Pigott, D., Hobbs, V.J. and Gammack, J. (2004) *Just below the surface: developing knowledge management systems using the paradigm of the noetic prism.* In: Burstein, F., Linger, H. (eds), 2004, *Managing Knowledge with Technology*, Australian Scholarly Publishing Pty Ltd, Kew Vic Australia. Pp 126-140.

Developing knowledge management systems using the paradigm of the noetic prism

Diarmuid J. Pigott

School of Information Technology
Murdoch University
Murdoch, WA 6150
d.pigott@murdoch.edu.au

Valerie J. Hobbs

School of Information Technology
Murdoch University
Murdoch, WA 6150
v.hobbs@murdoch.edu.au

John G. Gammack

School of Management
Griffith University
Nathan, QLD 4111
j.gammack@griffith.edu.au

Developing knowledge management systems using the paradigm of the noetic prism

Abstract: In this paper we examine how the principles embodied in the paradigm of the noetic prism can illuminate the construction of knowledge management systems. We draw on the formalism of the prism to examine three successful tools: frames, spreadsheets and databases, and show how their power and also their shortcomings arise from their domain representation, and how any organisational system based on integration of these tools and conversion between them is inevitably lossy. We suggest how a late-binding, hybrid knowledge based management system (KBMS) could be designed that draws on the lessons learnt from these tools, by maintaining noetica at an atomic level and storing the combinatory processes necessary to create higher level structure as the need arises. We outline the "just-below-the-surface" systems design, and describe its implementation in an enterprise-wide knowledge-based system that has all of the conventional office automation features.

Keywords: knowledge management, knowledge management system, intelligent organisation, noetic prism, spreadsheet, database, frames, late-binding hybrid system

1. INTRODUCTION

Creating a knowledge management system as the underpinning for an intelligent enterprise has immediate and tangible problems. The goals of the organisation, and separately the goals of the administration of the organisation, have to be encoded in such a way as to permit implementation at a very low level. A significant reason for this is that intelligent organisations have been modelled as organisms ever since Newell proposed the "Knowledge Level" in 1982 (Newell, 1982). The knowledge level was formulated by analogy with the *aspirational* level of awareness in an individual, as suggested earlier by Simon (1968) and substantially revised in Simon (1996). Yet from the beginning the challenge has been to identify this aspirational level aim and get it to be performed in a manner that can be assured at the *satisficing* level – the level at which a sufficient satisfaction of the goal has been achieved.

When this model is brought from the domain of theorised biological systems and applied to corporate entities, these two separate awareness levels are coalesced into the *applicative* level, the level at which declarative instructions are made. With this, the gap between goal and result – in effect, the planning stage – is lost, and with it the importance of the double conversion between the intentional and the operational. The distinction between the applicative and the imperative levels, so familiar from computer science, is not sufficient to describe what happens here: the aspirational has to be translated to the imperative, and then back to the satisficing level, before the effectiveness of any planning or modelling strategy can be assured. The problem for modelling and implementing systems for an intelligent organisation is that the establishment of a verifiable and repeatable process to ensure this translation has proved elusive: generating a universal list of aspirational-applicative to imperative instructions, or imperative to satisficing-applicative recognitors, is simply not achievable.

Another, hidden, problem caused by the merging of these two different awareness levels is a common naive view of the way in which computers work: an analysis whereby the necessary process of translating strategy to tactics, and thence to operational instructions, back to results and finally to a conclusive outcome is seen as one of value-added transformation of material from simple to complex. This is, at least on a superficial level, highly plausible, and the transformational model of data-information-knowledge and its variants is almost unquestioned in the knowledge management literature (see, for example Alavi & Leidner, 2001).

Elsewhere (Hobbs, Pigott, & Gammack, 2003; Pigott & Hobbs, 2001) we have argued that there is one base material that is the subject of computation and analysis, and that the common terms “information”, “data” and “knowledge”, far from being stages in a transformational model of computational enrichment, are in fact aspects of the same material based on the extent to which the principles of granularity (aggregation), shape (commonality) and scope (interrelatedness) inform the decision-making or processing of the material, and the complexity of the structure that it exhibits. We further argued that this material, which we term noetica, can be concurrently regarded as all three without contradiction: particular bindings (to data/shape, information/granularity, or knowledge/scope) are brought out through the noetica being analysed and modelled using particular tools and formalisms. We introduced the conceptual space of the “noetic prism”, a four-dimensional vector space, to explore the implications of this paradigm.

The challenge then was to investigate the extent to which the noetic prism could be used not only to design new systems that had the benefits of a clearer perspective, but also to examine the reasons why some established software worked in a satisfactory manner, or why others did not. The paradigm of the noetic prism enables us to clearly distinguish whether it is scope, granularity or shape that principally informs a collection of noetica, and it is a nontrivial problem to achieve a view of the noetica within which all three vertices of the prism are equally important. From this, the importance of late-binding evaluations of material in computing is obvious, as it is the purpose of the viewing, the tools of the viewer and the viewer's intention in the consultative process that determines whether the noetic material is seen as knowledge, information or data.

Three standard computing tools – frames, spreadsheets, and databases – may be seen as reflecting this trichotomy. In examining each of these, we shall see how each tool, as embodying a view for which one vertex is significant, relies on and subsumes noetical aspects from the hegemony of the other two vertices. Within an overall organisational management system created by integrating such tools as these, we will show that in repeatedly moving noetica from the hegemony of one vertex to another there is a genuine loss of content to the total system, and that this “lossiness”[1] is an inescapable product of the refinement resulting from complexification with respect to a single vertex.

The conclusion we draw from this is not, however, a nihilistic one, but a realisation that the answer lies in a system that incorporates all three approaches to dealing with noetica, and we describe such a late-binding, hybrid system in the third section of this paper.

2. EXISTING TOOLS

2.1 Frames

There have been many proposed representational structures for knowledge and knowledge management. Part of the problem for knowledge representation generally has in fact been the reluctance to have a widespread establishment of any such standard. This is not by accident either: different schools of knowledge representational thought have required a different kind of structure to match their epistemological stance or research paradigm. For our purposes, it behoves us to look for the closest system that has been given universal credence as a “knowledge base”: frames.

Frames, originally proposed by Minsky (1974; 1981) and their close cousin scripts (Schank, 1975), were designed to represent stereotypical situations or concepts that covered classes of instances. Attributes

[1] The term lossy, coined originally in the context of image compression, is generally applicable in describing encoding schemes where some detail or nuance is lost during conversion. Lossless is its lexical antonym, and lossiness the noun.

were stored as slots, filled with values, and frames were structured hierarchically, so that generalisation and specialisation could apply. This allowed default values to be inferred, without explicit representation, (although some provision was made for frame instance representations). Thus, in a typical example, the frame for bird would have the attribute “can fly”. Knowing that Tweety was a bird allowed the inference that Tweety could fly, and knowing birds were mammals, one could infer Tweety was also warm blooded. They broke down in cases where default values did not apply, adapting the classical example, if Tweety was a kiwi, the “can fly” value was not true. Frames, like other derivatives of first order logic, were best suited to efficient reasoning over static representations in closed worlds, and did not handle change or exceptions elegantly. They also required externally specified procedural semantics to become operative. The frame problem, (essentially knowing what stays the same and what changes) is a philosophical conundrum for the ages, and acutely shown by the representational limitations of such approaches, particularly in temporal contexts.

The value of frames largely lies in their power to represent hierarchical relations efficiently in domains with a naturally hierarchical ontology, such as accepted taxonomic structures, and to represent the default values as an expressive heuristic for general closed world reasoning. In Damon Runyon’s words, “The race may not always be to the swift, but that is the way to bet”, and it is generally true that birds can fly. Taxonomic structures however are not found in nature, and are rarely uncontentious or absolute. And to work they require acceptance as an ontology by all users and stakeholders as the best available structuring of knowledge, which is rare outside established scientific fields.

Contemporaneous research in the psychology of concepts and categories was moving away from classical, Aristotelian notions of category definition in terms of essential attributes, and Rosch’s work (e.g. Rosch, 1975) was seminal in this field. More sophisticated conceptualisations of category representation emerged from this line of work, based more on notions of constructive processes than of inherent defining attributes. The essential impossibility of representing all the possible attributes of instances in a class, which any object oriented approach would require, and the ambiguity and inconsistency that would result in an attempt to do so, means that the clarity of the frame formalism is at odds with its practical potential. Furthermore measures of similarity, which are sometimes used to determine classifications and hierarchical relations, have been discredited in the symbolic representation field by Medin’s work (Murphy & Medin, 1985), which shows their circular and arbitrary nature.

Computer scientists however still attempted to develop this essential approach to knowledge representation through various object oriented formalisms, and in the AI tradition, the cognitive architectures or intelligent environments of SOAR and CYC.

SOAR (Newell, 1982) remains current (now known as Soar since the original acronym became seen as less relevant) and is now in version 8. Its guiding ethos is captured in the following, taken from the Soar homepage (*Soar homepage*, 2003): “The ultimate in intelligence would be complete rationality which would imply the ability to use all available knowledge for every task that the system encounters. Unfortunately, the complexity of retrieving relevant knowledge puts this goal out of reach as the body of knowledge increases, the tasks are made more diverse, and the requirements in system response time more stringent. The best that can be obtained currently is an approximation of complete rationality.”

Also unfortunate is Soar's architecture that embodies older notions of memory, in particular an impenetrable long term memory of fixed associations. Such structures stem from the information processing era of psychology where equivalence between human and computer memory structures was adopted in cognitive modelling, and a correspondence to a fixed real world was assumed. This is basically the frame problem writ large, and as the representation becomes more elaborate to deal with its inherent problems, it loses the efficiency for which it was designed.

The recognition of an objective "level" above the symbol level was one of Newell's greatest insights (Newell, 1982) but formalising such in symbolic structure terms merely makes the symbolic more complex, and misses its essence.

Frames have a semantic close-coupling with the world, but not an analytic one. The beauty of their representational style lies in their selective simplicity and their eductive recursion. Their structural nature (being labelled) is insignificant, so the ad hoc selection of detail is not at all the same as the rigour of a spreadsheet – an absent value might be significant or not, as with the presence or absence of a slot. What frames do supply perfectly is a representation of intention and context, and when structural or statistical use is made of frames, this is lost.

2.2 Spreadsheets

The abstract nature of spreadsheets has been less examined that knowledge-bases or databases, partly because of their intensely practical nature, and partly because of the highly applicative interaction style. Their success is attested to by the fact that they were the application that took the small computer system and made it ubiquitous – the original "killer app". Certainly by far the majority of research into spreadsheets has had more to do with CHI and end user computing than with abstract modelling – the research cited below is atypical in its approach.

Yoder and Cohn (1994) have remarked on many similarities between spreadsheets and the intensional dataflow programming paradigm, pointing out such features as relative and intensional addressing, their eductive late-bindingness, and their lack of high-level abstraction. However, recent changes in the programming paradigm for (e.g.) Lotus and Excel have moved them away from the dataflow model and towards the event driven paradigm, through the use of an object-imperative language for the scripting; but this has not altered their ability to be close-coupled and highly granular in their representation of the world, so this conception of their applicative nature as programmatic is insufficient.

More significantly, a dataflow paradigm ignores the fact that spreadsheets are interactive environments, perhaps the only truly successful conversational programming environment. Better comparisons could be made with the paradigms of statistical systems (such as S, R, and Minitab), mathematical languages (such as Matlab, Mathematica, and Octave) and simulation systems (such as Simscript, CSSL, and ROSS)[2]: these are all conversational in nature, but each lacks a significant feature set of the spreadsheet as well as sharing aspects.

An interesting approach to the theoretical basis of spreadsheets has been taken by researchers such as Harel (1988) and independently by Burnett et al. (2001) as visual languages, wherein the visual/interface aspects are treated with the respect they are due. Of all of these efforts, it is Harel's proposal of the Statechart – a generalised case of the spreadsheet based on a late-binding temporal and spatial reflexive hygraph – that holds the most promise. However, that formalism does not do justice to the interactive calculative aspects of spreadsheets which are perhaps their most significant feature. Burnett's approach is to define a superset of spreadsheets and other similar systems, which she calls "spreadsheet languages", to refer to "all systems that follow the spreadsheet paradigm, in which computations are defined by cells and their formulas" (Burnett et al., 2001, p.2). She also points out the similarity between the cell in a spreadsheet and the pure conception of the object as originally conceived by Kay (1984), and quotes him as stating that "a cell's value is defined solely by the formula explicitly given it by the user" (Burnett et

[2] These are only a representative sample; there are many more of each type of system. (See Chambers, 1998; Eaton, 2001; Mitchell and Gauthier Inc, 1993; Kiviat, Villanueva, & Markowitz, 1969; MathWorks Inc., 1992; McArthur, Klahr, & Narain, 1985; Ryan, Joiner, & Ryan, 1985; Venables, Smith, & R Development Core Team, 2002; Wolfram, 1988.)

al., 2001, p.2). This call to a visual language system as a general case of the spreadsheet is persuasive, but misses out on the applicative formalism that was noted by Yoder and Cohn.

Taking the alternative route and describing spreadsheets as simple systems in their own right also leads to problems in their description. In essence, they are self-adjusting matrices of interdependent variables that are analogues of book-keeping worksheets, but this analogy does not extend easily beyond a naïve two-dimensional form. However, within this domain we can certainly see that spreadsheets model the world far more closely than any other computational system.

The matrix in a spreadsheet is representational of a decision to measure and record common features across instances. This commonality of features is akin to the selection of fields in database design, but the physical layout of the spreadsheet permits implicit inheritance of value conveyed as absence, while the ad hoc addition of columns and rows give a much later binding of the situation (Hobbs & Pigott, 2001). In addition, the reflexive dataflow aspect afforded by cell formulae together with charts, solvers, pivots etc, gives the ability to achieve continuous application of the value-adding process. This instantaneous responsiveness is lost when noetica is taken from spreadsheets and imported into databases, while the use of spreadsheet-derived noetica as a basis for a frame-set would never be at a level that supported the types of processes for which frames are created.

2.3 Databases

Databases are the result of a theoretically grounded approach, while still remaining fundamentally practical in outlook as the research in the late 1950s that led to the creation of the entire field had as its impetus some fairly substantial real-world problems (the Minuteman data, the Apollo program, the military spares system). However, there is a good deal of disagreement as to the extent to which of the individual formalisms (tree, hierarchy, network, relation, or object) has the best map onto reality, a quandary which is not made any easier by a pseudo-historical account often made of the transcendence of one database style over another (flat file replaced by hierarchical replaced by network…). What is an aspect of all models is the recognition of common feature-placement with regard to the representation of the world, and the feature-placement with regard to assertions of identity in that real world. The main variation between the different models lies in the formalised representation of the relationships between any two objects (either those that are similar or co-represented, or those that have a real-world connection, or those where requirements for stability and reliability necessitate the creation of virtual entities).

This commonality of features is a penultimate form of abstraction – we are not looking for *absolute* features, but for *common sets* of features, and for reliable mechanisms for recording those features (such as attributes). We need rigidity in a system to enable the set-related process of sub-selection or set-intersection in propositional form that is at the heart of database-representation. There is a kind of systemic, chicken-and-egg problem here: we need the common expression of features to establish the applicability of databases, yet seek out those items which suit a database treatment by looking for just such features already present. We are not seeking the same reactive close coupling as we sought in the spreadsheet here: we are trying to get a representational set of features for a set of objects across a universe of measurement or observation.

By the same token, this rigidity and universality makes for a rather static picture of the world, which in turn means that databases can be seen as less responsive than the protean spreadsheets and frames. To this extent, they are far more closely tied to the Aristotelian tradition of essence + existence + accidence = proposition than are other forms of computing. The true power in making a viewport of the noetica via a database lies in the comparable arrays exhibiting similar features. It is this universal characteristic of propositions – this perpetual present of statement made – which makes them difficult to verify from within the application as one can with spreadsheets. To counter this, databases have been extended in

various ways, adding transactioning systems, triggers, close-coupling extenders, and calculated fields, while at a more theoretical level distinctions have been made between real-time and standard databases (Ozsoyoglu & Snodgrass, 1995), active and passive (Paton & Díaz, 1999) or between kernel and characteristic entities (Codd, 1979).

Despite all of this, the database is not responsive enough to act as the real-time core for an adaptive enterprise support system. We have remarked elsewhere (Hobbs & Pigott, 2001) on the steps taken to adapt the content of spreadsheets to define and then fill tables in databases (including identifying the nature of unspecified information, normalisation, and error-location). The transfer back from database to spreadsheet is far cruder – the resultant will always be structurally defined, regardless of header rows and formatting. And this placement by structure will always have a coarse granularity with respect to the world, and the order in which data is seen to occur in the series will be unreliable and deceptive (the irrelevance of order is a pre-condition for a logically defined relation; Codd, 1970).

2.4 Limitations of the three tools

When we examine noetica using these three tools, we find we gain insight into the nature of the noetica at the expense of its flexibility with regard to the movement between prism vertices. However, that insight is derived from the tools' adherence to the nature of their significant vertex, not with any form inherent in the tools. For what gives the close-coupling to the spreadsheet is its granularity: its ability to represent the aggregated noetica in a late-binding calculable form. Similarly with frames, the power of the frame/slot/value contextualisation is representative of the ability of the labelled graph to create complex structures that have no structural bounds, which is due to the nature of the interconnectedness present in them. And just as the benefits of scope and granularity rather than any individual formalism or format give the spreadsheet and frames their power, it is the tabular representation of commonality and its coalescence into set-related attribution that gives power to the database.

Thus we see the need for a system designed to have these tools' representations of the noetica, without the restriction of their formats. What is missing from the database – the variability and semantic significance of structure and late-binding calculability of value – should be added to its firm representation of set and set-membership through commonality of structure. What is missing from the spreadsheet – semantic significance of grouping and rigour in naming – should be added to its powerful late-binding and close-coupling. What is missing from frames – a sense of objective representation and a measurement of similarity that transcends a priori judgements – should be added to the semantic richness of labelling and attribution that characterises the frame.

In addition, what is needed to ensure that the system as a whole is representative of the host organisation is what is missing from all of these tools – a lossless process of integration and a preservation of lineage. Without this, any new application would be simply to create new, as yet undiscovered shortcomings. By beginning with the noetic prism as a means of verifying systems and applications, we can ensure that no such shortcomings occur. We can get the power of each of these systems (and others as well, as we shall mention below) without being trapped by either their formalisms or the individual nature of their lossiness on conversion.

Declarative or applicative paradigms for explanation are the most effective in terms of their power and simplicity; they are also closest to the standard declarative usages of conversational explanation. However, as we said at the beginning of this paper, it is this very high-level nature that makes them all the more difficult to use as a predictive framework, and there has to be an intermediary layer between applicative and imperative before any implementation. By the same token, the largely context-free operating environment of the algorithm and the database is abstracted in a different direction, towards generalised process of an infinitely and reliably repeatable nature (Figure 1).

Figure 1. Mediation process between declarative and imperative levels of querying.

By recognising the cyclical (rather than two-way) nature of this mediation process, we can see that there is a particular set of requirements (ones that can be easily represented in a computational process) that is unique to each value-adding interaction with the noetica.

All of the tools we examined displayed this mediated multiple-level data structure, when they were called upon to give advice – the spreadsheet through summarisation, the frames through automatic deduction, the database through querying. Here we had a high level abstraction for all of them that was largely set-related, while below them we had a computer model that was hidden, and dealt with in an indirect way. While noticeable in the spreadsheet and frames, the same process happened in the database – it was merely hidden by the identical nature of the recursive structure.

We can also see these features in other systems that share the significant vertex of the three examined tools: project management software, statistical analysis, and market research tools derive their power from granular close-coupling; bibliographies and qualitative analysis systems benefit from a frames-like late bound application of titles and labels to slots, while pattern-based software design invokes notions of commonality in applications. The principle features of different applications lies in their ability to expose the noetica with respect to a particular vertex, and the extent to which they permit further manipulation in this manner.

Our challenge, therefore, is to work out a system that can support all three approaches equally at a more basic level and to develop tools for making the higher level one through a series of program applications.

3. A LATE-BINDING, HYBRID SYSTEM

We have established the requirement for the storage of noetic simples together with the mechanisms for viewing them in a rich value-added way. The noetic prism shows us how these applicative qualities can be modelled as a vector function (Pigott & Hobbs, 2001), and that we can therefore realise the creation of high level semantic structures through the combination of template structures, late-binding matrix representations with rigorous commonality, and the addition of calculated and dependent fields.

In analysing the requirements for our late-binding hybrid system, we find we need

- core membership of an entity type (to permit attribution and standard behaviour)
- event occurrence and nature (to preserve order, occurrence and context)

- interrelation between entities, entity patterns and attributes (to permit reconstitution of over-normalised entities)
- topic interrelation (to preserve natural meaning and interdependence)

It is important that the system is designed to permit the late binding of overall structure: this involves modelling meta-structure as well as metadata. By organising the data in a particular structure we can argue for a preservation of all of the features described above, without losing the capability of close-coupling with the intentional (aspirational in the sense of Simon, knowledge in the sense of Newell) level.

This approach to database structure is consistent with the concept of over-normalisation, wherein sets of attributes that are consistently structured (i.e. similar) across a database are modelled as entities rather than the norm of significant entity fields (Figure 2). More complex views than normal would be necessary to reconstitute the data. Some might even involve coding and re-composition of the disparate elements into pseudo-tables, or in-memory entities with virtual fields.

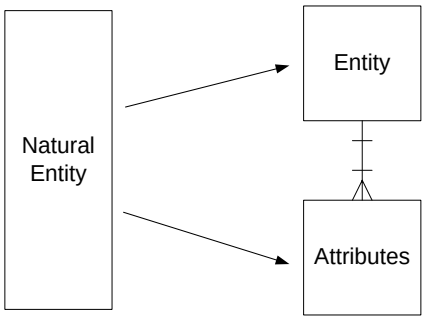

Figure 2. Over-normalisation of attributes into entities.

An example of this approach would be if co-occurrence of events were significant: all of the entities that had date components would be split into two disjunct components, those attributes that indicated dates and those that indicated static values. A view based on date could then be made: a snapshot of all events to do with a particular day could be retrieved, with the happenings linked only by co-extension of date, rather than propositional entailment (in effect, a query based on granularity rather than on shape). In a similar manner, details to do with documents, participation in projects, materials and so on could be taken as entities – in fact a similar clustering could be made around any aspect of the data set that is of interest. Figure 3 illustrates this principle with a hypothetical knowledge based management system (KBMS) that merges many and diverse time-based information (stock tickers, press releases, government notices, interest rates) for the purpose of analysis.

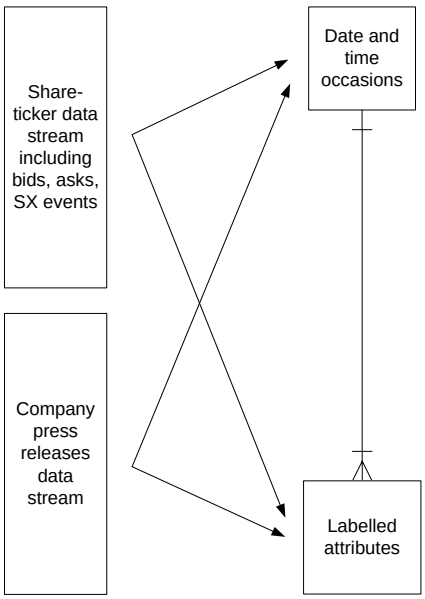

Figure 3. Example of over-normalisation and recombination of entities and attributes.

The next stage in our refinement process enables us to delegate more significant operational roles to the various entities, as shown in Figure 4. Here we have a system where all fields are delegated to grouped field-entities: the fields-entities are gathered together by reason of similarity of nature, and linked as in the fashion of a LISP database in an ad hoc recombinant structure.

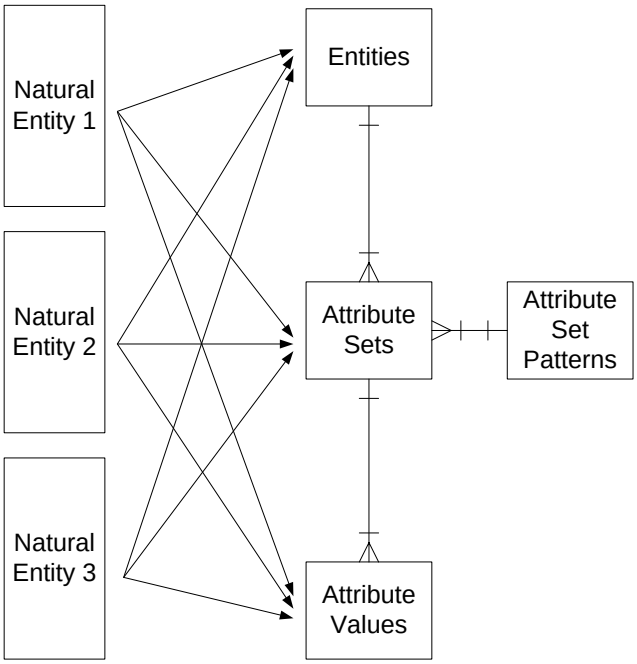

Figure 4. A hyper-normalised design for maximal recombination.

Now we can design our complete KBMS: the structure we have enables particular reflexive display of columns of data to the individual in the manner of a spreadsheet, but at the same time grounding the material displayed in the rules given by referential integrity. Records can have varying structure, but the

act of adding or removing fields from access and display would be constrained. In a similar manner, code-blocks can be stored in fields to enable calculated or summarised data to be shown, search, edited, and sorted without compromising the system.

Using this system, the data itself can be viewed in any manner deemed necessary or useful. For example, it could present a spreadsheet interface, an address book, project interfaces, calendars or thesauri to the user by judicious use of dynamic HTML and the use of a mediated scripting layer, while the nature of the data-based application would be hidden from the user (Figure 5). This is why these may be properly called just-below-the-surface (JBS) systems, by analogy with just-in-time delivery (JIT) systems. In JIT systems, reliance on the mechanism as a whole gives an economy of effort and capital investment through late-realisation of assets (in inventories) and interpretation (in computer systems). The guarantee of supply is ensured by the system, so the users and even the administrator can be declarative in their control. In JBS systems, a similar declarative control is possible through the hermeneutic agency of the system. A JBS system relies for its rigour on referential integrity (for the shape vertex), taxonomic and ontological consistency (for the scope vertex) and existential entailment (for the granularity vertex), while the binding code and templates that make searching and display possible are informed by the interconnectedness of the prism as a whole. This entire mechanism lies just below the surface, and the user (and indeed the programmer, given a sufficiently sophisticated API) may remain in ignorance of them.

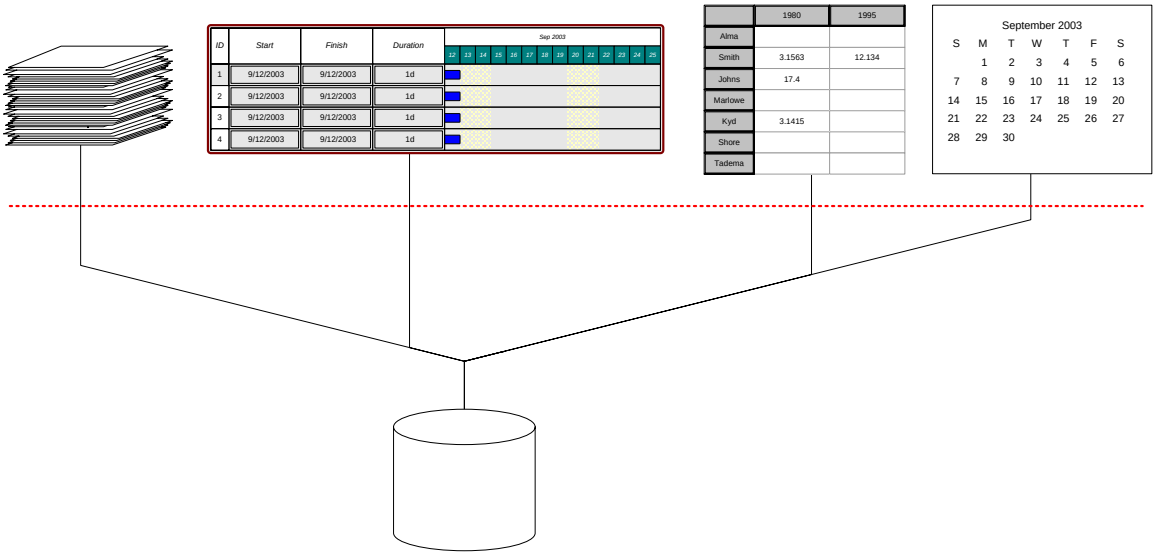

Figure 5. A "just-below-the-surface" (JBS) system presenting address book, project management, spreadsheet and calendar interfaces.

The important features of a JBS system are a polymorphic scripted thin client that is delivered both functionality and data by a protean black-box system. The tight integration of the various "skin" applications means that it is possible to have a fully late-bound approach to system use, as well as system design. The *same resources* would be viewable on the screen in whatever view the user considered best suited their needs. This type of close-coupling is different from the interconnectedness required of knowledge representation through directed labelled networks, yet they too can be accommodated in this approach to database design. Although at first sight the JBS approach may seem unduly reliant on the system in providing a support mechanism, and may make for a sense of unease about concomitant complexity and slowness, it should be remembered that reluctance to rely on the mechanism of foreign keys and referential integrity delayed the acceptance of the relational model, despite the obvious benefits

that the model gave. With the JBS systematic, acceptance of a complex enabling mechanism permits the delivery of much greater power.

A system conforming to this design (Pigott, 2002) has been developed that had all of these features, and in addition had high level strategic decision support tools such as Bayesian analyses of alternate paths, critical path analyses and workflow optimisation. A revolutionary outcome of the system has been that the separation of dynamic, labelled structures (complete with semantically-rich metadata) from the actual content has enabled the repackaging of entire business processes down to the skill, event timing and document delivery level. This has radical implications for office automation, as it enables the intent of the work – the aspirational aspect, as it were – to be separated from the delivery, which as a satisficing component may diverge from it.

4. CONCLUSIONS

In this paper we have examined how the principles that are embodied in the noetic prism paradigm can illuminate the construction of practical knowledge management systems. We have drawn on the formalism of the noetic prism to examine existing successful computer processing systems, and seen how they attain their unique characteristics, and how their shortcomings arise from their domain representation, not from idiosyncrasies of design. We have suggested how a hybrid system might be designed that draws both from the lessons learnt from these systems, and from the first principles of the prism itself.

By maintaining noetica at an atomic level, and storing the combinatory processes necessary to create higher level structure as the need arises, we have shown how the functionality required of KBMSs can be created via a late-binding combination of these two elements. In this way, any representational formalism is only ever one extraction process away from atomic noetica, and movement between different viewports is lossless. Finally, we have outlined the "just-below-the-surface" systems design, and shown how it has been implemented in an enterprise-wide knowledge-based system that has all of the conventional office automation features.

5. REFERENCES

Alavi, M., & Leidner, D. E. (2001). Knowledge management and knowledge management systems: conceptual foundations and research issues. *MIS Quarterly, 25*(1), 107-136.

Burnett, M., Atwood, J., Djang, R. W., Gottfried, H., Reichwein, J., & Yang, S. (2001). Forms/3: A first-order visual language to explore the boundaries of the spreadsheet paradigm. *Journal of Functional Programming, 11*(2), 155-206.

Chambers, J. M. (1998). *Programming with data: a guide to the S language*. New York: Springer.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM, 13*(6), 377-387.

Codd, E. F. (1979). Extending the Database Relational Model to Capture More Meaning. *ACM Transactions on Database Systems, 4*(4), 397-434.

Eaton, J. W. (2001). *GNU Octave Manual*. New York: Network Theory Ltd.

Harel, D. (1988). On Visual Formalisms. *Communications of the ACM, 31*(5), 514-530.

Hobbs, V., & Pigott, D. (2001). *Facilitating end user database development by working with users' natural representations of data.* Paper presented at the Managing Information Technology in a Global Economy: IRMA International Conference, Toronto.

Hobbs, V. J., Pigott, D. J., & Gammack, J. G. (2003). Inaccuracy, ambiguity and irrelevance: an analysis of the nature of quality in knowledge management using the noetic prism. In F. Burstein, and Linger, H. (Ed.), *The Role of Quality in Knowledge Management: Proceedings of the Australian*

Conference on Knowledge Management and Intelligent Decision Support (ACKMIDS 2002) (pp. 42-54). Melbourne, Australia: Australian Scholarly Publishing, Melbourne.
Mitchell and Gauthier Inc (1993). *Advanced Continuous Simulation Language (ACSL) Reference Manual.* Mitchell and Gauthier Associates (MGA) Inc.
Kay, A. (1984). Computer software. *Scientific American, 251*(3), 53-59.
Kiviat, P. J., Villanueva, R., & Markowitz, H. (1969). *The Simscript II programming language*. Englewood Cliffs, N.J.,: Prentice-Hall.
MathWorks Inc. (1992). *Matlab*. Englewood Cliffs, NJ: Prentice Hall,.
McArthur, D., Klahr, P., & Narain, S. (1985). *The ROSS Language Manual*. Santa Monica, California.
Minsky, M. A. (1974). *A Framework for Representing Knowledge*. Cambridge: MIT.
Minsky, M. A. (1981). A Framework for Representing Knowledge. In J. Haugeland (Ed.), *Mind Design: Philosophy, Psychology, Artificial Intelligence* (pp. 95-128). Cambridge, MA: MIT Press.
Murphy, G. L., & Medin, D. L. (1985). The role of theories in conceptual coherence. *Psychological Review, 92*, 289-316.
Newell, A. (1982). The Knowledge Level. *Artificial Intelligence, 18*(1), 87-127.
Ozsoyoglu, G., & Snodgrass, R. T. (1995). Temporal and Real-Time Databases: A Survey. *IEEE Transactions on Knowledge and Data Engineering, 7*(4), 513-532.
Paton, N. W., & Díaz, O. (1999). Active database systems. *ACM Computing Surveys (CSUR), 31*(1), 63-103.
Pigott, D. (2002). Omnium. Perth, Western Australia.
Pigott, D. J., & Hobbs, V. J. (2001). *The Noetic Prism: A new perspective on the data-information-knowledge complex.* Paper presented at the Western Australia Workshop on Information Systems Research, University of Western Australia.
Rosch, E., and. Mervis, C. (1975). Family resemblances. studies in the internal structure of categories. *Cognitive Psychology, 7*, 573-605.
Ryan, B. F., Joiner, B. L., & Ryan, T. A. (1985). *Minitab handbook* (2nd ed.). Boston: Duxbury Press.
Schank, R. C. (1975). *SAM - A Story Understander* (Research Report 43). New Haven, Connecticut: Dept. of Computer Science, Yale University.
Simon, H. (1968,1996). *The Sciences of the Artificial* (Third ed.). Cambridge, Massachusetts: The MIT Press.
Soar homepage (2003). Retrieved 10 September 2003, from the World Wide Web: http://www.eecs.umich.edu/~soar/
Venables, W. N., Smith, D. M., & R Development Core Team (2002). *An Introduction to R*. New York: Network Theory Ltd.
Wolfram, S. (1988). *Mathematica : a system for doing mathematics by computer*. Redwood City, Calif.: Addison-Wesley Pub. Co., Advanced Book Program.
Yoder, A. G., & Cohn, D. L. (1994). *Observations on spreadsheet languages, intension and dataflow* (Notre Dame Computer Science and Engineering: Technical Report TR 94-22): University of Notre Dame.