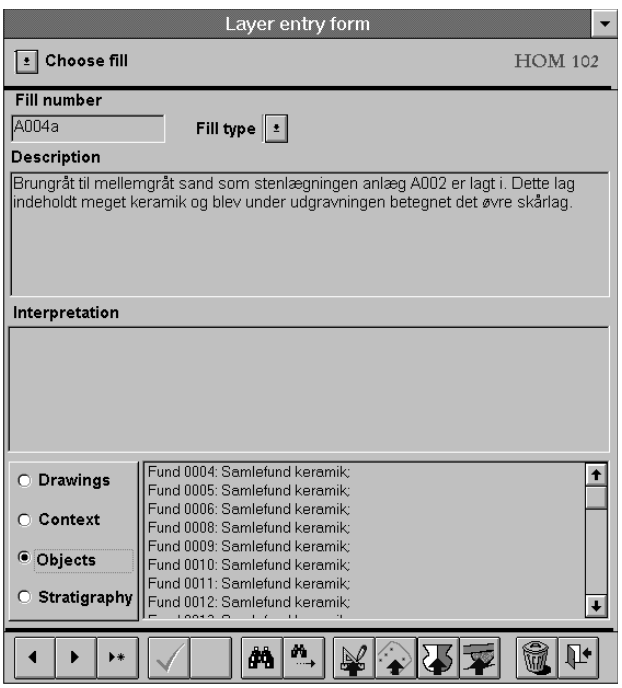

Figure 5. A standard entry form for a basic entity in IDEA (here the entity Layer). Buttons with arrows across call up either link forms of the type shown in figure 4, or other data entry forms with a forced link to the current record.

same time all other buttons leading to a linking of Object information can be disabled, making linking of Objects to Layers the only option available.

Thus by controlling what buttons are available, where and when, and what they will call up, structure can be provided to the recording of excavation information. Different views on the structure of excavation information — that is different data models — can be mapped onto the system by varying the availability of input forms, and not least through the sequence in which these must occur, as well as what possibilities of linking to other entities are available.

Figures 6 and 7 show two examples of data models for excavations. In figure 6 a three-level model is shown. The Construct is seen as the basic entity categorising all observations at the excavation level. That is the excavator has to interpret his observations in the field in terms of structures and features as he proceeds. The Constructs which may well be nested in internal hierarchies are characterised by containing one or more layers or fills, and each layer may contain one or more objects. The principal constraint of this model is that an Object has to be a part of a Layer, which has to be a part of a Construct. Photos and Drawings are seen as independent documentation evidence that may be freely linked to any of the three main entities.

Figure 7 shows a two-level model based on Layers and Objects. A model of this nature, where the Construct entity has been excluded, is widely used in excavations from the old Stone Age, where a geological frame of reference rather than one of man-made structures is prevalent. However, even here the Construct entity is needed. As stated in the paragraph on the conceptual model the Construct covers more than we immediately observe during the excavation. During the post excavation phase or even during the excavation itself we may add interpretations in terms of categorisation of information that goes beyond 'evident structures'. Potentially at any point we can record what has been termed 'latent structures'. Thus in old Stone Age excavations analytical entities like 'living floors' or 'activity areas' are frequently isolated as if they were observable. As a consequence we have to supply forms that allow the entering into the Construct table of latent or inferred structures and the free linking of these regardless of how the model is otherwise structured.

A specific data model should be applied to any excavation to avoid ambiguity in recordings. However, there are reasons why it should be possible to bypass the data model and let experienced operators go into a mode of unconstrained data entry. First of all, if data entry is to occur in a post excavation situation from written lists, it provides far the fastest way of data entry. Secondly, if mistakes occur for one or other reason, it may provide the fastest way to clear things up. In unconstrained data entry mode the operator has to be very much aware of the model (s)he is dealing with, and use those links only that will give the intended structure.

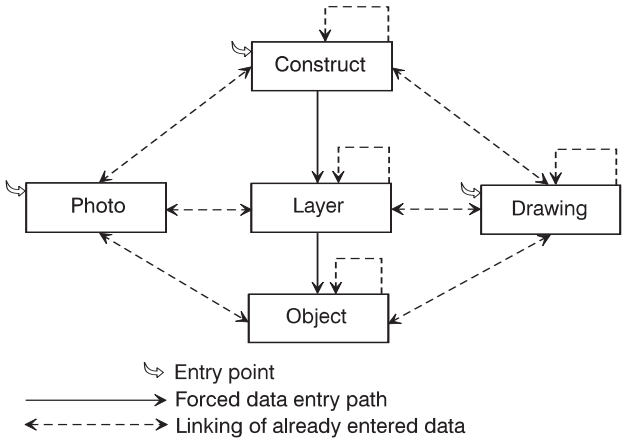

Figure 6. A three-level model for the recording of an excavation with constraints demanding that any object must be part of a layer, which must be part of a construct. 'Entry point' indicates where data input can be initiated.

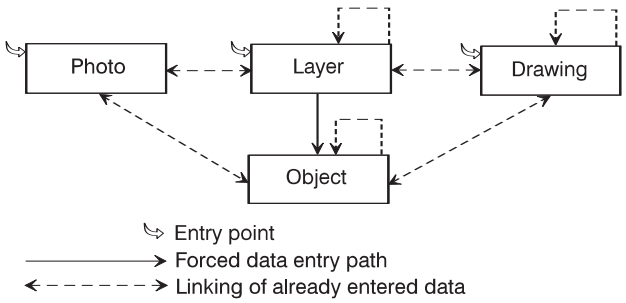

Figure 7. A two-level model, where the Construct entity has been excluded, and where any object must be part of a layer. "Entry point" indicates where data input can be initiated.

As demonstrated, it is fully possible to map different data models onto the underlying table structure. However, it is not sufficient for us to have a system where we as programmers can set up different views of the database. It is important that it is the user who can customise the system dynamically.

To make user customisation possible we have created a data model definition module, where it is possible to define how the connections between tables should be presented in terms of optional linking or forced data entry flows. Basically, the customisation is implemented through a full square matrix, where the five basic tables are cross related

with each other (fig. 8). A number of possible values for each cell is defined by us, and can be selected by the user in drop down boxes for each cell. A data model may then be defined by setting a combination of the values in the matrix. The choices in each cell include link, no link and (forced) path in the off-diagonal cells and a combination of entry/no entry and link/no link values on the diagonal.

The user can define and select a model. However, we cannot allow this to happen at any time. It is important that there is consistency in recordings throughout any particular excavation. Thus, when initially a project is defined, the user must select the model to be used, and from that moment it is a binding choice for that particular project, not to be changed. Anything else would be an invitation to chaos.

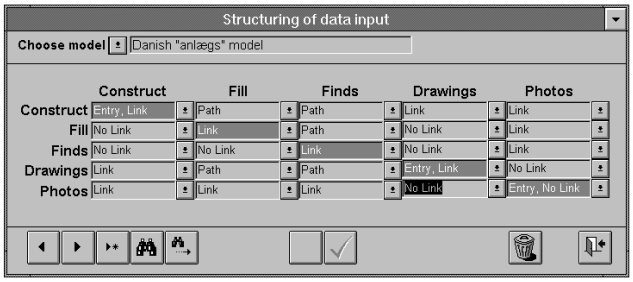

Figure 8. Form used to define a data model. Each cell has a drop down box providing a number of pre-set alternatives.

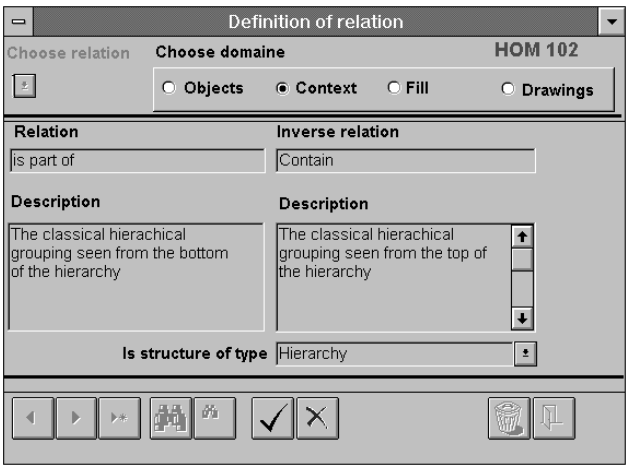

Figure 9. Form used to define internal relationships in entities.

4.2 Definition of internal relations of entities

The capability to handle relationships internally between instances of entities makes the system extremely powerful and versatile. It enables users to build up data structures dynamically, and thus removes one of the major weaknesses of most digital excavation recording systems sofar — the predetermined data structures hard coded into the system by the programmer. Thus for the Constructs entity it would in most traditional systems be necessary to operate with a pre-defined hierarchy of features, structures, groups etc. and to stick with these. In our system, however, it is possible to assemble data structures of Constructs dynamically into higher levels of interpretation units to any level and any complexity.

In order to make the use of the internal relations as flexible as possible the users can define the relations they need (fig. 9). Four types of information have to be supplied in the definition form. The domain (i.e. whether the relation is valid for Objects, Layers, Constructs or Drawings), what the relation should be named, the name of the inverse relation, and what abstract data type the specific relation is an instance of. If the relation is symmetric, (i.e. 'same as') one enters the relation name in the inverse relation field as well. In figure 9 an example of defining a parts breakdown is shown: the domain is Constructs, the relation is 'is part of', the inverse is 'contain', and the type of structure is 'hierarchy'.

Following the definition of the relationship the user may use it to relate an instance of a Construct to another instance of a Construct. Thus a specific 'post-hole' may be assigned to a specific 'house' as being 'is part of' that house. Seen from the viewpoint of the house, a 'contain' link to the post-hole will automatically be entered by the system in the same transaction.

The type of structure is selected from a number of types we have defined. Currently we have isolated a set of six types: Set, Series, Hierarchy, Web, Directed web without loops, and Directed web with loops.

A 'Set' represents an unordered collection of items. The relationship 'same as' is a typical representative of this type. A 'Web' is an unordered collection of items, too. But in contrast to the 'Set', the edges between the items are significant. The relationship 'fits together with' (i.e. used for refitting of sherds) is a typical representative of the 'Web' type. The type 'Directed web with loops' is used if a relationship has an inverse, and if it is possible to return to a specific element when a path through the structure is followed. Currently, we have not come across representatives of this type but we will not exclude their existence, i.e. in complex webs of interpretation.

A 'Series' is an ordered collection of items. The relationship 'is younger than' is a typical representative of this type. A more complex structure is the 'Hierarchy' mentioned above. It is a recursive structure with only one top-node and only one edge pointing to any other node. One may distinguish several variations of 'Hierarchies', but we have found no reason to do so at the moment. A more general structure is the 'Directed web without

loops' used, for instance, in stratigraphical relationships. In this structure a node may be pointed at by one or more edges.

Each type represents a characteristic organisation of data (fig. 10), which can be used in the system in different ways. Thus knowing that the structure being defined in figure 9 is of the type 'Hierarchy' we can set up a checking mechanism on data input which prohibit logic failures. I.e. if 'post-hole A' is part of 'house B' then an attempt to record 'post-hole A' as being a part of 'house C' would result in an error, because the entry would violate the constraints of the structure defined.

Another area where the type information can be used is in connection with the presentation of data to the user. As can be seen from figure 10 each type has inherent graphical characteristics that may be utilised in presentation screens (Ryan 1995). Thus we do not need to know the actual content of the recordings, only their type in order to do a proper presentation. Another perspective is that we will be able to combine and display several relationships, as for instance when combining 'same as' and 'lies above'/'lies below' relationships for stratigraphy.

4.3 Setting up classification and description systems for entities

A further and very important point of user customisation is the possibility to classify and describe the content of the individual records of the entities Constructs, Layers and Objects.

It is of course easy enough to implement a user defined classification, unique within each individual project. However, the critical issue is the varying number of variables relating to each class defined, and their values. We are not yet through with the implementation of this, but we have made some successful proto-typing exercises. The first step is to set up a structure to hold the user defined classification and description system. As shown in figure 11 this can be done in three tables, Object Type, Description Variables and Values of Description Variables linked in that order to each other with one-to-many relationships. This will allow any type to have an unspecified number of variables and any variable to have an unspecified number of potential values.

To use this system in our database we will need at least two tables. The Object table is identical to the one already existing in the database, including a field for the basic classification of the Objects. The other table, linked to the Objects table in a one-to-many relationship, holds the description of the object in terms of the particular variables relevant to the type of the object and the values they exhibit.[3] The two table solution may not, however, be flexible enough. If an object has to be classified as type x as well as type y, we need to add an extra table holding the classification exclusively. That is, we will need a table holding the artefact as an object (the one we already have), a table holding one or more classifications of the artefact, and a table holding the descriptions. These three tables are linked of course in one-to-many relationships in the order mentioned.

A possible way to implement the variable description in relation to a classified object is through a pop-up form

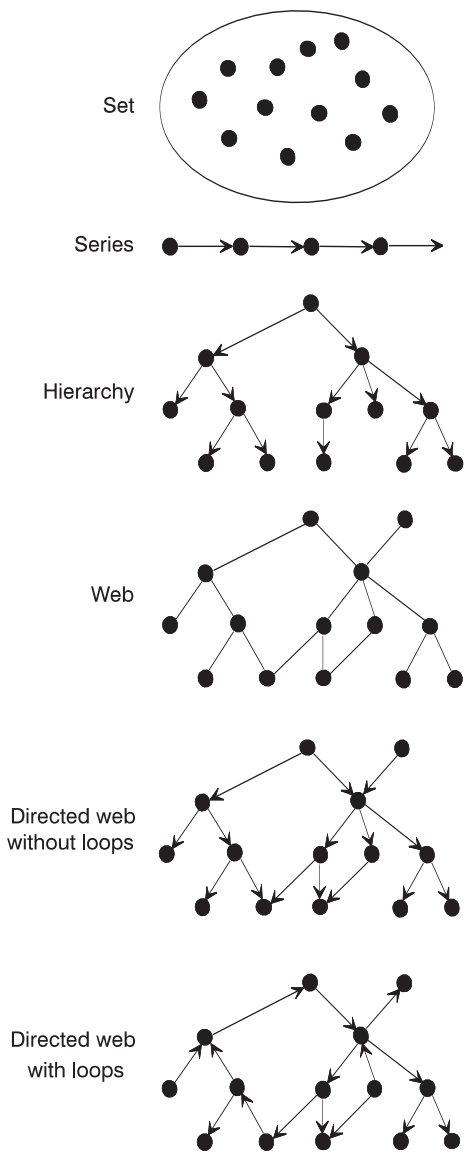

Figure 10. A graphical presentation of the characteristics of six data structures recognised in IDEA.

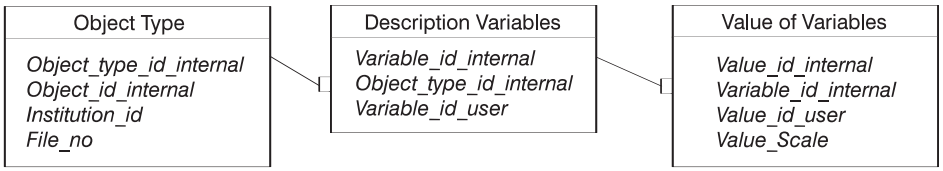

Figure 11. A table configuration for a general object classification system.

(fig. 12). The form works in tight correspondence with the classification tables. The choice of a type for an object, decides which variables will be available in the drop down list of the pop-up form for the variable field, and subsequently the choice of a variable will decide what values are available in the drop down list of the value field. Pressing the buttons of the drop down fields in the wrong order will merely result in a lack of available choices.

4.4 Setting the Language

An obvious area of customisation is the language presented to the user on the screen. Whereas our documentation of the system is held in a language akin to English (Danelish), the user interface has to be native to the area where it is presented. We have solved this problem by storing all labels, messages, etc. in a special language table, where each language has its own column, and each string to be presented to the user has its own row. Initially we enter all strings in this table in Danish and English, the two languages currently supported. Adding a new language is simply a matter of translating the strings in the table and writing the translation in a new column. Selecting a language is a simple choice among the currently available languages in a combo box in the customisation form. The language will change instantly. The currently selected language will be saved to an ini-file, and thus be remembered from session to session until a new language is selected.

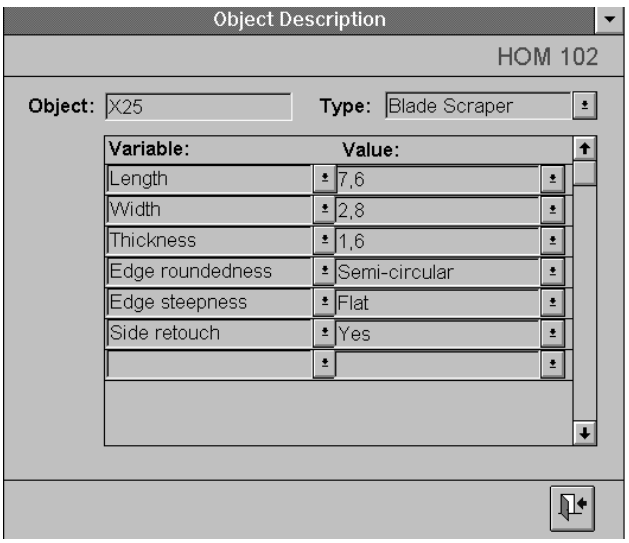

Figure 12. A form for entering a classification and a formal description of an object. Variables available in the drop down boxes will vary according to type chosen, and values will vary according to variables chosen (at the time of writing the classification system has not yet been implemented in IDEA).

5 Programming with a minimum of code

Since the beginning of the project we have placed a lot of effort in making the system as easy and cheap to maintain as possible. We have tried to keep the amount of written code to an absolute minimum, as software development projects producing thousands and thousands of lines of code tend to drown in their own complexity. The result is all too often progressively rising costs in maintenance at best and total failure at worst.

5.1 Event programming of forms

The answer of the software industry to the code boom problem, is Object-Oriented Programming (OOP) (Booch 1991). One of the key features of OOP is that objects are self-contained modules, which respond to events. In the case of Microsoft Access, the various objects of the form (i.e. the controls or the form itself) offer the programmer the possibility to write code defining the functionality of an object responding to a particular event. The code itself is not written in an object-oriented language but instead in a Basic dialect. This Basic dialect is able to interact with the so-called DataAccessObjects library.

Using this functionality, the five basic entry forms include only approx. 250 lines of code each, including error-messages. Most of the buttons at the bottom of the forms are enabled and disabled by one global module which contains approx. 150 lines of code. The two forms for setting the internal and external relationships contain approx. 300 lines of code each. Altogether the amount of code is negligible.

5.2 Creating complex reports with no code at all

One of the areas, where Access really proves its 'fourth generation status' is through its report generator. At first it does not look much. At a second glance you realise that it is very much akin to the forms generator, and indeed has inherited most of its properties from this (Access is indeed object oriented behind the curtains). Finally you realise that a number of features has been added, compared to the forms definitions, making the report generator a truly flexible tool by itself.

One of its stronger features is that reports can be embedded and fully synchronised within reports three levels deep (the same applies to forms). The result is that you can set up a highly complex hierarchical structure involving a number of entities without writing as much as one line of code. In fact for all practical purposes we can handle our rather complex data structure in one code-less construction. Thus without a line of code we can build a list of features from the construct table, where for each feature the layers are listed, and for each layer of the feature, the objects with their descriptions are listed.

In Denmark there is an authorised format for so-called Level II data for the archives (see Carver 1985, fig. 4 for this concept — with reference to the Frere report). This format involves a fair amount of hierarchically organised lists, where especially the lower levels of the hierarchy are cross-referenced to other lists. We have succeeded in implementing this format in the system, and it only takes a press of a button to write out the complete report. Codes are used in two areas only. One is for the creation of an index to the report. The other is to tie parallel parts of the report together in a sequence (in fact we could have avoided this had a fourth level of embedding been available).

6 Future development of the IDEA

What we have achieved sofar is a flexible system for recording traditional textual information from excavations. This is the result of the first year of the project. There is, however, a very long way to go before the IDEA becomes reality.

Next, we have to add full support for the recording of three-dimensional data, not just in terms of a point in space for an Object, a Layer or a Construct, but also in terms of two-dimensional polygons positioned in a three-dimensional space. We have no ambition of achieving a full three-dimensional presentation, since the archaeological recordings in the field in plan and section are — and will always be — ambiguous with respect to a full 3-D representation.

We have characterised the system as an integrated database for excavation analysis, and for very good reasons. Our primary objective is to create a system which is analytical in its approach to excavation recordings rather than just descriptive. In order to obtain this we will implement analytical methods into the system. These are planned to be of two kinds. One set will be implemented around a GIS-type of interface. Another set will be based on a graph browser type of representation of data. The idea is that we will allow for different types of views on the same set of data, and that any interrogation in one representation is reflected dynamically in others (Ryan 1995).

Providing a system with the ability to handle spatial data, requirements for constraints in the spatial location of entity instances will be created. If for instance a find is assigned to a layer, it is natural that the system should check for consistency in their spatial relationships, that is whether the find location is within the boundaries of the layer. Furthermore it will become necessary to include a quantifier in the various link tables to store information like 'post-hole A' 'is x meters from' 'post-hole B'. The analytical perspective is that the quantifier may serve as a distance operator in descriptions of complex and heterogeneous archaeological objects (Dallas 1992; Dickens 1977).

One set of problems we probably cannot solve within the three year limit of the current project, is the question of user-defined queries and reports. One may foresee the demand for a 'Wizard' (essentially another specialised database) which sets up screen presentations and reports according to the defined model for the project. Furthermore one can foresee a demand for facilities to query multiple projects with different data models, classifications and relationships.

7 Conclusion

In our opening form we have used the Globe as a symbol. By conviction we seek global rather than narrow *ad hoc* solutions to problems. We believe quite simply that by addressing problems in their total complexity we will come up with solutions which in the long run are worth far more than the quick *ad hoc* solutions.

Our project is to run another two years (at the date of CAA95), and we have no guarantee that we will be given money to continue thereafter. It is not however to be an academic project, where the results disappear with the money. We are determined that the destiny of the system is to be used and to be further developed. To ensure this we are very open to cooperation with anybody who can *seriously* contribute to the development of the system. We have already initiated co-operation with different people on various aspects of the system, and it is indeed our hope to be able to widen this co-operation.

notes

1 We coined the word construct because it sounded as a good and meaningful term for the concept. At that time we did not know (shame on us) that Gardin (1980) had used the same word with a very similar meaning.

2 This solution is due to Paul Zoetbrood, generously offered during a very memorable evening at the CAA88 in Birmingham.

3 The idea for this implementation was abducted from a set of tables presented by Lene Rold (1990: 14)

references

Andresen, J. T. Madsen 1992 Data Structures for Excavation Recording. A Case of complex Information Management. In: C.U. Larsen (ed.), *Sites & Monuments. National Archaeological Records*, 49-67, Copenhagen: The National Museum of Denmark.

1994 Strukturen af data fra udgravninger, *KARK Nyhedsbrev 1994* 1, 4-27.

Arroyo-Bishop, D. M.T. Lantada Zarzosa 1992 The ArchéoDATA System: A Method for Structuring an European Archaeological Information System (AIS). In: C.U. Larsen (ed.), *Sites & Monuments. National Archaeological Records*, 133-154, Copenhagen: The National Museum of Denmark.

Booch, G. 1991 *Object oriented design with applications*. Redwood City, Ca.: The Benjamin/Cummings Publishing Company Inc.

Carver, M. 1985 The friendly user. In: M.A. Cooper/J.D. Richards (eds), *Current Issues in Archaeological Computing*, 47-61, BAR International Series 271, Oxford: British Archaeological Reports.

Dallas, C. 1992 Relational description, similarity and classification of complex archaeological entities. In: G. Lock/J. Moffet (eds) *Computer Applications and Quantitative Methods in Archaeology*, 167-178, BAR International Series 577, Oxford: Tempus Reparatum.

Dickens, P. 1977 An Analysis of Historical House-plans: A study at the Structural Level (Micro). In: D.L. Clarke (ed.), *Spatial Archaeology*, 33-45, London: Academic Press.

Gardin, J.C. 1980 *Archaeological constructs. An aspect of theoretical archaeology*. Cambridge: Cambridge University Press.

Madsen, T. 1993 Archaeology, data structures and computer science. In: J. Pavúk (ed.), *Actes du XII^e Congrès International des Sciences Préhistoriques et Protohistoriques Bratislava, 1-7 septembre 1991*, 269-273, Bratislava: Institut archéologique de l'Académie Slovaque des Sciences à Nitra.

1994 Integrating methods and data: reflections on archaeological research in an IT environment. In: I. Johnson (ed.), *Methods in the Mountains. Proceedings of UISPP Commission IV Meeting, Mount Victoria, Australia 9th - 13th August 1993*, 27-34, Sydney: Sydney University Archaeological Methods Series #2.

Madsen, T. J. Andresen 1993 'Den elektroniske gravebog': En informationsteknologisk løsning på udgravnings-registrering og -bearbejdning, *KARK Nyhedsbrev 1993* 3, 7-14.

Rains, M.J. 1995 Towards a computerised desktop: the Integrated Archaeological Database System. In: J. Huggett/N. Ryan (eds), *Computer Applications and Quantitative Methods in Archaeology 1994*, 207-210, BAR International Series 600, Oxford: Tempus Reparatum.

Rold, L. 1990 Databaseteknologi, teori og design. In: C.U. Larsen (ed.), *Arkæologi, statistik og EDB*, København: Københavns Universitet.

Ryan, N. 1995 The excavation archive as hyperdocument? In: J. Huggett/N. Ryan (eds), *Computer Applications and Quantitative Methods in Archaeology 1994*, 211-220, BAR International Series 600, Oxford: Tempus Reparatum.

Jens Andresen and Torsten Madsen
Institute of Anthropology and Archaeology
University of Aarhus
Moesgård
8270 Højbjerg
Denmark
e-mail: farkja@moes.hum.aau.dk
farktm@moes.hum.aau.dk