From Data Capture to Code Generation: Tools for Entity Modeling

Heather Fulford
Loughborough University

David Bowers
University of Surrey

search

From data capture to code generation: tools for entity modelling

Heather Fulford
Business School
Loughborough University
Loughborough
Leicestershire
LE11 3TU
UK

David Bowers
Dept. of Computing
School of Electronic Engineering, IT & Mathematics
University of Surrey
Guildford
GU2 5XH
UK

***Abstract* – The entity-relationship approach to conceptual modelling has long been at the heart of information systems design. Most automated tools and CASE environments created to support database design tend to start at the conceptual modelling stage. This assumes that somehow the analyst has been able to deduce, from the initial requirements specification, what entities are to form part of the system and how they are interrelated. We bring together in this paper two strands of our research to present a set of prototype tools to support the major stages of database design, starting with the tasks of document analysis and data capture, and progressing through to code generation. We conclude with a proposal for an integrated environment for database design.**

I. Introduction

The process of Information Systems Design has been presented by several authors as one of constructing multiple models of the system, each one an abstraction of the output of the previous phase. These successive phases are represented graphically in Figure 1, adapted from [1]. The result of each phase, indicated on the left of the figure, is used as input for the next. (The original figure, in [1], indicated also the feedback required between phases for the overall process to be successful.)

The Entity-Relationship (ER) approach to conceptual modelling [2] has long been at the heart of the Conceptual Modelling phase. Traditionally in this approach, the analyst begins examining relevant system documents and extracting from them constructs, including entities, relationships, and attributes. Using these basic constructs, the analyst then draws entity relationship diagrams both as a means of documenting the required system and also to provide a graphical expression of the system requirements to verify with the client.

As with the other stages of design, the entity relationship approach is laborious and expensive, often requiring much iteration to construct an acceptable conceptual model. The information systems and software engineering literature of recent years contains details of a number of efforts to automate various stages of the approach: programs have been written to automate the task of capturing entities from system documents, and diagramming tools have been developed for creating entity relationship diagrams.

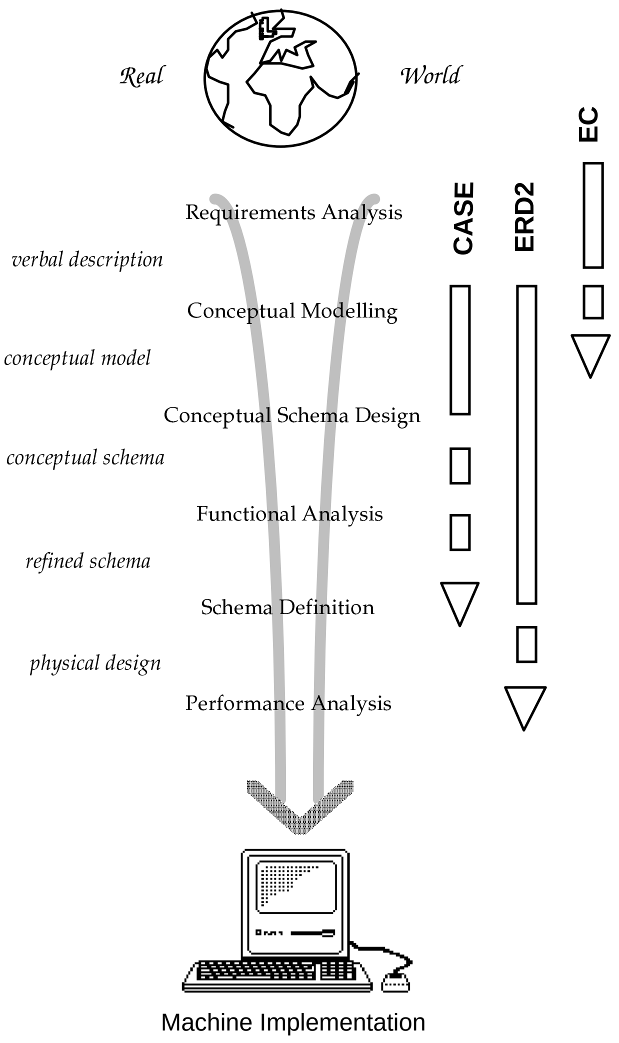

Figure 1: The Phases of the Design Life Cycle

Typically, Computer Aided Software Engineering (CASE) tools have been developed to assist with the primarily

search

graphical phase of Conceptual Modelling using ER models. As suggested by the first ("CASE") outline arrow on the right of Fig. 1, such tools can be limited in their ability to generate high quality code (physical schema designs). This is largely because they tend to overlook one of the principal difficulties associated with graphical languages: it is quite easy in most such languages to write (or, rather, draw!) models which are inconsistent or redundant.

In the case of Entity-Relationship modelling, the level of normalisation of a generated relational database schema, for example, depends on the absence of redundancy in the Entity-Relationship model from which it has been generated. We have reported elsewhere [3] our development of a prototype tool, ERD2, which analyses an ER model to identify potential redundancies, and attempts to resolve these by means of a dialogue with the user. Having resolved redundancies in the initial model, it is able then to generate higher quality SQL code, including several types of constraint and the definition of appropriate indices. As suggested by the second ("ERD2") arrow in Fig. 1, this prototype extends the support provided into the later stages of the design process.

Most automated tools and CASE environments tend to start at the conceptual modelling phase. This assumes that, somehow, the analyst has been able to deduce, from the initial requirements which they have been given, what entities are to form part of the system, and how they are interrelated. Often, the initial requirements will have been provided in the form of written documents or verbal descriptions, and a large proportion of the initial task involves recognising, from text, the entities, their attributes and the relationships between them which are required in the system. It is our work on automating this first phase of the overall process, to which we refer as "entity capture" (indicated by the third, "EC", arrow in Fig. 1.) which is presented in this paper.

It is widely acknowledged that automating the task of capturing entity relationship models from text is a non-trivial exercise, owing to a large extent to the inherent complexity of natural language, and in particular to the complex representation in text of entities, relationships and attributes.

A major objective of our research has been to design and develop a toolset to extend support to the database designer through all stages of the information systems design process: from document analysis and data capture through to code generation. In seeking to meet this objective, we have first studied the linguistic representation of entities, relationships and attributes in text, and have been developing an approach to capturing these constructs based on the identification of co-occurrence patterns and 'linguistic cues'. As with the code generation tool described in [3], our emphasis has been very much on the provision of tools to 'guide' the analyst through the task of scanning system documents and providing 'evidence' of relevant constructs relevant to entity modelling, rather than an approach which purports to achieve full automation of the data capture task.

In this paper, we present an overview of our research to date, and indicate how we intend to carry this work forward to the creation of an integrated toolset to support each key stage of the entity modelling approach to information system design.

The paper is structured as follows: we begin (section II) with some background details about the entity relationship approach to conceptual modelling, noting some of the difficulties associated with automating the data capture task, and outlining some of the recent efforts to extract entities automatically from system documents. Next (section III), we present an overview of our approach to capturing entities, relationships, and attributes from text, and illustrate how the approach works. In section IV, we discuss the features of metaCASE technology which are required to allow our entity-capturing approach to be incorporated in an integrated design tool. Finally, in section V, some concluding remarks are made about our work to date, and some indications given of how our work is being further developed to produce an integrated toolset or workbench to support database designers throughout the entity modelling approach, from data capture through to code generation.

II. Background and Statement of Problem

We consider in this section the nature of the problem of data capture for entity modelling and outline some recent attempts to automate the task.

Entities are generally associated with nouns or noun phrases in descriptions of required systems [4], attributes typically with adjectives, and relationships with verbs. A similar mapping has been suggested for objects [5], with verbs being used to define methods.

Thus, one heuristic presented in systems analysis and database texts, such as [6], [7], [8], is to scan the narrative descriptions (documents) of a required system for potential entities, which would appear in the descriptions as nouns. The task, however, is not as straightforward as it might at first seem (as a number of authors have noted): for example, not all nouns in system descriptions are entities. Unfortunately, nouns can also represent attributes, and the texts need to qualify the heuristic with warnings to this effect. Indeed, some, such as [9], admit that there is no simple answer to the question, "what constitutes an attribute and what constitutes an entity?", and that the distinction depends both on the structure of the enterprise being modelled and on the semantics of the items concerned.

There are complexities too on a purely linguistic level rendering the identification of entities, attributes and relationships a difficult undertaking. For example, there is no single way of representing nouns in text: nouns may, for example, comprise single words (e.g. *bungalow*) or possess one or more preceding adjectival modifiers (e.g. *detached house, semi-detached house*). Linguistic knowledge alone is often not sufficient to determine whether an adjectival modifier represents a 'one-off' modification or whether the modifier is an integral part of the underlying concept. Consider, for instance from the domain of ornithology, the functions of the adjectival modifier *great* in the following two phrases: *I saw a great seagull yesterday* and *I saw a great auk yesterday*. In the former case, the adjectival modifier *great* simply tells us that the seagull was very large (or perhaps terrific), whereas in the case of *great auk*, the adjectival modifier is an integral part of the a specific concept (i.e. a particular species of bird). The analyst has to apply his/her own judgment, domain knowledge, and linguistic understanding in order to be able to distinguish between these various linguistic representations and to 'decode' the system description appropriately.

A further set of issues concerns *homonyms* and *synonyms* [8]. The former term refers to the same word or phrase being used – often in different documents – to refer to different concepts. The latter refers to a situation that is more common, but no less confusing, where different words are used to refer to the same concept. For the analyst to be sure that such problems are resolved correctly, significant interaction with the client is usually required.

What is clear from discussions in the literature of the entity modelling approach is that the task of data capture from systems descriptions is highly complex, involving the analyst in a series of judgments and decisions based on, inter alia, domain knowledge, linguistic knowledge, client interaction, experience, and intuition.

Moreover, whilst the foregoing discussion has been concerned with the identification of entities and attributes from text, it should be noted that broadly similar approaches are usually recommended for the classical object-oriented approaches [10], [11]. However, in [12] it is noted that object oriented requirements analysis is still unsatisfactory and difficult to perform; indeed, the same range of linguistic and semantic problems seem to arise.

In order to capture entities – or objects – from system descriptions, some form of system description must exist: this suggests the question, where do such descriptions originate? One possibility is given in [13], which refers to a study of JSD (Jackson Structured Development) analysts, who were found to precede their application of the JSD method by some form of fact gathering, in order to acquire a detailed understanding of the required functionality. The study found that the analysts most frequently employed simple narrative, thus generating a form of textual description for the required system.

A particular difficulty with descriptions generated during the discovery process is noted in [14], which asserts that customers express their requirements, when describing them to analysts, in their own terms, assuming implicit knowledge of their environment. It is then the analysts' task to comprehend and re-express the requirements in a manner which all those involved in the analysis process can understand. This echoes [15], in which requirements engineering is said to be,

> *"… a systematic way of developing requirements through an iterative process of analysing the problem, modelling the resulting observations and checking the accuracy of the understanding so gained with domain experts using the model as the baseline of communication."*

It is suggested, further, in [16] that the construction of an initial domain (object) model can facilitate the exploration of the vocabulary of the domain. Since the clients participating in the process are unlikely to express their understanding in terms either of entities or of objects, this again suggests the generation of some form of verbal description.

Several approaches have been proposed for improving the requirements engineering process. In [13], a problem-solving approach, involving both analysts and clients, is suggested. The technique employed is the systematic exploration of several informal models or "scenarios", each of which addresses a part of the requirements. When each scenario has been agreed as being accurate, its description is analyzed, using the linguistic basis of conceptual graphs, and the results of the analyses are consolidated into a domain knowledge base and a user fact base.

An alternative approach, which also emphasises cooperative exploration of the problem area with the client, is described in [17]. Here, soft systems approaches are used to enhance the client's participation in the process. By involving the client, it is claimed that objects which are sensible and appropriate to the user will be identified than would be the case with more traditional data- or process-driven methods.

A third approach, presented in [12], attempts to define a formal correspondence between linguistic patterns and conceptual patterns. This is achieved by expressing the former in terms of predicate logic, the latter in terms of set theory, and testing the mathematical equivalence of the two representations.

It is this last approach which is closest to our own work on data capture from system descriptions. We restrict our attention, in this paper, to data capture for Entity-Relationship

search

modelling, which might be regarded as a subset of the object modelling concepts explored in [12]; indeed, the constructs of Entity-Relationship modelling are essentially the same as the static conceptual patterns used in [12].

The focus of our research on data capture has been to provide tools that 'guide' the analyst through the task of data capture from systems documents, rather than on producing tools that purport to automate the task fully. This allows the analyst to apply his/her skill and insight to resolving issues mentioned above, such as homonyms and synonyms, and, similarly, to address the question noted in [9] of the distinction between an attribute and an entity. Since such issues depend on domain knowledge and the semantics required of the system, it is unlikely that a satisfactory set of criteria could be developed to permit fully automated resolution.

The approach to data capture described in this paper draws on recent work in the areas of corpus linguistics, computational lexicography and terminology in which emphasis is placed on observing and studying language in use in free text (see for example [18], [19] and [20]). Specifically, our work arose from some earlier studies we conducted into data capture from English texts for knowledge engineering for expert system development [21] and for the compilation of English technical glossaries and dictionaries (terminology extraction). This earlier work is reported fully in [22]. There were a number of features of our earlier work on data capture which led us to consider its applicability to data capture for entity modelling purposes. These features include the fact that our approach, unlike some other existing approaches (discussed in [22]), is both domain and text-type independent; there is no requirement for the input text to be tagged; the approach does not rely on (computationally intensive) parsing; we can process single texts or batches of texts; and the entity capture facility identifies single word entities as well as multi-word entities.

It will be apparent that these features are appropriate for an environment where the range of text descriptions is not predetermined, and could well range from operation manuals or specifications for existing systems to informal notes taken during a fact-finding interview.

An overview of our approach, including an illustration of data capture from a sample system description, is presented in the next section.

III. An Overview of Our Approach to Data Capture

In order to construct an Entity-Relationship model, it is necessary to identify a number of constructs. These include:

- Entity types *including subtypes*
- Attributes *including identifiers and inherited attributes*
- Relationships *including degree and optionality*

Our research to date has focussed principally on the capture of entities from system documents, and it is this aspect of our work on which we report in this section. More recently we have begun refining our approach to entity capture to incorporate the identification of subtypes, attributes and relationships. We give some brief indications here of these refinements.

We illustrate our proposed approach by considering the following example. The initial description of the system requirements, given in Figure 2, is typical of a problem statement, and might have been produced either as part of a request for a system or as the result of an analyst's initial interview.

An Estate Agent's Database

An Estate Agent is responsible both for arranging sales of properties and also for managing property rentals. Properties can be houses, maisonettes, bungalows or apartments, and contain a number of bedrooms. Each has a unique address. The dimensions (length and breadth) of each bedroom, in metres, are to be stored, as are those of the main living room. For rented properties, the monthly rent and the minimum rental period, in months, are required, whereas for properties for sale the asking price is needed. The owner of every property is to be recorded also, giving the name and address of the owner. It should be noted that people may own several properties.

Figure 2. System Requirements for Estate Agent's database.

Applying standard entity modelling techniques should yield a model of the form in Figure 3. Identifiers are denoted by **bold** attribute names, "crow's feet" are used to specify "many" (i.e., one or more) cardinality, and the connection between *Property for sale*, *Rental property* and *Property* indicates that the first two are (exhaustive) partitions of the last.

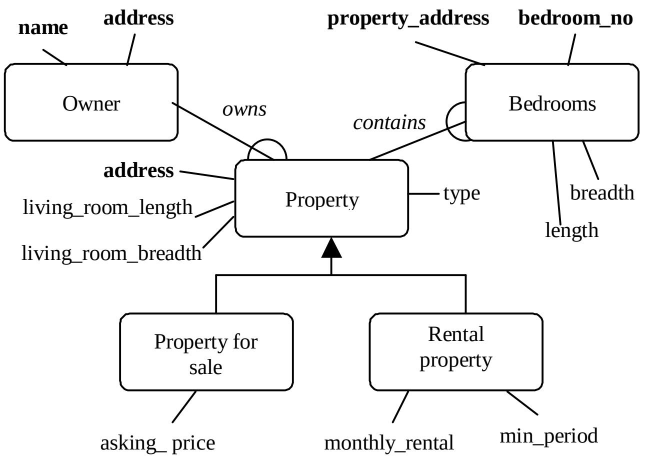

Figure 3. Entity-Relationship model derived from description in Fig. 2

It should be noted that the description of Figure 1 suggests additional subtypes for *property* (*house, maisonette, bungalow* and *apartment*), but the simple attribute, *type*, is adequate representation in the absence of further distinguishing properties.

A further complication is associated with the existence of an *address* attribute for both *property* and *owner*; these must be recognized as being distinct. This is a trivial example of a pair of homonyms.

The choice of identifiers for *owner* and *bedroom,* and, indeed, the introduction of the attribute *bedroom_number* are standard assumptions which a skilled analyst might make, and which it would be difficult to automate completely. Once made, such assumptions require verification with the client, in the same way as any suggestion made by an automated tool would have to be accepted or rejected.

As noted above, the systems analysis literature encourages analysts to scan narratives for nouns in order to identify entities. Taking as our starting point this heuristic that nouns are likely to be entities, we have developed a tool for capturing nouns (candidate entities) from system documents. This tool, originally implemented in Prolog, and currently being refined using a set of macros in Microsoft Word, is capable of capturing single-word nouns (e.g. *bungalow*) as well as multi-word nouns (e.g. *estate agent*). Our approach to entity capture relies on an analysis of co-occurrence patterns in text, the basic premise of the approach being that an entity (i.e. a noun) in text is likely to reside in English texts *preceded* by a 'boundary marker' comprising a so-called function word (determiners, pronouns, prepositions, conjunctions, etc.) or a punctuation mark, and *followed* by a boundary marker comprising a function word or punctuation mark, as summarized in the following pattern:

Function word/punctuation + ENTITY + function word/punctuation

We arrived at this basic premise about the co-occurrence patterns of nouns in text from a careful study of a selection of specialist and technical texts spanning a range of subject areas and text types (as reported in [22]), as well as by drawing on studies of corpus linguistics and term extraction [see for example X-Sager].

Table I below provides a summary of the possible permutations of co-occurrence patterns (taken from the sample 'estate agent' systems description presented above).

TABLE I
CO-OCCURRENCE PATTERNS FOR ENTITY CAPTURE

Boundary marker	Entity	Boundary marker
an	estate agent	is
of	bedrooms	.
,	bungalows	or
,	maisonettes	,

Implementing our approach was quite straightforward from a computational point of view since both function words and punctuation can be thought of as a closed set: we could easily compile a list of such items for use by our program. Using this list of function words and punctuation, the program scans a given text highlighting as candidate entities items which reside between function words and/or punctuation.

In common with other programs for noun identification (discussed in [22]), the output from our co-occurrence pattern approach to entity capture typically contains a certain amount of noise. In order to keep this noise to a minimum, we filter our output using a stoplist (again, in common with other approaches to noun identification). This stoplist was created following a study of the theoretical and empirical literature of specialist and technical writing (see for example [23]) and comprises commonly occurring words and phrases that are typically used in technical writing. The list includes reporting verbs (e.g. *to note*, *to state*, *to say*), and phrases (sometimes referred to collectively as 'linking words') that are used to 'refer back to points which have already been stated or forward to points which will be made later' [20]. It further contains a collection of other general linking expressions, such as *consequently*, *in addition to*, *a number of*, and *for example*, and some frequently occurring verbs.

Applying our co-occurrence pattern approach to the estate agent system description (presented above) would yield the following (candidate entities marked in bold):

> *An **Estate Agent** is **responsible** both for **arranging sales** of **properties** and also for **managing property rentals**. **Properties** can be **houses**, **maisonettes**, **bungalows** or **apartments**, and contain a number of **bedrooms**. Each has a **unique address**. The **dimensions** (**length** and **breadth**) of each **bedroom**, in **metres**, are to be **stored**, as are those of the **main living room**. For **rented properties**, the **monthly rent** and the **minimum rental period**, in **months**, are required, whereas for **properties** for **sale** the **asking price** is needed. The **owner** of every **property** is to be recorded also, giving the **name** and **address** of the **owner**. It should be noted that **people** may **own** several **properties**.*

Comparing the candidate entities highlighted in the above text with those presented in the entity relationship diagram (Figure 3) reveals that all of the entities represented in the diagram are proposed as candidates except for *property for*

search

sale. This entity is missed from our approach because it is a (rather rare) example of a nominal expression containing a function word (for). Note that our approach also proposes a number of attributes as candidate entities (e.g. *houses*, *bungalows*, and *maisonettes*), thus illustrating the problems cited in the literature (and referred to in section II above) of distinguishing between entities and attributes. We envisage that the preliminary work we have undertaken on attribute capture (outlined briefly below) will go some way to addressing this issue.

It will be seen from the above illustration that our approach incorrectly proposes as entity candidates *responsible* and *people*, the latter being a synonym for *owner*. Clearly, an analyst could easily apply his/her judgment and knowledge to filter such anomalies from the output.

When developing our co-occurrence pattern approach to noun identification for knowledge engineering and terminology extraction purposes, we devised a framework for evaluating its success (reported in [22]). We are now applying this framework to an evaluation of the approach within the context of capturing entities from system documents.

We believe that this framework has application beyond the evaluation of our own approach to entity capture, and hence represents a significant part of our contribution to the area of entity capture. We present here a brief summary of our framework, and then, using this evaluation framework, suggest how our approach to entity capture performs.

As reported in full in [22], in order to evaluate the success of our noun identification program in the context of terminology extraction and knowledge engineering, we selected documents from several subject domains and asked a domain expert from each domain to scan the documents manually marking the technical (domain) terms (which are largely nouns). The output of each expert was deemed to be reliable and thus was used as our datum. We then compared the output of each expert with the noun candidates proposed by our noun identification program. As a further evaluation measure, we analyzed the manual scanning output of terminology extraction experts (who are typically not domain experts), and compared this with the output of the domain experts as well as with that of the program.

Our evaluation framework comprised five measures of comparison:

1. **Match**: an item selected by a domain expert was also proposed as a candidate by our program;
2. **Truncation**: an item selected by a domain expert was 'shortened' (truncated) by our program, and hence only partially identified;
3. **Expansion**: an item selected by a domain expert was 'lengthened' (expanded) by our program, and hence only partially identified;
4. **Undergeneration**: an item selected by a domain expert was not proposed as a candidate by our program;
5. **Overgeneration**: an item not selected by a domain expert was proposed as a candidate by our program.

An illustration of this evaluation framework is presented in Table II below.

TABLE II
EVALUATION FRAMEWORK FOR ENTITY CAPTURE

Comparison	Item selected by domain expert	Candidate proposed by entity capture program
Match	bungalow	bungalow
Truncation	property for sale	property
Expansion	property sales	managing property sales
Undergeneration	name	-
Overgeneration	-	metres

Within a terminology extraction context, our noun identification program consistently proposed correctly 80% of the items selected by a domain expert. Further, the program partially identified (i.e. truncated or expanded) the remaining 20% of items selected by a domain expert. In common with over noun identification programs, our program substantially overgenerated (i.e. the output contained a certain amount of noise). Whilst we are investigating means of reducing this noise to a minimum, we believe it is preferable for the program to overgenerate rather than undergenerate. In the context of entity capture, for instance, it is arguably better for the program to propose too many items as entities (which the analyst can the eliminate using his/her domain knowledge, experience, linguistic knowledge, intuitions and judgments), than it is for the program to undergenerate, thereby leaving the analyst no option but to go back to the original documents and scan them manually to identify the items missed by the program.

The results of our evaluation within a terminology extraction context held true across subject domains and text types. Given the close correspondence between technical terms (largely nouns) and entities (also largely nouns), we believe that similar results can be expected from the program used in an entity capture context (with the proviso already alluded to that the entity-attribute distinction is addressed).

We turn our attention now to some preliminary work we have undertaken to identify sub-types of entities and attributes in system documents. Again, this work builds on some earlier research undertaken in the context of knowledge engineering and expert system development, and it draws on the literature of linguistics in which the linguistic devices used for representing semantic relationships are discussed (see for example [24], [25]). In this earlier work, we investigated means of tracking, in text, the lexical-semantic relationships holding between noun terms, including relationships of hyponymy (X is a kind of Y), and part-whole relationships.

Taking the hyponymy relationship as an expression of types and subtypes has allowed us to extend this work to include the capture of entity subtypes. Using the basic frame: X is a kind of Y as a starting point for the hyponymy relationship, we have collated a number of 'linguistic cues' which are used in text to denote this relationship. Such cues include *type of*, *sort of*, and so on. We have written a program which searches text and highlights portions of text containing the cues, the aim being to guide the analysts to the 'rich' portions of the system description in which relevant data are likely to be found.

Continuing our studies of language in use in text, we are currently investigating the adoption of the 'linguistic cue approach' to the capture of attributes from system documents. As noted earlier, it is acknowledged in the literature that distinguishing attributes from entities in text is a complex task. Like entities, attributes are often represented by nouns. The analyst's task is to decide which nouns are entities and which are attributes of entities. Again, this task typically involves the application of domain knowledge, experience, intuition and judgment. Hence, as noted in the literature, the automation of this task is fraught with difficulties. Whilst we are by no means claiming to have solved the problem of attribute capture, we believe that our existing work on identifying linguistic cues can be used to begin to tackle this issue.

Having studied a number of system descriptions and examined the items an analyst might select as attributes of entities, we have begun to collect a set of linguistic cues which seem often to be used to 'point' to the presence of attributes in a sentence. These include words and phrases such as *each*, *every*, *of the*, and *of a*. Identifiers might be captured by searching for such linguistic cues as *unique*, *only*, *sole*, and so on.

We believe that our approach to entity capture based on the analysis of co-occurrence patterns represents a feasible means of providing support to analysts engaging in entity modelling. Furthermore, we believe that this approach could be significantly enhanced by the development of our 'linguistic cue' approach to capturing entity subtypes, attributes and identifiers. We intend to investigate this latter area of work within the scope of a further research project, and to incorporate also an approach to identifying relationships holding between entities.

In the next section, we discuss our provision of CASE tool support to the next phase of system design, and indicate how we plan combine this work with the data capture work outlined above.

IV. Providing CASE Tool Support

It was noted in section I that we have developed already a prototype CASE tool that addresses the later phases of the design process [3]. By combining the work we describe in this paper with that tool, support would be available for all six phases of the process depicted in Fig. 1. The entity capture techniques, however, are provided currently in a tool separate from the CASE environment.

CASE tools normally function by supporting one or more forms of diagrammatic input, deriving textual, tokenised or frame-based representations of the objects represented diagrammatically, and manipulating the resulting text-based representations to generate code.

In our approach to capturing entities, not only do we start with (arbitrary) text, but the output of the process is also, essentially, text. Whilst this might suggest that the diagrammatic representation could be obviated completely, it is the diagrammatic form which is most readily comprehended, and which is normally used, therefore, to verify the constructed model with the client.

Further, it has been noted above that the entity capture technique will tend to overgenerate. In addition, there may be several refinements which a skilled analyst would wish to incorporate into a model based on documentation and description alone – not least to ensure completeness (what system is ever fully documented?). Such interaction with a CASE environment is likely to require rather more freedom than is normally available in a "traditional" CASE tool which supports a specific method.

The combination of using text as a starting point and the requirement for a high level of interaction with a tool incorporating "automatic" entity capture suggests that, rather than incorporating the entity capture technique directly into existing CASE tools, it will be necessary to develop a new environment. The facilities required – the abilities to generate, dynamically, graphical representations from text-based descriptions, and to support dialogues with the (tool) user to control such generation – are readily supported by meta-CASE tools, such as Lincoln Software's Toolbuilder [26], which we have used to develop the tool reported in [3]. The fundamental features of the meta-CASE environment which are relevant here are a fully object-based representation for every concept, with the ability to generate diagram objects

search

from text, or vice-versa, and a computationally complete language and user interface, which will allow construction of arbitrary control processes.

We have used Toolbuilder also to construct a design environment which supports input in a number of formats, including natural language and structured English [27]. Thus, we have demonstrated one mechanism for inputting raw text into a structured CASE environment, and this could readily be exploited to provide an input mechanism for an entity capture tool.

Finally, it should be noted that the generation of an ER diagram from a list of entities, attributes and relationships is itself non-trivial. In [28], an early approach to this problem is described, and that paper notes the extreme complexity of capturing sufficient semantic information, or, indeed, aesthetic rules, to allow the automatic generation of an "ideal" diagram. Fortunately, however, one of the essential features of a CASE tool, often referred to as "rubber banding", is the ability to retain (and stretch) connections between diagram elements even if they are moved. Thus, it would be straightforward for an analyst to improve the layout of diagrams which had been generated automatically.

V. Concluding Remarks and Further Work

We have shown that we have provided support for each phase of the information system design process, and we have noted that the meta-CASE environment we have used to generate related CASE tools provides the facilities required to build a fully **integrated** tool to provide support throughout the entire design process. This is subject of a further research project proposal, which will address issues such as:

- the refinement of data capture techniques
- an investigation of view integration issues
- building an integrated tool (text to generated code)

Whilst we have concentrated on Entity-Relationship modelling, and, in separate work, on the generation of code for Relationship databases, it was noted above that similar techniques are appropriate also for the design of object oriented systems.

Further, the data capture techniques we have explored in the context of generating entity relationship models could, we believe, be more widely employed. One obvious possibility would be in the analysis of legacy code, in order to support reverse engineering. Consideration of legacy code suggests that dynamic as well as static properties might be amenable to capture. This implies that similar techniques could be applied also to behaviour modelling within object-oriented environments.

References

[1] D. Bowers, *From Data to Database*, 2nd ed., London: Chapman & Hall, 1993, p11.

[2] P. Chen, "The entity relationship model: towards a unified view of data." *ACM Trans. on Database Systems* vol. 1 (1), 1976.

[3] D. Bowers, "Database schema improvement techniques for CASE tools", Proc. UKAIS 2000, April 2000

[4] R. Rock-Evans, *Data Analysis*, Sutton: IPC Business Press, 1981, p10.

[5] R. Abbott, "Program design by informal English description", *Communications of the ACM* Vol 16 (11), pp. 882-894, 1983.

[6] M. Lejk, D. Deeks, *Systems Analysis Techniques*, Hemel Hempstead: Prentice Hall Europe, 1998, p76.

[7] D. Bowers, *From Data to Database*, 2nd ed., London: Chapman & Hall, 1993, p52.

[8] T. Connolly, C. Begg, *Database Systems: a practical approach to design, implementation and management*, 2nd ed., Harlow: Addison Wesley Longman, 1999, p231.

[9] A. Silberschatz, H. Korth, S. Sudershan, *Database System Concepts*, 3rd ed., Singapore: McGraw-Hill, 1997, p29.

[10] G. Booch, "Object Oriented Development", *IEEE Transactions on Software Engineering*, vol. 12 (2), pp. 211-221, 1986.

[11] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, *Object-Oriented Modeling and Design*, Englewood Cliffs: Prentice-Hall, 1991, pp. 21-47.

[12] N. Juristo, J. Morant, A. Moreno, "A formal approach for generating oo specifications from natural language", *Journal of Systems and Software*, vol. 48 (2), pp. 139-153, 1999.

[13] P. Loucopoulos, R. Champian, "Concept acquisition and analysis for requirements specification", *Software Engineering Journal*, vol. 5, pp 116-124, 1990.

[14] I. Sommerville, P. Sawyer, *Requirements Engineering: a good practice guide*, Chichester: John Wiley & Sons, 1997, p64.

[15] W. Rzepka, Y. Ohno, "Introduction: special issue on requirements engineering", *IEEE Comp.*, vol. 18 (4), 1985

[16] M. Fowler, K. Scott, *UML Distilled: Applying the standard Object Modelling Language*, Addison Wesley Longman, 1997, p19.

[17] Y. Liang, D. West, F. Stowell, "An approach to object identification, selection and specification in object-oriented analysis", *Info Systems Journal*, vol. 8 (2), pp. 163-180, 1998.

[18] J. M. Sinclair, *Corpus, concordance, collocation*, Oxford University Press. 1991

[19] K. Aijmer , B. Altenberg, *English corpus linguistics: studies in honour of Jan Svartvik*, Longman, 1991

[20] J. Sager, *A practical course in terminology processing*, John Benjamins Publishing Co., 1990

[21] H. Fulford, S. Griffin, K. Ahmad, "Resources for knowledge transfer and training: the exploitation of domain documentation and database technology", *Proceedings of the 6th International Conference on Urban Storm Drainage,2, J. Marsalek and H. C. Torno (eds), Seapoint Publishing, 1993*

[22] H. Fulford, *Term acquisition: a text-probing approach*, Doctoral thesis, University of Surrey

[23] J. C. Sager, D. Dungworth, P. F. McDonald, *English special languages, principles and practice in science and technology*, Oscar Brandstetter Verlag KG. 1980, p. 199.

[24] J. Lyons, *Semantics*, Cambridge University Press, 1977

[25] D. A. Cruse, *Lexical semantics*, Cambridge University Press, 1986

[26] Lincoln Software Ltd., *Toolbuilder User Manual*, Macclesfield, 1998.

[27] H. Fulford, L. B. Work, D. Bowers, "Tools for information systems teaching: making a case for metaCASE", *Proceedings of the 7th Annual Conference on the Teaching of Computing, S. Alexander and U. O'Reilly (eds)*, University of Ulster, 1999

[28] P. Feldman, "A diagrammer for the automatic production of entity type models", *Proc. 3rd British National Conference on Databases*, pp 57-70, 1984.