Application-Driven Architecture: A Case Study of SYNTEL™

Peter E. Hart

Syntelligence
1000 Hamlin Court
Sunnyvale, California 94089

Abstract

The special requirements imposed by important classes of financial risk assessment applications have driven the development of a new expert system architecture. We discuss the relation between the demands of these applications and features of the architecture, and describe our experience with large-scale, deployed products that have been built using this new approach.

Historical Context

The 1980's can already be seen as the decade in which the expert system technology developed in research institutions during the 1970's first began to be exploited on a substantial commercial scale. Naturally, the initial applications of expert systems were guided by the lessons of the 70's, and made as much use as possible of the familiar rule-based and frame-based representations that had been first developed to address tasks in medical and other technically-oriented domains. By adopting the strategy of heeding these lessons, exploiting established technology, and judiciously selecting applications, a significant number of expert systems were successfully fielded [Feigenbaum (1988)].

We describe here our experience over the past five years with financial expert systems where this strategy was quickly seen to be infeasible, chiefly because the existing technology was poorly matched to the demands of the selected financial applications. As the application requirements became better understood, our design response led to a new architecture that departs sharply from the rule- and frame-based architecture that dominates--and indeed is virtually synonymous with--expert systems today. The new architecture, termed an *active functional system*, is embodied in a programming system called SYNTEL™ that has been used as the basis for two lines of commercial products: the Underwriting Advisor™ system and the Lending Advisor™ system. Technical details of SYNTEL have been reported elsewhere [Reboh and Risch, (1986); Duda, et al., (1987); Risch, et al., (1988)]; in this note we focus on the application setting, describing only enough about SYNTEL to make the discussion (hopefully) intelligible.

Design Requirements for Financial Expert Systems

There are many opportunities for applying expert system technology within the broadly-defined financial field. [See, e.g., Clifford, et al. (1986); Friedman & Jain (1986); Dhar & Croker (1988); or Leinweber (1988) for examples and commentary.] The great diversity encountered makes it difficult to define a useful set of system requirements that adequately characterizes all such financial applications but, as always, this difficulty is greatly eased by focussing on a selected subset.

Our own interest has centered specifically on two applications: underwriting commercial insurance risks and assessing commercial creditworthiness. These two applications are similar in many respects. They both are concerned with estimating or assessing numerical and symbolic quantities (among which "financial risk" is an important but not unique instance); both tasks prove, upon investigation, to involve comparable styles of reasoning; both involve substantial amounts of case-specific, or transient, data as well as more permanent reference data; both serve business users having comparable attitudes and needs; and, while both applications are largely self-contained, they both must exist in the real world of commercial data processing with its attendant interface and operational requirements.

Even this brief summary of application requirements suggests a number of technical implications for system

architecture. We review here the most important of these, noting where appropriate why conventional rule- or frame-based approaches do not appear to be well-suited to the needs.

Assessment vs. Classification Reasoning Styles
A fundamental reasoning task for our applications is to assess (or estimate) the value of a continuous quantity such as "Projected Profit". Any number of such sub-assessments--whether numerical or symbolic--are then weighed together in order to perform the overall "apples-to-oranges" assessments that are part of every business analysis (e.g., as in assessing the importance of a firm's poor financial performance against its attractive market and strong management team.) This style of reasoning contrasts with classification (or diagnostic) reasoning styles in which a selection is made from among a finite set of alternatives, and which are so well-addressed by rule- and frame-based representations.

The distinction between estimation and classification might superficially appear unimportant; one could imagine discretizing a continuous variable, thereby converting estimation to classification and making a rule-based approach an obvious one. Unfortunately, this becomes increasingly unattractive with increasing depth of sub- and sub-sub-assessments, because the number and complexity of rule antecedents needed to identify various combinations of assessments increases exponentially.

Inexact Reasoning
We expect the system to offer advice (more properly, assessments) in the face of imprecise knowledge or missing input data. Obviously, we prefer to have a smooth transition in the "exactness" of computed assessments as input data is made more complete. This suggests the need for a principled approach for dealing with missing data, default assumptions, and imprecise knowledge. More specifically, because of the importance of continuous variables, it suggests using probability distributions to represent the several forms of inexactness.

User Initiative
We have observed that users insist on retaining the initiative in moving through the system. This suggests that the overall control scheme will have to be data driven, like the standard spreadsheet applications that are so familiar to business users. However, data-driven approaches can lead to serious inefficiencies when used on very large knowledge bases; means for controlling excessive computation must be part of the basic design.

"What-If's?": Controlling Side Effects
Users frequently engage in "what-if?" explorations in which input data is repeatedly entered, modified, and then returned to its initial state. Understandably, users expect system-generated outputs to be consistent with restored inputs regardless of intervening computations. This suggests that the basic internal computations must be referentially transparent, so that restored output values depend only on the restored input values and not on sequence- or side-effects.

We note in passing that applications built on conventional expert system approaches usually embody procedural escapes to the underlying implementation language, a practice that considerably complicates the problem of satisfying this requirement.

Data Management and Data Models
The number of variables needed to describe a single case (e.g., a borrower) is large enough to require a significant data management facility. This transient data is accessed and modified so frequently that it is desirable for efficiency reasons to have the data management facility be an integral part of the expert system, rather than to rely on calls to an external database server. For theoretical and pragmatic reasons, it is most desirable to have a uniform data model for both transient data, permanent reference data and expert knowledge.

The User Interface
Business users operate in a world of standardized business forms. To be acceptable, the user interface will have to mimic this familiar world as closely as possible.

The SYNTEL Programming System

We sketch now the principal elements of SYNTEL, drawing particular attention to the relation between architectural features and the application requirements noted above. Further details can be found in the cited references.

Value Tables and Probability Distributions
SYNTEL uses a *value table* as its means for storing permanent reference data, case-specific input data, and the assessments derived therefrom. A value table can be regarded as a single database relation having an arbitrary number of keys and exactly one non-key column. For example, the value table Revenue[*Year*,*State*] might hold historical and projected values of Revenue for several years and states. Formally, value tables fit the functional data model [see, e.g., Gray (1984)].

Each row, or *instance*, of a value table can contain in its non-key column either an exact (numeric or symbolic) quantity, a probability distribution over that quantity, or a token indicating that the quantity is entirely unknown. Value tables are supported internally by efficient data structures and indexing schemes, in recognition of the corresponding need to handle substantial amounts of transient data.

Primitive Functions

SYNTEL uses a collection of some 60 primitive functions to represent how input data are to be combined into progressively higher-level assessments. In addition to the usual classes of arithmetic, logical and string manipulation functions, there are special functions that express how sub-assessments are to be weighed against each other to produce a higher-level assessment. There are also functions that modify the structure of value tables (e.g., by reducing the number of keys.)

Primitive SYNTEL functions are considerably more complicated than the functions found in conventional procedural languages. First, notice that a single function maps a set of value tables into a single value table; as a simple example, the function **Difference** could be used to map Revenue[*Year,State*] and Cost[*Year,State*] into Profit[*Year,State*]. All functions make use of a join-like operation when computing derived instances. In addition, since function arguments can be probability distributions, primitive functions must map a set of (assumed independent) distributions into a derived distribution.

Primitive functions for knowledge representation mesh with value tables (note that value tables mathematically are extensional functions.) Taken together, this representation has proven to be well-matched to application requirements: Most importantly, it accommodates the reasoning style characteristic of the domain (including graceful degradation when data is absent); it supports "what-if's?" because primitive functions are referentially transparent; and, with the control strategy described below, it provides the user-initiative capability that has proven to be essential.

Forms: The User Interface

Syntel incorporates a major sub-language called the *forms system* for defining the appearance, structure and behavior of typical business forms. The role of the forms system extends much beyond being a conduit for input and output information. Indeed, it is an integral part of the overall control mechanism, as described next.

Data-Driven Control and Efficiency Issues

SYNTEL operates, with a few exceptions, in a data- driven mode. Whenever a value table element is changed (whether by the user or by accessing external data through system interfaces) SYNTEL re-computes functionally dependent values to maintain consistency with the new independent values. For this reason, we term SYNTEL an *active* functional system; each function springs into action when its arguments change. As noted earlier, this provides user flexibility at a potentially large computational cost.

SYNTEL employs several interacting mechanisms to control this cost. Using information derived both from the current state of the user interface as well as from an extensive compile-time analysis of functional dependencies, the run-time system effectively implements a strategy that is easy to subscribe to but less easy to implement: viz., identify the minimal set of logical dependencies, compute only those dependencies that are required to support the current display state, and compute them only once.

User efficiency is at least as important as system efficiency. SYNTEL can conditionally display objects, a capability that allows an interactive session to be dynamically tailored to reflect the current state and thereby shield the user from irrelevant information or data requests.

Implementation Status

The SYNTEL formalism has been fully implemented since mid-1986, following three years of development, and supports a family of knowledge bases that define the Underwriting Advisor and Lending Advisor product lines. These products were originally developed on Xerox 11xx workstations. However, they are delivered to end users in a cooperative (or distributed) processing environment consisting of an IBM System/370 mainframe under MVS/XA using CICS and an IBM PC/AT or PS/2 workstation.

Each knowledge base contains several thousand value tables and about the same number of functions. A single value table might hold hundreds or even--in exceptional cases--over one hundred thousand instances. The depth of sub-assessments (more formally, the depth of function composition) can exceed 300. We know of no principled means for directly comparing functions with rules or frames, although for reasons described earlier it does seem clear that a much larger number of rules would be needed to represent equivalent expertise. Regardless of comparisons, these are by any measure large expert systems; careful attention to efficiency at all levels of design and implementation is mandatory.

Creating SYNTEL and all its surrounding software, as well as creating the various underwriting and lending knowledge bases, has not been inexpensive; over five years and well over one hundred person-years of direct technical development effort have been invested to date.

Deployment and Measures of Success

The Underwriting Advisor and Lending Advisor products based on SYNTEL have been purchased by insurance companies and banks in the US and abroad. Noting that: (a) the software system price ranges from half a million to several million dollars; (b) internal expenses of purchasers make total installation cost rise above product price alone; and (c) each purchase was made only after a sophisticated institution conducted a lengthy evaluation and analysis, one might well conclude that the innovative SYNTEL functional architecture has been successfully deployed. This conclusion is buttressed by the observation that the products go far beyond typical "back office" automation; they directly affect the "front office" decision-making process that lies at the core of every financial institution.

On a deeper level, things are less simple. The roll-out schedule for conventional large-scale system software in the financial industry is measured in calendar quarters or years, not weeks or months; expert systems are unlikely to shorten this timetable. Once fully deployed, a period of years is required before enough history accumulates to judge the consequences of the decision-making process (Did the loan eventually go bad? Did the building burn down?). Even then, it is difficult to establish objective standards of evaluation because of ever-changing business conditions: e.g., the changing business cycle, changes in tort law, changes in personnel, or changes in institutional policy.

For these reasons, a statistically valid analysis of results based on large, multi-year sample sets is beyond reach at present. Lacking that, we are forced to rely on less formal evaluations based on our current users' experiences in processing actual lending and underwriting cases. On this non-statistical basis we have seen remarkably favorable results: Users at all levels of experience have reported many examples of "I would have overlooked (or misinterpreted) the significance of a critical piece of data regarding...". More importantly, system-generated assessments have been in full agreement with the judgments of senior credit and underwriting officers. At this juncture, it is fair to say that the issue of "Does the system work?" has been resolved affirmatively.

Summary and Conclusions

We have emphasized that the real-world requirements imposed on financial risk-assessment systems differ in many important ways from what might be encountered in more traditional application domains. The unique architecture of SYNTEL arose not from an innate desire to invent something new, but rather as a design response to those requirements as we came to understand them.

The payoff from all this invention and effort comes in many forms, some of which still lie in the unknowable future. At present, the most tangible is a profitable and growing company.

Acknowledgments

The writer is merely the chronicler of, and contributor to, the efforts of a talented group of colleagues too numerous to name.

References

Clifford, J., Jarke, M., and Lucas, H. C., Designing expert systems in a business environment, in *Artificial Intelligence in Economics and Management*, (L. F. Pau, ed.), Elsevier Science Publishers, Amsterdam, 1986, pp.221 - 232.

Dhar, V. and Croker A., Knowledge-based decision support in business: issues and a solution. *IEEE Expert* 3,1 (Spring 1988) pp.53-62.

Duda, R.O., Hart P.E., Reboh R., Reiter, J., and Risch, T., Syntel: using a functional language for financial risk assessment. *IEEE Expert*, 2,3 (Fall 1987) pp. 18-31.

Friedman, J.Y. and Jain A., Framework for prototyping expert systems for financial applications. *Proc. AAAI-86* (Philadelphia, August). Morgan Kaufman, Los Altos, 1986, pp. 969-975.

Feigenbaum, E., Nii, H. P., and McCorduck, P., *The Rise of the Expert Company*, Times Books, New York, 1988.

Gray, P., *Logic, Algebra and Databases*, Ellis Horwood/John Wiley and Sons, New York, NY, 1984.

Leinweber, D., Knowledge-based systems for financial applications. *IEEE Expert* 3,3 (Fall 1988) pp. 18-31.

Reboh, R., and Risch, T., Syntel: Knowledge programming using functional representations, *Proc. AAAI-86*, (Philadelphia, PA, Aug.). Morgan Kaufman, Los Altos, 1986, pp. 1003 - 1007.

Risch, T., Reboh, R., Hart, P.E., and Duda, R.O., A functional approach to integrating database and expert systems, *Comm. ACM*, 31, 12 (December 1988) pp. 1424 - 1437.