COMPUTING LABORATORY

Security and databases: a methodological approach

J.E. Dobson

TECHNICAL REPORT SERIES
No. 264 September, 1988
Security and databases: a methodological approach

J.E. Dobson

Abstract

There is a sharp distinction between a software-oriented view of information and a social view. This distinction has particular implications for the design choices involved in the integration of a conceptual model of security with a conceptual model of data. Such integration raises some difficult philosophical problems.

A case study is examined to support the argument that reconciliation of security and privacy issues with data modelling issues through gross over-simplifying assumptions is not only a retreat from the problems we have identified, but can be positively damaging and counter-productive.

A methodological approach is required to analyse problems where values, organisational and social issues, and facts are intertwined. Relevant ideas from philosophy are raided in order to show how such a methodological approach may be structured, and the details of a design schema are outlined.

Series Editor: M.J. Elphick

© 1988 University of Newcastle upon Tyne.
Printed and published by the University of Newcastle upon Tyne, Computing Laboratory, Claremont Tower, Claremont Road, Newcastle upon Tyne, NE1 7RU, England.
Bibliographical details

DOBSON, John Edward


Newcastle upon Tyne: University of Newcastle upon Tyne, Computing Laboratory, 1988.

(University of Newcastle upon Tyne, Computing Laboratory, Technical Report Series, no. 264.)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE
Computing Laboratory. Technical Report Series. 264

Abstract

There is a sharp distinction between a software-oriented view of information and a social view. This distinction has particular implications for the design choices involved in the integration of a conceptual model of security with a conceptual model of data. Such integration raises some difficult philosophical problems.

A case study is examined to support the argument that reconciliation of security and privacy issues with data modelling issues through gross over-simplifying assumptions is not only a retreat from the problems we have identified, but can be positively damaging and counter-productive.

A methodological approach is required to analyse problems where values, organisational and social issues, and facts are intertwined. Relevant ideas from philosophy are raided in order to show how such a methodological approach may be structured, and the details of a design schema are outlined.

About the authors

Mr. Dobson is a Principal Research Associate at the University of Newcastle upon Tyne, with particular interest in a system approach to security. He has been active in the field since 1982 and is the author of more than a dozen papers.

Suggested keywords

DATA MODELLING
DATABASE
INFORMATION ENGINEERING
SECURITY
SOFTWARE ENGINEERING
CONCEPTUAL MODELLING

Suggested classmarks (primary classmark underlined)
Dewey (18th): 658.47 001.6442
U.D.C. 519.718 681.322.06
SECURITY AND DATABASES:
A METHODOLOGICAL APPROACH

John Dobson
Computing Laboratory
University of Newcastle upon Tyne
NEWCASTLE NE1 7RU
United Kingdom

ABSTRACT

There is a sharp distinction between a software-oriented view of information and a social view. This distinction has particular implications for the design choices involved in the integration of a conceptual model of security with a conceptual model of data. Such integration raises some difficult philosophical problems.

A case study is examined to support the argument that reconciliation of security and privacy issues with data modelling issues through gross over-simplifying assumptions is not only a retreat from the problems we have identified, but can be positively damaging and counter-productive.

A methodological approach is required to analyse problems where values, organisational and social issues, and facts are intertwined. Relevant ideas from philosophy are raided in order to show how such a methodological approach may be structured, and the details of a design schema are outlined.

This is Technical Report number 7 produced by the Computing Laboratory, University of Newcastle upon Tyne for the Royal Signals and Radar Establishment, Malvern as part of the research contract entitled 'Security Modelling for Distributed Systems'.

Copyright (c) University of Newcastle upon Tyne, December 1987.
1. INTRODUCTION

There seems to be growing understanding[1, 17] that to develop computer systems requires two very different activities which we might call 'software engineering' and 'information engineering'.

The universe of discourse for the software engineer is confined to data, and devices and processes for manipulating data, which is on the whole quite a tractable area of problems - or puzzles, since they have solutions. We can now accept that formal analysis and methods of proof are (or may become) the right tools for dealing with the central problems of software. Mathematics and logic are the key disciplines of the software engineer, who has to deliver a system demonstrably capable of effecting the desired data transformations with reasonable efficiency and economy.

The information engineer, on the other hand, has to examine the validity, effectiveness, meaning and reference of the information. The software engineer's ability to manipulate, store and distribute information correctly is no guarantee of the information's usefulness in the context of the business enterprise. The quality of information must be decided by the information engineer in the light of its fitness for some human purpose, by its ability to provide knowledge or guide and restrict people in their actions in a sense appropriate to the organisation and its policies. The information engineer must specify a system whose data have meanings in terms of the real world of the enterprise. We have yet to accept that logic and mathematics, though helpful, cannot furnish total solutions to the problems - not puzzles, since they have no solutions - of information engineering.

It is the thesis of this paper that there are two different views of 'security and databases' corresponding to the software engineering activity and the information engineering activity; that the information engineering view has not received proper attention despite the fact that it is logically prior to the software engineering view which is derived from it; and that the derivation is non-trivial because of a number of logical and philosophical differences between the two views. I shall argue that in order to combine issues of security with issues of database, one must choose between impoverishing the conceptual model of security down to the current capabilities of database and enriching the conceptual model of database to take into account the more complex requirements of security; and I shall offer some suggestions as to what the latter might require.

2. TWO VIEWS OF SECURITY AND DATABASE

Now whereas the problems of software engineering are receiving massive investment, the problems of information engineering are overlooked, or treated as matters of common sense, or are relegated to 'requirements capture'. This is a mistake, because badly specified systems, no matter how well built, will only serve the enterprise badly. Good specifications can only be developed on the basis of a good understanding of the real complexities of business and administrative decision-making. Unfortunately, the technical and financial emphasis on the software engineering viewpoint has led to a software-oriented view of the
role and nature of the specification activity. Most software systems are engineered upon
the assumption that the world is comprised of a set of objects about which we can obtain
objective knowledge and about which we can reason logically. Now this assumption is
dangerous when applied to a world of business and administration and in which security is
a concern. It makes more sense to think of security as an issue in an organisational
universe containing a host of different worlds created by different social groups with
differing motives and differing purposes, and therefore differing views of what is meant by
'security'. Information engineering must not make cosy simplifying assumptions but must
face the real world complexity and (somehow!) present to the software engineer a problem
devoid of this complexity. However, it is a mistake during this process to over-simplify the
complexity in order to make it more tractable in software engineering terms.

In order to explain what is meant about over-simplification, we will now consider two views
of the relationship between security and database taken in the context of an organisation
where both are matters of concern. It is possible to regard the subject of 'security and data-
bases' as one of the intrinsic security of databases, on a par with the consistency or
integrity of databases. According to this relation, which is the way the software engineer
would see it, the processes that occur in the world, and whose security is an issue, are func-
tional processes, defined by their inputs and outputs or pre- and post-conditions or what-
ever. For each kind of transaction a very limited and quite specific formal language is
used. Transformations follow rules which are quite specific to the application and consist of
simple arithmetic, selection and rearrangement of data elements. Questions of semantics,
validity and security are relatively simple and (this is the important thing) entirely within
the control of the people who authorise the system.

In contrast, it is also possible to regard the relation between security and databases as the
extrinsic security aspect of databases in supporting an organisation where security is an
issue. In this information engineering view, the interesting processes are social processes.
Although a tacit assumption is often made, in order to facilitate software engineering, that
meanings of words are intrinsic properties carried by expressions as they migrate from one
part of the system to another, this assumption is only true where there is a social process
tightly restraining the exchange of data and the context within which the data is inter-
preted. Since meanings do in fact shift, particularly in a more loosely-coupled social con-
text, it is necessary to take care to detect such shifts and take time to explain them in
order to avoid misleading the decision-makers. The abstract syntax available within the
DBMS approach is at present too weak to handle the real complexities of semantics at this
level.

From the viewpoint of the organisation, then, the question might be put "We are concerned
about security: what are the implications for our information sources, processes and reposi-
tories?" This is a general question in the social context, and might well give rise to ques-
tions about the powers and role of the database administrator (e.g. (s)he should not have
the power both to authorise and implement changes to the database). It would be an
oversimplification simply to transform the question into one of security mechanisms in the database, though they would clearly be relevant.

3. ON MODELLING

We shall start with a particularly simple and standard view of the modelling process that says that given a problem, one constructs two kinds of model of it: a conceptual model which is abstract and is designed to sort out the various issues involved (I like the term 'concept separator' to describe this model), and a concrete model (or set of concrete models) which is derived from the conceptual model and which is used to give practical guidance on implementation issues. So when we have two issues, we have two sets of models:

- Database problem
- Conceptual Model of Data
- Concrete Models of Data

- Security problem
- Conceptual Model of Security
- Concrete Models of Security

What we can then do is to compare the two sets of conceptual models. What we can not do is to use the conceptual model of data to derive (some aspects of) the concrete security model or use the conceptual model of security to derive (some aspects of) the concrete model of data. This is shown by the dashed cross below:

- Database problem
- Conceptual Model of Data
- Concrete Models of Data

- Security problem
- Conceptual Model of Security
- Concrete Models of Security

But it would appear that this is just what is being done in much of the current work on database security; and it is a mistake, because it ignores two basic facts:

(i) the conceptual models of data and security may be so different that it might not be possible to derive anything useful about a concrete model in the one domain from a conceptual model in the other domain.

(ii) Those parts of the enterprise that are interested in security because they have something of value that they wish to protect may have a completely different view of the structure of the enterprise and the processes that go on in it from those parts of the enterprise that store and manipulate data about it. Hence even within a single
domain there may well be more than one conceptual model, each with its own derived set of concrete models.

We shall in this paper be concerned only with the first of these observations, while recognising that the second is, in some respects, a more basic problem. In order to compare the conceptual models of security and database, it is convenient to classify the differences between them into four general areas, corresponding to differences in views on

- what there is in the world
- the nature of our knowledge about the world
- what happens in the world and the basis for actions
- the underlying logic of our discourse about the world.

The usual names for these categories are ontology (what there is), epistemology (how we know things), phenomenology (what happens), and logic (how we reason).

4. ONTOLOGICAL DIFFERENCES
or What is there in the world?

Most database models nowadays articulate the world into entities, relations, and attributes, together with a set of functions and higher-order functions over these, and possibly a set of rules or constraints on these basic constructs. Not all database models have all of {entities, relations, attributes, functions, rules} but most have most of them.

However, there is no single agreed set of basic constructs for security models. The classical model, which we will call the access model, sees the world as divided up into subjects and objects with the active subjects accessing the passive objects according to some set of rules embodied in an access matrix or state transition table.

A second model, again widely used, sees the basic construct as information which flows along channels connecting nodes. Associated with the information and/or channels and/or nodes is a colour (i.e. some semantics-bearing attribute) in terms of which security constraints can be expressed (e.g. black information can only flow between black nodes and along black channels). We will call this the flow model.

But there is another sub-culture which has a quite different view of security, which is regarded as an organisational issue rather than as a (software) engineering one. (See[10] as an example of security considered from this perspective.) It approaches the problem from the risk management and enterprise modelling side. The key terms in this vocabulary are words like information, value, risk, interpretation, stakeholders. Since this language may be a bit unfamiliar to some readers, a brief explanation is in order.

Information is seen as something that has an abstract value. Although abstract, this value is definitely finite, though it may be probabilistic. It represents something like the expected or potential loss that could occur from inadequate protection of the information. Furthermore, information is regarded as something that has to be interpreted in some
context in order to have a value. Thus one way to protect the value of information is to prevent a particular interpretation either by withholding the right context or by deliberately failing to withhold the wrong one (but, obviously, withholding the fact of its wrongness). We shall call this the value model, in recognition of the fact that its major concern is not protecting the information per se, but determining whether the cost of protecting the value is less than the expected value loss of non-protection. A word that is sometimes used in this connection is axiology, meaning a theory of values and, in particular, study of the ways in which values provide a basis for judging and deciding on actions.

Now it seems that the access model of security can indeed be mapped onto the database model in a fairly straightforward sort of way, though there are some logical problems to which we shall return, and some technological ones which we shall here ignore. Databases are quite capable of representing relations between subjects and objects, and attributes and rules and constraints on those relations.

It is harder to map an information flow model onto a database because most databases have no concept of fluxion.

As far as the value model of security is concerned, this seems so far removed from the way in which databases attempt to represent the world that it would appear there is almost no connection between them. Appearances can be deceptive, though, and we shall briefly look later, when we consider what happens in the world, at a value theory of security and what relationship it might have with a database.

The point to be reiterated is that just because we can map the access model of security onto the relational model of data, it does not follow that by so doing we have captured the notion of security even in a data-oriented organisation. It would be a mistake to let the ease of mapping onto the relational model dictate the choice of security model and its implementation if security, and not data, is the more important concern of the organisation. That would be to allow the conceptual model of data to force the concrete model of security.

5. EPISTEMOLOGICAL DIFFERENCES
or What is knowledge?

5.1. Types of Knowledge
It is convenient to start with a simple epistemology, by looking at four different kinds of knowledge. It is useful to distinguish

- **factual knowledge**: independent nuggets of information, such as who works for what department and what their salaries are, and so on.

- **procedural knowledge**: knowledge of how to do something, such as bake a cake, compute the millionth digit of π, or acquire a passport.

- **definitional knowledge**: knowledge of what things mean and how one determines meaning.
causal knowledge: knowledge of how the world behaves so that one can make useful predictions about it and provide helpful explanations about why it is the way that it is.

Now database theory would rank these kinds of knowledge in the given order as increasingly intractable. Databases really are very good at storing and retrieving facts. Procedures can be represented in a database, though there are still some research issues involved. Definitional knowledge at the present time is in the province of knowledge-based systems though it is fair to say that the techniques are still in their infancy and support a rather limited notion of 'definition'. Causal knowledge, which requires a model of the way the world works, is still an AI long term research area.

But for the security problem owner, these kinds of knowledge are probably ranked in increasing order of value, which means that the problem owner will regard them as increasingly important problems. From this point of view, database technology is best able to represent the least important kind of knowledge, and least able to represent the most important kind.

There is a difference also in the roles of partial knowledge and the so-called Closed World Assumption[16] in the domains of database and of security. Databases cannot well handle partial knowledge (John belongs to either the Computing Department or the Philosophy Department, but it is not known which) and makes the Closed World Assumption (If John is not recorded as a member of the Philosophy Department, then he is not a member of the Philosophy Department). Security issues often arise by virtue of partial knowledge (either Peter or Roger is a Russian spy, but it is not known which, and maybe both) and the CWA is in many instances just false. So any attempt to use a database to hold data about security matters is going to have to be treated with a great deal of care.

5.2. Computational and Semantic Theories of Information

There is a difference between computational and semantic theories of information[1] which is perhaps best explained by saying that a computational theory relates information to technical artefacts (e.g. processors, memories, switches, hardware and software components, people regarded as formal role players in some formal procedural structure) whereas a semantic theory relates information to decision-making often in the face of uncertainty, where the keywords are words like users, operations, plans, organisations and policies. The distinction that is being made is essentially that between information and the meaning of information. The software engineer talks about the generation, transmission, and reception of information without assuming anything about its interpretation. Software engineers can study how it can be efficiently stored or retrieved, or how to build a robust multi-user environment for an information base, and still assume nothing about the meaning of the information. Conversely, the information engineer can develop an elaborate semantic theory for temporal and historical knowledge, beliefs, goals, policies, etc., and yet say nothing about storing all this knowledge on a machine, or making it accessible to multiple
users.

The difference can be seen in classical databases, extended (or semantic) databases, and knowledge bases, as the following table indicates:

<table>
<thead>
<tr>
<th>PROCESSING REQUIREMENTS</th>
<th>simple business data processing</th>
<th>integrity constraints error recovery expert systems shared, federated DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA MODEL</td>
<td>classical models (hierarchical, network, relational)</td>
<td>semantic models knowledge representation frames</td>
</tr>
<tr>
<td>INFORMATION MANAGEMENT</td>
<td>standard DBMS</td>
<td>extended DBMS knowledge-based management</td>
</tr>
</tbody>
</table>

Database research has proceeded by defining the information processing requirements (for simple business DP), proposing some data models to satisfy the requirements, and creating a computational theory to support the development of efficient data management systems.

More recently, new requirements for data processing in more complex environments led to a need for richer "semantic" data models, and these models in turn gave rise to computational issues not addressed in earlier DBMSs (e.g. efficient support for referential integrity and foreign keys).

We can see the same pattern in knowledge-based systems, where the requirement for decision support and expert systems leads to issues in the development of knowledge representation schemata, which in turn have then to be supported by an efficient computational-theoretic KBMS (though at this time it is doubtful if one such really exists, even if inefficient atheoretic ones do).

Thus the computational theory relates the second and third rows of the table whereas the semantic theory relates the first and second rows.

Our thesis is that in general a lot more is known about the computational theory than about the semantic theory, and this is particularly true for security issues. It is less true for data issues. Thus we have a quite well understood computational theory of data, we are developing a semantic theory of data[8, 14], and although we have some computational theory of security, we are far from having a semantic theory of security. This is a pity, since the security mechanisms of which the computational theory treats derive their importance from the semantic theory, which treats of what knowledge we are trying to protect and why.

Possibly the most important difference between the computational and semantic theories is that they take place in different environments. Lytinen[14], following Ives[11], distinguishes the operational environment, which contains the resources, software, hardware, database, documentation and formal procedures necessary for the operation of the
information system from the organisational environment, which contains the enterprise policies and goals, task structures and semantics, and management style and culture which gives meaning to the information system. The computational theory is conducted in the operational environment whereas the semantic theory is conducted in the organisational environment.

6. PHENOMENOLOGICAL DIFFERENCES

or What happens in the world?

Phenomenology can for the purpose of this paper be defined as "the analysis of the intentional or cognitive acts we perform in everyday life; a putting into words of what is commonly and unthinkingly done without necessarily knowing how to describe accurately what is familiar in performance". [Macmillan Encyclopaedia of Philosophy, u.i. "Phenomenology".] We are, however, taking the liberty of extending the concept to include those acts performed on our behalf by computational agents such as DBMSs.

Nowhere is the difference between security and database more marked than in their phenomenology. In the world of database, the main thing that happens is that information changes, and the database notions of transactions, updates, access are all to do with information change (or lack of change). In the world of security, what happens is that things fail and people have to make decisions in the light of incomplete knowledge of the various modes of failure. Thus key elements in the phenomenological vocabulary of security are vulnerability, threat, countermeasure, risk.

The importance of phenomenology is that for each computer discipline it gives rise to a dominant theory of inference - which, with some licence, can be overstated as a 'theory of intelligence'. For logic programming, the theory of inference is resolution. For object-based systems, it is property inheritance. For databases, it is modus ponens. For the value theory of security, however, the key cognitive operations are hypothesis generation and consequence exploration. Now to the extent that intended uses of the information are consistent with the dominant theory of intelligence, the chosen form of information representation (e.g. rules, objects, relations) may be said to be appropriate. But in order to facilitate certain forms of inference, it does seem necessary to make other forms of inference more difficult. Any information representation schema must make some structural commitment to the inference process that it supports, and our phenomenological claim is that the inference process is primary and hence must govern the information representation. Thus security policies generate the need to distinguish various possible worlds (one of which may, or may not, be the actual world) and hence the logic needs to support alternative model sets. Some design and historical databases already do this, but the techniques of design databases have apparently not yet been applied to security or even examined for suitability for this purpose.

Now we are not claiming that the phenomenological differences are such that databases are incapable of supporting security management, merely that the relation between the
conceptual models has to be examined carefully because of the differences in role. There are some similarities here with the relation between databases and profitability. Like security, profitability depends on decision-making and risk analysis in the face of incomplete knowledge and uncertain failure modes. But something is known about the relation between profitability and database, as shown by the following (where -> is to be read "depends on")

Profitability -> spreadsheet models -> decision support systems -> database technology

BUT

Security -> ?? -> ??

In other words, what in the security world plays the same role as spreadsheet and decision support systems in the profitability world? and on what sort of information base would such a thing rest?? It may well be that the answer to the first question is fault tree analysis or something similar. (See [7] for something similar.) As to the second question, it is probably an open question whether the structure of databases is or is not most appropriate to the kind of inferencing involved in security-related processes.

7. LOGICAL DIFFERENCES

7.1. Model Theory and Proof Theory

There are two schools of database logician, corresponding to the two main approaches to the semantics of logic: the proof theorists and the model theorists.

The model-theoretic view of logic dictates that formulae have (truth-)values only with respect to interpretations, and a database is viewed as a first-order interpretation of the query language. The value of a relational calculus query is determined by those instances of its free variables that make the query true w.r.t. the interpretation specified by the underlying database. Similarly, insofar as an integrity constraint can be regarded as a first-order formula, a database can be said to satisfy this constraint iff the constraint is true w.r.t. the database as interpretation. Thus the database is interpreted as a model (in the standard Tarskian sense) of some set of integrity constraints and a query is some formula to be evaluated w.r.t. this model.

The proof-theorists, on the other hand, view a database as a set of first-order formulae, not as a model. Queries are formulae to be proved, given the database formulae as premises. Satisfaction of integrity constraints is defined in terms of consistency (again, in the logical sense). Intellectual energy is invested in obtaining algorithms for efficiently finding proofs.
The two schools are, however, in agreement that a database is extensional in nature, i.e. that the truth (however interpreted) of a database query language sentence is some function of the facts as recorded on the database.

Now it can be doubted whether security is extensional in nature. I believe that "It is secret* that ..." creates a modal context. (A modal context is one created by sentences like "I believe that X" which can be true even if X is false and false even if X is true, and hence the truth value of the sentence does not depend on the truth values of its components.) Another way of putting this in the language of formal logic is to say that security is intensional in nature.

In formal logic, there are two ways of treating modality. It can be regarded as a sentential predicate ('It is secret that X' is regarded as predicate of the sentence 'X'), or it can be regarded as a sentential operator, i.e. a function from sentences to sentences. ("It is secret that X' can be functionally derived from 'X', but the two sentences are otherwise independent of each other.)

In terms of the proof-theoretic view of database, it is naturally more appealing to take the syntactic view of modality which treats it as a predicate of sentences. Appealing, perhaps, but alas ill-founded, as the results of Montague[15] show. It does appear that any theory which is expressive enough both to be able to handle modality as a sentential predicate obeying a few simple and apparently reasonable axioms and can allow arithmetic to be done (and not much arithmetic either - enough to permit Godelisation, roughly) is inconsistent. Denning (e.g. in [3]) has pointed out that there are inconsistencies in attempts to apply semantic integrity constraints involving multi-level security markings of individual values in a database. What these results means in practice is that this kind of conundrum is in principle unavoidable if the view is taken that secrecy is treated as an attribute of a field value. However, if we regard secrecy as a sentential operator, then a model theory can be found using the standard semantics of modal logic, which involves 'possible worlds' or - for those who don't like metaphysical notions - alternative model sets. The practical application of such model-theoretic semantics is polyinstantiation, where the creation of each new instance (of a relation or of a tuple) is taken as creating a new 'possible world'. However, the logical soundness of polyinstantiation here asserted is only soundness in a logical sense; there are other difficulties with it, and not only technical ones of implementation.

7.2. Compositionality

There is another difference between security and database from a logical point of view, and that concerns the so-called 'aggregation problem'. The logical structure of the problem is that the sensitivity of a conjunction of propositions $P \& Q \& R \& \ldots$ may be higher than

---

* we are not here confusing security with secrecy; but if security subsumes secrecy as an issue, and if we are correct in our beliefs about secrecy as creating a modal context, then surely we are at least partly right in our beliefs about security.
the sensitivity of any proper subset of them, since knowledge of \( P, Q, R \) separately may enable the inference of some proposition \( S \) which one is not supposed to know.

The property of a composite logical formula that some attribute of it is deducible from the corresponding attributes of its components is known as compositionality, or Fregean compositionality after Frege who first enunciated it. What seems to be the case, then, is that the aggregation problem is a breakdown in compositionality. The logical structure of databases, on the other hand, is such that compositionality does hold, always.

Philosophers of logic argue about the status of the compositionality principle, about whether it is desirable, reasonable, and so on. This is not the place to enter that argument, but it should be noted that a debatable logical principle is involved here, and needs to be resolved, before we can marry security and databases. Unfortunately, philosophers and logicians have not yet succeeded in resolving the issue.

8. A CASE STUDY

To make some of these philosophical issues perhaps a little clearer, we shall now examine a short case study based on actual occurrences (quoted from Stamper[17] with slight changes in wording).†

A local government had one simple system for generating cheques to pay for repairs to houses certified by the social services department as 'unfit for habitation'. Another separate system registered the state of dereliction of property for urban planning purposes, the lowest category being 'unfit for habitation', which was used to identify areas suitable for redevelopment. Local government also had powers of compulsory purchase at the price of cleared land in areas designated for redevelopment.

The two computer systems were federated, using a shared database, in the interests of efficiency. No longer had houses to be inspected by officials from both the social services and city planning departments. What happened as a result of combining the two systems, each innocuous in its own right, was that short-term problems of poor housing were tackled by grants but with the side effect of creating long-term property blight problems by causing it to appear on the planners' maps as 'unfit for habitation' - i.e. derelict.

Political pressures in the interests of efficiency again influenced the situation. Lawyers, needing to search records in the course of house conveyancing, pressed for a flow of information between the national land registry and the local government planning department. Again, the link was made without consideration of the consequences.

The opportunity was seized by a software house to produce an expert system which was bought by all the major estate agents to help their clients make pricing decisions. The package used information on prices asked and paid, on the state of houses in different neighbourhoods, and on the urban planners' classification of areas. It helped the agents increase the volume of sales by

† In writing this section, I have benefitted greatly from reading Stamper's own discussion of his case study. However, my presentation is different from his, and there are several points where I disagree with his analysis, as he would doubtless disagree with mine. Obviously, therefore, Stamper is not to blame for the (mis)use I have made of his insights.
suggesting lower prices in areas which the expert system believed to be in decline. As a result, run-down but basically sound houses in areas known to be marked as derelict and scheduled for redevelopment were bulk purchased by potential property redevelopers at prices so low that domestic renovation costs were now uneconomic compared with the market value of the house.

Local government now had no other option but to compulsorily purchase the remaining inhabited properties. Thus the end result was that social workers, by using the right to help families with short-term repair problems, accelerated their displacement into homelessness and resettlement.

Clearly something had gone wrong. In essence, we shall claim, a whole series of mistakes had been made, many of them arising from the kind of confusions between database and security models that we have already mentioned.

From the database point of view, it might be thought that some information should have been privacy protected and was not. The question might be raised, what information should have been protected and from whom, in order to break the positive feedback flow resulting in social work decisions made in the best interests of individual families ultimately being in the worst interests of the whole community?

But the computational aspect of the problem in this example is not just one of the correct security and privacy classification of information. I shall now argue that it can be analysed more fundamentally as some combination of inappropriate ontology, epistemology, phenomenology and logic.

The first point to notice is that the notions of 'unfit for habitation' as used by the social workers and by the urban planners actually meant different things (immediate problems of condensation and dilapidated rooms as against long-term breakdown in the underlying fabric of the building and street infrastructure). Of course a proper systems analysis could have sorted this semantic confusion out, but it is just this kind of error that is positively encouraged by an ontology of physical objects and attributes. Questions of ontology bear upon the unavoidable tendency of the language we use to constrain our ways of thinking and the semantic assumptions we make. It is not at all easy to recognise and see through these constraints and assumptions, and many issues, particularly in the area of privacy of computerised information, spring from the assumption that information concerns entities with static attributes of fixed meaning, independently of the use of the information. As Wittgenstein[18] argues, however, the meaning of a term lies in its use, and hence a better ontology might be to start from the notions of actions performed by distinct agents. Thus the world seen by an agent (observer) comprises only the actions the agent can perform. There are obvious similarities here with the actor and object models of computation used in software engineering, suggesting that these models might be a better starting point for the implementation of systems involving privacy and other security issues than models based on entities and attributes.

The second point is that the case study exhibits an epistemological failure or mismatch. Epistemology concerns the kind and validity of information purporting to convey knowledge. The fact that a house is 'unfit for habitation' (however interpreted) is not an
isolated fact or nugget of information; it is a value judgement made by some individual within some model of the world. Valid value judgements are those that are fit for the purpose in the appropriate models, where 'fitness' is ultimately a social criterion. In terms of our simple epistemology introduced earlier, 'unfit for habitation' is, as information, more convincingly classified as definitional rather than factual, since the knowledge it conveys derives more from the meaning and implication of the terms that from its status as a statement of fact.

The third point is that there is a phenomenological error in the database schema. By classifying a dwelling as unfit for habitation, the social worker is not recording a fact but issuing a directive (to pay grants). The basis for the actions of recording and of authorising are quite different, and the criteria for judging their validity are quite different too. The urban planner, by recording a dwelling as unfit for habitation is again not recording a fact but giving a verdict on the suitability of the environment for redevelopment; and the criteria by which such an act is judged are quite different again. What is missing in the database is the ability to record who it was that assigned a particular value. As a general rule, when value-laden judgements are made, the valuer and the purpose for which the valuation is made are at least as important as the actual value arrived at - and this applies particularly to value judgements such as 'secret' or 'private'. Questions of accountability and responsibility mean that the recording and use of information have to be traceable to individual human actions.

The fourth point is that there is in the case study a good example of a logical error, in the form of dubious inferencing. The question at issue concerns the limitations that should be placed on the use of deductive methods, particularly in areas of social importance such as privacy. Quite apart from the issue of whether the real estate expert system should have had access to the planners' land information base, there is the issue of whether it was valid to use the given information for inferencing and thereby drive the market. One of the factors to be considered in a social context is the legitimacy of using a value judgement as a predictor for action. In a logical world, inferencing can indeed be captured in a set of syntactic rules, but in a social world, the logical conclusion of a chain of argument may be to recommend a course of action that is unacceptable for perfectly good political, economic or judicial reasons. (See Leith[13] for a powerful statement of this principle in the context of his critique of the attempts of Kowalski and others to codify the law in Prolog.)

Incidentally, it is interesting to speculate whether the planners could have foreseen the possible effects on house prices in the affected areas as a result of their categorisation, and explored the possible consequences of its publication. That might well have been a more appropriate use of the decision support technology.
9. A SYSTEMS DESIGN APPROACH

There is of course no software engineering solution to the information engineering problems described in the previous section. What we shall now do very briefly is to outline a methodology by which issues such as security and privacy can be clarified at the information engineering level in order that the software engineer can be given a well-grounded specification and can make a competent choice of security design and mechanism.

The fundamental questions in the design of an information system are "How can we transform the statement of a problem expressed in terms of human concerns and requirements into a blueprint for a technological solution?"; and "How can we demonstrate that the technological solution matches the original brief?" The designer can demonstrate the validity of a design only if it can be shown that all of its technical aspects and features can be traced back to the original requirements and that all the requirements have found expression in the design. Consequently there is a need for a design process that records the steps between the expression of the problem in the language of the human concerns (such as security and privacy), through the language of the information engineer, and into the language of the software engineer and ultimately of the system implementers.

A methodological approach to this problem has been developed at the University of Newcastle upon Tyne[4-6] and has been adopted by the ANSA Project[9]. We shall briefly outline its salient features and demonstrate its application to the topic of database in the context of an organisation in which privacy is an issue, by showing how it might have been used in the case study outlined in the previous section.

A schema of five levels of design representation is used. The transition from design at one level to design at a lower level increases the orientation of the design towards the target technology. The schema is organised so that objects introduced at one level of the design can still be perceived in the design at lower levels, so that the designer can track design decisions back to the original requirements.

The five levels are called the enterprise level, the information level, the computational level, the engineering level, and the technology level. We shall deal with them each in turn.

A design at the enterprise level describes the overall objectives of the system. This description specifies the activities that take place within an organisation using the system, and the interactions between the organisation, the system and the environment. The interaction between the organisation and the designers of the system is exposed at this level so that the processes of procurement, installation and maintenance (and their impact on the design) are represented. Further details of the human processes involved in creating the design at this level are given in [5].

A design at the information level shows a set of conceptual models of a solution to the requirements determined from an enterprise level design. Design at this level is concerned with processing, storage, communication and management functions, and is firmly within
the province of the information engineer. The people in the organisation can (in part) be represented as information resources, processors, and communicators. Including members of the organisation in the system model can reveal ways of restructuring the organisation so as to gain best advantage from the system. The representation of potential activities in the information processing, storage and communication tasks that can take place within the system is called a script.

Design at the computational level involves the refinement of the information resources and scripts identified at the information level into formal representations in specification languages, relational database calculi and algebras, and so forth. At this level, which constitutes the interface between the information engineer and the software engineer, the designer becomes more involved in the technical features of system components.

The transition from a design at the computational level to one at the engineering level requires the designer to make trade-offs between costs and physical constraints. Design at the engineering level explores limitations of the scope of a computational level design in the context of resource limits, desired system attributes (such as security, reliability, and performance), and topics such as scheduling and networking. In a distributed system, for example, the designer will map the structure of the computational level design onto a structure of networked computers and assign parts of the design to individual processors.

A design at the technology level will show the mapping of an engineering level design onto a selection of proprietary components. Design at this level deals with CPUs, memories, I/O devices, physical network interfaces, and various software components and packages.

10. THE CASE STUDY RE-EXAMINED

Let us briefly see how the application of our design principles might have helped analyse the case study previously discussed. We shall be concerned only with the first three layers; the engineering and technology layers are of less relevance to our purposes in this paper; since we wish to illustrate the general principles for designing a system in which security and privacy are of concern.

The issues to be examined at the enterprise level, then, are the activities which take place in the system. There are any number of methodologies, as for example the soft systems approach[2], for examining such activities and their interactions with the structure of the organisation. A good survey of available methodologies can be found in [14]. For the given example, the relevant activities include authorising and paying grants, clarifying urban areas, compulsorily purchasing dwellings, inspecting and certifying houses, searching local and national land records, analysing house price movements, predicting future price patterns, and so on. Note that at this level the key terms are all verbs. Nouns (names of departments, individuals, physical entities etc.) are conspicuous by their absence. That is as it should be; business and administration is best thought of in functional terms, and security and privacy can be regarded as constraints on possible behaviours. With each
major activity we can associate a notional actor who is responsible for performing that activity; later on, some actual (human or mechanical) processing object will be assigned to play that role.

What should come out of the analysis, as far as security and privacy issues are concerned, is an identification of the powers and limitations of each of the major actors involved, and of the authority and reporting structures between them. (This identification is sometimes called 'task semantics'.) It is the constraints on the activities of each of the actors identified at this level that should govern the structure of the database (which is decided at the computational level). Thus if it was felt undesirable that social services should have the power to influence the classification of land use, then the design at this level should exhibit no means by which the capabilities of the social workers can (directly or indirectly) influence the capabilities of the city planners.

Design at the information level is to decide upon the types of information processing and storage functions, and to produce a script that represents the potential activities that can take place within the system. At this level, then, the enterprise is represented as a collection of generalised information processors and information resources which can be given a script specifying the actions to be taken in response to events that may occur. For example, the certification by a social worker of a home as 'unfit for habitation' should be shown to result only in the authorisation of maintenance grants for the home. Concentration at this level on actions, events and the script as the basic components of the ontology should do much to prevent later abuse of shared variables that do not have shared semantics.

A little more detail of the general structure of any design at the information level will show its relevance to issues of security and privacy. A simple information level design is shown in the figure.
The environment of the system is seen at the information level as interacting with the system in three ways: as an event trigger for tasks to be performed, as a source of knowledge to be captured by the information resources, and as a source of disruption to the information processors. The system can only conduct those tasks which are identified at the enterprise level and described in the script; thus the script defines the system's response to all events in the environment anticipated by the designer. Disruptions are events which the designer has not catered for explicitly.

The information resources are objects which provide information access functions such as knowledge bases, databases, file systems (and, of course, people). The choice of information representations for knowledge is an issue for the computational level.

The co-ordinator encapsulates the planning and policy aspects of the system design. Its function is to bind together the resources needed to carry out the information processing activity and to assign a script to the information processors. It is responsible for responding to the environment by scheduling activities for the information processors and routing the necessary information from the information resources.

The information processors are objects which represent the information processing capabilities of the system, either computers or people. The processors follow the script assigned to them by the co-ordinator and obtain information from the information resources.

It can be seen from the diagram that there is no direct coupling between two information resources nor between two information processors, and all bindings are managed by the co-ordinator. There is thus the possibility of separation by suitable internal design of the co-ordinator. In terms of our case study, a major flaw in the system design was that the information resource shared between the social services department (considered as an information processor) and the city planning department (considered likewise) was poorly coordinated and its use was unscripted, leading to a bad design at lower levels. Aspects of security and privacy at the information level should be reflected in the design of the co-ordinator and in the structure and role assignment of the script. The co-ordinator may have therefore to include the functions of a reference monitor; the resources and processes to be separated and monitored are specified in the script.

Although it is possible to show a generic design at the information level, it is not possible to provide one at the computational level, since the major structural change at this level is the formalisation of the activities of information processing and of the definition of the information resources.

The computational level is the first point at which heterogeneity has an impact upon design, since the design team has the option to use different formalisms for the script and information resource representations in different computations. Activities will typically be formalised in a specification or programming language, of which there are a great number. Information resources may be formalised in various ways, as for example knowledge representations, database algebras, and document architectures.
The important point is that security and privacy issues are identified at the information level, and hence the conceptual model of security is a result of design at that level, although its formal expression is not designed until the computational level. On the other hand, the information level identifies just the sources, processors, and sinks of information; the data model (in the standard use of that term) is not designed until the computational level is under consideration. Design of the conceptual model of security should precede design of the conceptual model of data. In the case study, federation of already existing databases meant that the conceptual models of data had already been designed, and so any security or privacy issues arising from the federation could not govern the federated data model but had to be uneasily accommodated within it, thus violating our design principle. Another model that is not designed until the computational level is the kind of inferencing supported. The case study exhibits rather nicely what may happen when systems for aggregating and assessing numerous small value judgements are allowed to use inappropriate deductive methods and so produce a self-fulfilling prophecy. (The inappropriateness lies in the failure to incorporate value judgements about the result of lowering house prices; i.e. that it might be undesirable from a social community point of view.)

11. CONCLUSION
The conclusion to be drawn from this paper is a methodological one. In the design of any system in which social concerns such as security and privacy are important, the information engineer must proceed by tackling the major philosophical issues in the following order:

1) Clarify the axiology
The way we describe, and perhaps even perceive, the world depends on the actions we perform and the theory of values that provide the basis for judging of actions. There is a tendency in some quarters to have a very simple axiology to the effect that the potential loss due to violation of any security constraint is incalculable, and hence security breaches must be prevented at any cost. However, most business and administration cannot make such a sweeping and simplifying assumption, and have to employ a much more sophisticated theory of value. What exactly that theory is, often goes unstated in explicit terms. But our conviction is that until the enterprise knows what is of value, and on what basis that value is judged, it has no basis for the making of rational decisions - including decisions on the design of a system to support it in its mission.

2) Clarify the phenomenology
Once an axiological commitment to security and privacy as fundamental values within a system is made, and an appropriate theory of value outlined, implications follow for the responsibility and accountability of actions. The ontology, logic and epistemology should be structured so as to make reference to the agent responsible for deciding on the basic
entities or building blocks, the propositions and inferences, and the structure of the information representation. This means that questions of what happens, who does what, and what are the limitations and obligations on actions have to be decided before security models and data models can be constructed. As a consequence, any attempt to impose some particular security model before doing the phenomenological analysis is almost certainly a methodological error.

3) Clarify the ontology
Hidden beneath the choice of formalism in which a system is specified are the ontological assumptions about the basic building blocks of the world. In a world of social concerns, the assumption that individuality and identity of objects are non-problematic is often false (see Kent[12] for a stimulating discussion of this point). A logic of action avoids having to believe in an objective reality independent of viewpoint.

4) Clarify the logic
Increasingly, problems of security and privacy arise from the fact that decision-making is being made in a distributed environment where no-one fully understands who makes the rules, who exercises judgement, who knows the facts. All the participants have at best partial knowledge; and the overall effect of the distributed decision-making process urgently needs to be understood. The role of inferencing in the decision-making process cannot be clarified until best efforts have been made to clarify the other factors involving values, actions, actors, and the basic building blocks for the social construction of reality.

5) Clarify the epistemology
Knowledge always relates to some agent through whom or which the world is apprehended; and, as we argued earlier, the structure of the knowledge representation depends on the kind of inferencing it is designed to support and must include consideration of social aspects and limitations. The emphasis on the software engineering aspects of a system over-emphasises the role of knowledge in a system involving social concerns. The assumption that all we need is an accurate consistent record of the state of affairs of an objective reality is naive and often false, and the belief that the subject of 'security and databases' concerns some aspects of the mere technical correctness of an epistemic system is just about as wrong as it could be.
ACKNOWLEDGEMENTS

I wish to thank Mike Martin, Andrew Herbert and John Monk for all the discussions we have had and Brian Randell for his helpful comments on earlier drafts of the paper. Security work at the University of Newcastle upon Tyne is funded by RSRE Malvern, and I am grateful to Derek Barnes for his continuing support.
References


