Chapter 2

Literature Review

In this chapter, we first provide an overview of DFA. In the next section, we review the Modeling Language which is usually adopted to construct DFA models based on stochastic programming. (Birge, Dempster, Gassmann, Gunn, King, and Wallace, 1987; Yu, Ji, and Wang, 2003) In the last section, we introduce the concept of Decision Support System, and give a brief description about the software currently in the DFA market.

2.1 Dynamic Financial Analysis

One of the tasks of property/casualty executives is the assessment of the risks and rewards associated with strategic decisions. Too often, insurance executives must rely on intuition rather than systematic analysis when making these decisions. Even when financial decisions are based on a careful evaluation of the risk factors, the outcome of those decisions can be highly uncertain. Today, a systematic approach to financial modeling exists which projects financial results under a variety of possible scenarios, showing how outcomes might be affected by changing business, competitive and economic conditions. This approach has been called Dynamic Financial Analysis, or DFA.

DFA is a systematic approach based on large-scale computer simulations for the integrated financial modeling of non-life insurance and reinsurance companies aimed at assessing the risks and benefits associated with strategic decisions. (Kaufmann, Gadmer, and Klett, 2001) DFA is not an academic discipline by itself. It borrows many well-known concepts and methods from economics and statistics. It is part of the financial management of the firm. The objectives of DFA are helping management in:

- strategic asset allocation,
- capital allocation,
- performance measurement,
- market strategies,
- business mix,
- pricing decisions,
- product design,
- others.

DFA is a combination of software, methods, concepts, processes and skills. Skilled people are the most critical ingredient in carrying out the analysis. In Figure 2-1, we use the American Re-Insurance Company's Risk Management System (ARMS) (Berger, Adam, and Madsen, 1999) to describe an example of the methods and tools that are necessary for carrying out DFA. However, the structure referred to here is generic and it does not describe specifically any one of the DFA tools available in the market, but it identifies all those elements that are typical in any DFA model.

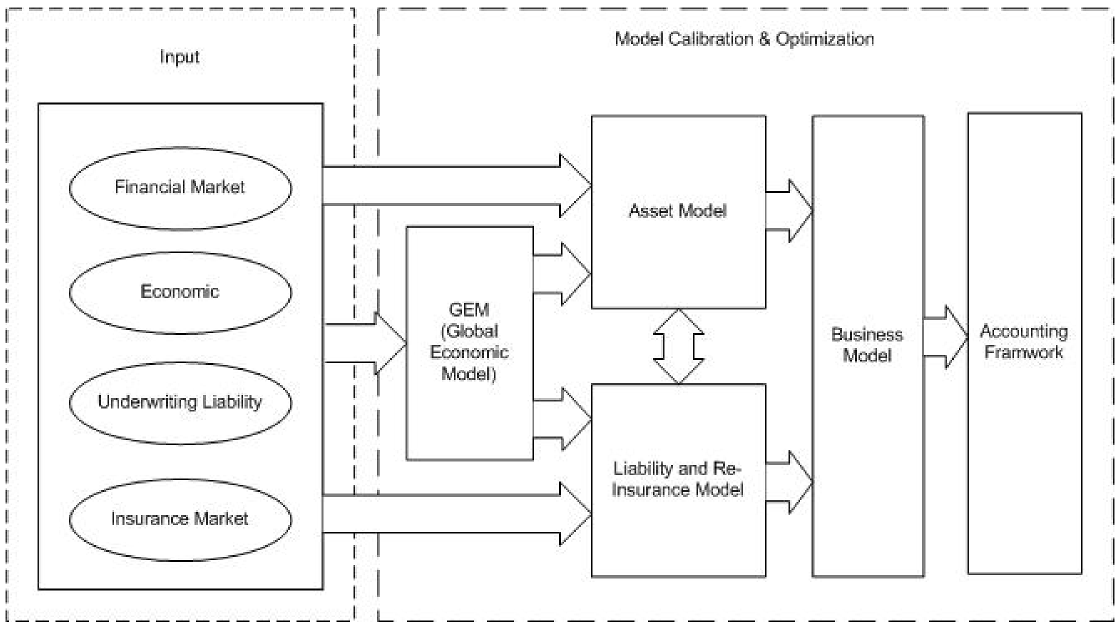


Figure 2-1 American Re-Insurance Company's Risk Management System (ARMS) (Berger, Adam, and Madsen, 1999)

A workable model must be a simplified version of reality. The Input includes the facts about the state of the company and industry at the time of the simulation. The Global Economic Model (GEM) is a scenario generator generating plausible time series outcomes of future economies based on user specifications and parameter settings. The GEM comprises stochastic models for the risk factors affecting a company (e.g. interest rates, inflation, stock market returns).

The outputs of GEM are fed to the Asset Model as well as the Liability and Re-insurance Model. These two models project different asset and liability classes along each economic scenario. The Business Model considers the underlying strategy of the business managers. It models the decisions that management makes as the business moves forward through time. This also includes any change in asset allocation or in re-insurance structure. The Accounting Framework refers to accounting implications.

The Model Calibration and Optimization expresses that finding suitable parameters for the models to produce sensible scenarios is an integral part of the DFA, and the simulation results should be used to readjust the strategy for the optimization of the target values of the company. Figure 2-2 (Kaufmann, Gadmer, and Klett, 2001) takes a clearer view of this concept.

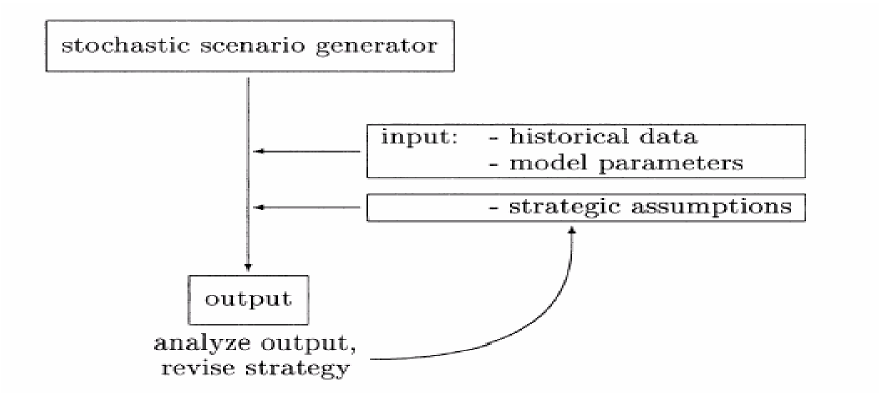


Figure 2-2 Main structure of a DFA model (Kaufmann, Gadmer, and Klett, 2001)

DFA incorporates feedback loops and management intervention decisions into the models. Management should analyze the output in order to improve the strategy. For example, if a given scenario shows that the loss ratio is unacceptably high for a line of business, then the model will assume that rate level and other underwriting decisions will be made by management. This can be repeated until management is convinced by the superiority of a certain strategy.

DFA grew in the late 1990s out of practical needs rather than academic research. The main driving force behind the genesis and development of DFA is the related research committee of the Casualty Actuarial Society (CAS). Their website (http://www.casact.org/research/dfa/index.html) provides a variety of background materials on the topic, in particular a comprehensive and easy-to-read handbook (DFA Committee of the Casualty Actuarial Society, 1999) describing the value proposition and the basic concepts of DFA. A fully example of a DFA, with emphasis on the underlying quantitative problems, is given in (Kaufmann, Gadmer, and Klett, 2001), whereas (Lowe, and Stanard, 1996) describes the development and implementation of a large-scale DFA decision support system for a company.

2.2 Modeling Language

Geoffrion (Geoffrion, 1987) identified two major problems confronting the management science/operations research (MS/OR) community. First, doing MS/OR tends to be a low productivity activity. Following factors contribute to this problem—

1) Three distinct representations are used typically for each model: a "natural" representation suitable for communication with people without special training in MS/OR, a mathematical representation suitable for analytical use, and a computer-executable representation (Fourer, 1983). Such multiple representations are inefficient of their redundancy, susceptible to inconsistency, and they demand too many different skills to complete even small projects.
2) Interfacing models with advanced solvers traditionally has been a laborious task requiring specialized skills.
3) Most modeling software addresses just one among the many kinds of models that arise—e.g., just linear programs.
4) Available modeling software typically serves just one or two of many phases of the total life-cycle associated with model-based analysis and system.

A second problem facing MS/OR is that managers and policy makers call for model-based assistance too infrequently. MS/OR professionals and their work often are incomprehensible to non-specialists. It is not surprising that managers are reluctant to use models that are both confusing and expensive to build. Unless this environment changes, MS/OR models will not achieve widespread use among non-technical managers.

The problems just enumerated call for a new generation of modeling systems with the following desirable features:

a) Modeling languages are designed to represent large and complex models using a few relatively simple statements (Geoffrion, 1987; Fourer, Gay, and Kernighan, 1990; Brooke, Kendrick, Meeraus, and Raman, 1998).
b) Modeling languages are designed to support the entire modeling life-cycle (Geoffrion, 1987; Geoffrion, 1989).
c) Modeling languages are designed to allow the accumulation, sharing, integration, and reuse of data, models, solvers, and derived knowledge (Choobineh, 1991; Brooke, Kendrick, Meeraus, and Raman, 1998).
d) Modeling languages are designed to improve the productivity and managerial acceptance of MS/OR activities (Geoffrion, 1987; Fourer, Gay, and Kernighan, 1990).
e) Desktop implementation with a modern user interface (Geoffrion, 1987; Dotzauer, and Holmstrom, 1998)

As mentioned above, the human modelers describe programs in a readable and symbolic form, such as the familiar algebraic notation for variables, constraints, and objectives. For the computer, an explicit algorithm is required to solve the problem. In the earliest, MPS standard which was named after an early IBM LP product was adopted as the computational representations of the corresponding mathematical model. (Kuip, 1993) The main drawback of MPS was its difficulties to be understood by human. Thus any application of mathematic programming should involve translating the one form to the other.

Matrix generators (Fourer, 1983) were the first step to facilitate the formulation of mathematical problems. Matrix generators were special computer programs which computer could execute to produce a program in the algorithm's desired form. However, matrix-generator form is not a modeler's form as previously defined. It is instead a distinct intermediate form intended primarily to facilitate translation to an algorithm's form. Because modeler's form and matrix-generator form are dissimilar, converting from the one to the other--which must still be done by a person--is not a trivial job. Indeed, writing a matrix generator is a job of computer programming, and has all of the characteristics of any programming task.

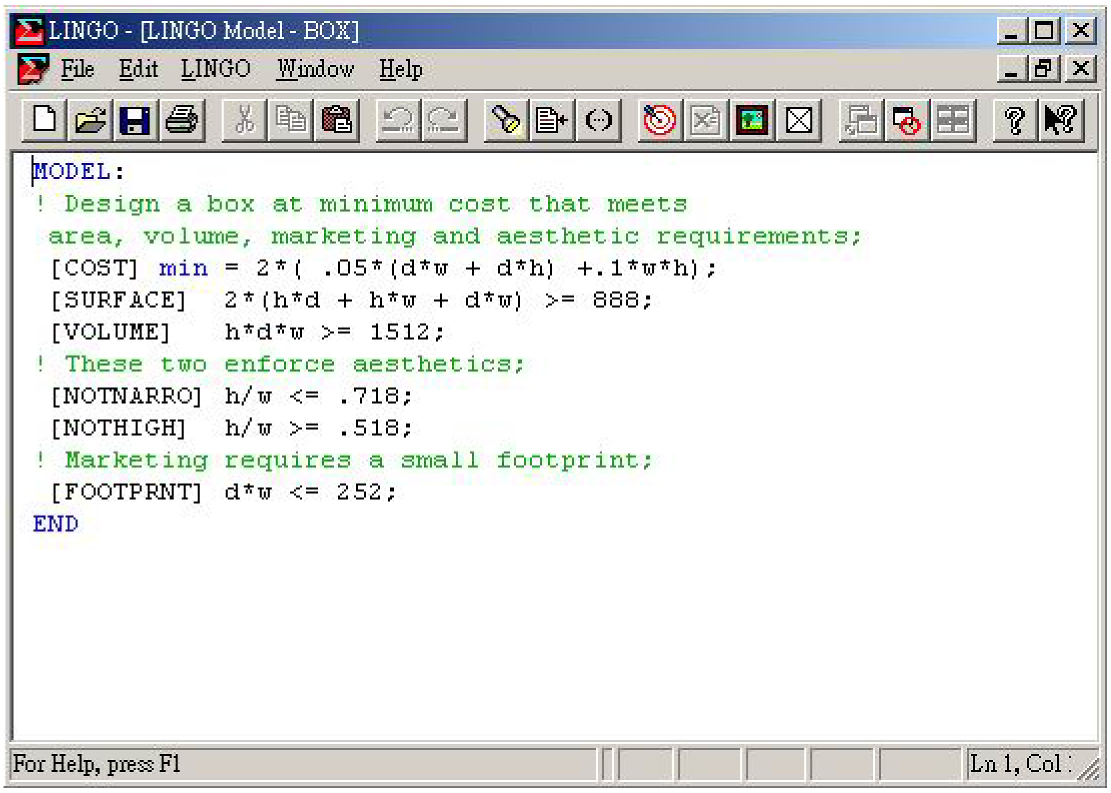
There is also a quite different approach to translation, in which as much work as possible is left to the machine. The central feature of this alternative approach is a modeling language (Fourer, 1983) that is founded on a simple idea: the modelers should deal with the computer directly in modeler's form. A modeling language is not a programming language; rather, it is a declarative language that expresses the modeler's form of a program in a notation that a computer system can interpret.

Unfortunately, however, because these forms are ambiguous and complex (as are natural languages generally), and important elements of their notation--notably subscripts and Σ signs--are incompatible with ordinary computer hardware, they are neither readable nor translatable by foreseeable computer system.

Nevertheless, an existing modeler's form is a sensible starting point for the design of an attractive and workable modeling language. Fourer also made several important statements on the design philosophy of modeling language. The variety and complexity of the existing modeler's form must be reduced so that every expression has an unambiguous syntax and meaning. Notation must also be altered to employ a standard computer character set.

Take present modeling languages for example, AMPL (Fourer, Gay, and Kernighan, 1990) and GAMS (Brooke, Kendrick, Meeraus, and Raman, 1998) have a powerful indexed-sum ("SUM") notation that corresponds to the algebraic use of Σ. The syntax of certain expressions is tighter and less ambiguous, as in the use of "t in 1 . . T" to mean t = 1 . . . . . T. Modeling language also enforces a certain amount of organization on the various parts of the model, again to avoid ambiguity and to facilitate translation.

Another point should be made about modeling language is the design of translators. A matrix-generator language is intended only as an intermediate between the modeler's and the algorithm's form; hence, such a language can be designed for ease of compilation and execution, and the more complex aspects of its translation can be left to people. A modeling language, by contrast, is exclusively an understandable modeler's form as Figure 2-3 indicates, so that the heart of a modeling language system is a translator that converts an ML model plus data to an algorithm's form.



```
MODEL:
! Design a box at minimum cost that meets
 area, volume, marketing and aesthetic requirements;
 [COST] min = 2*( .05*(d*w + d*h) +.1*w*h);
 [SURFACE]  2*(h*d + h*w + d*w) >= 888;
 [VOLUME]   h*d*w >= 1512;
! These two enforce aesthetics;
 [NOTNARRO] h/w <= .718;
 [NOTHIGH]  h/w >= .518;
! Marketing requires a small footprint;
 [FOOTPRNT] d*w <= 252;
END
```

Figure 2-3 Modeling language in LINGO

Fourer (Fourer, 1983) suggested that it is useful to distinguish two "phases" in the work of a modeling language translator:

1. The analysis phase reads the modeling language description of the model and determines how the algorithm's form is to be written.

2. The generation phase reads the explicit data for the model and actually writes the algorithm's form.

Any effective modeling language translator must implement both of these phases, either as two independent subsystems or as overlapping parts of a single system. Indeed, the essential tasks of modeling language translation--parsing the language, interpreting the expressions, and converting to a lower level form--are currently performed by many common computer systems, particularly programming-language compilers. There can be little doubt that it is practical to implement a modeling language translator, given present-day knowledge of computers and computer systems.

Successful implementations offer another persuasive argument for the value and practicality of modeling language. Several modeling languages have been developed. These include structured modeling language (SML) (Geoffrion, 1987), generalized algorithm for mathematical systems (GAMS) (Brooke, Kendrick, Meeraus, and Raman, 1998), a mathematical programming language (AMPL) (Fourer, Gay, and Kernighan, 1990), linear, interactive and general optimizer (LINGO), and structured query language for mathematical programming (SQLMP) (Choobineh, 1990).

2.3 Decision Support System

The complexity of Decision Support System (DSS) design process is the result of the need to model not only the problem data and processes, but also the mathematical relationships, the integration of data and models, and the decision making style of the decision maker. (Raghunathan, 1996) One of the most widely accepted approaches (Turban, 1995) decomposes a DSS into dialog management, data management, and model management components.

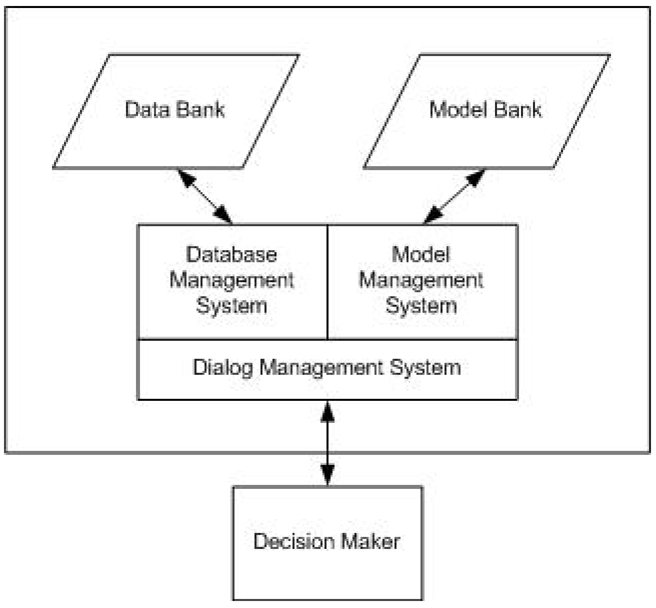


Figure 2-4 Typical architecture of DSS (Turban, 1995)

Dialog management comprises all user interfaces with the system such as a language for initiating system action (e.g., solve a model) and various means for representing information (e.g., graphics). Data management involves the creation, storage, manipulation, and retrieval of data. This may often require a database management system (DBMS) as a subunit of the DSS to perform this function. Model management involves equivalent functional capabilities for models, and a model management system (MMS) is the corresponding subunit of the DSS, which performs these functions.

A strong assumption of the DSS approach is that MMS should be designed to support the work of professional model builders throughout the model development life cycle, by providing its users with flexible and rapid access to models, data, and solution procedures. In Tsaih's resent study on DSS for credit rating (Tsaih, Liu, Liu, and Lien, 2004), the MMS design has taken some important steps. The system architecture is depicted as follows:

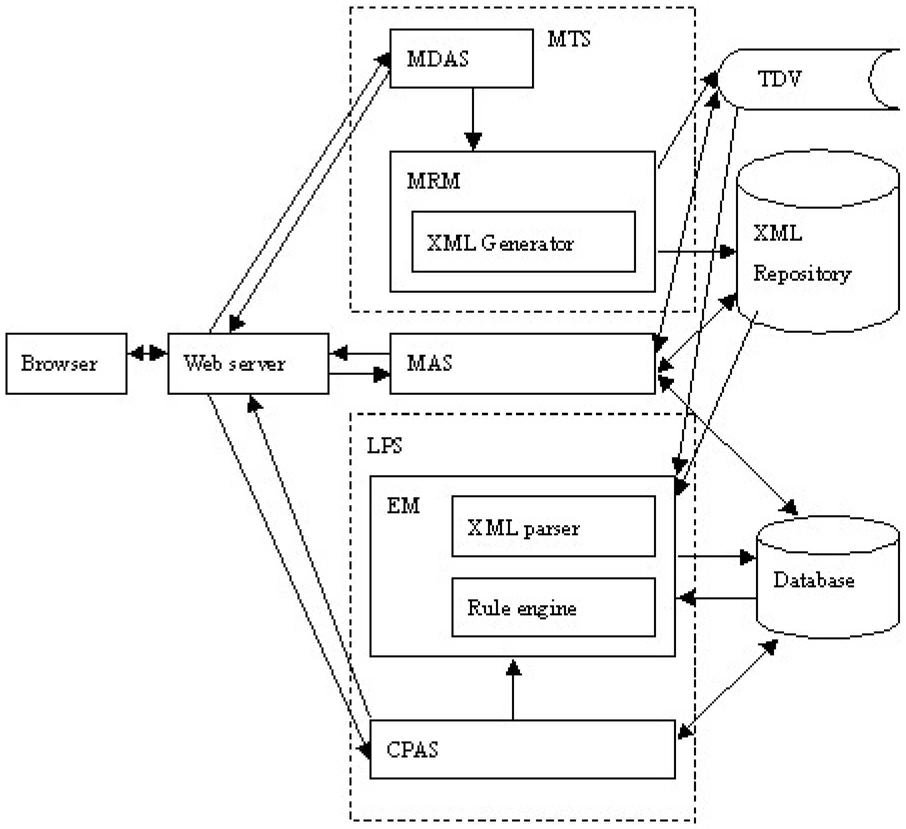


Figure 2-5 Architecture of DSS for Credit Rating (Tsaih, Liu, Liu, and Lien, 2004)

The MVC (Model-View-Controller) concept is adopted to implement web-based user interface. MTS (Model Transforming Subsystem) and LPS (Loan Process Subsystem) separate the concerns of building model from applying the model. MDAS (Model Defining Application Server), MRM (Model Recording Module) and TDV (Table of Defined Variables), together with the mechanisms like Calculation Priority, Rule Types, facilitate the complexity involved in building a well-defined mathematical model. The maintenance and upgrade of models are easy by merely re-defining the rule or adding new defined variables without managing any complex program.

With new breakthroughs in technology, it was no surprise researchers tried to apply their findings in ways that are useful to institutional and private investors. However, since the genesis of DFA was driven by the industry rather than academia, barely few academic studies have been made on the system architecture for DFA application. On the contrary, there are a number of companies in the market that offer software packages or components for DFA, usually in conjunction with related consulting services. Remetrica II by Benfield Group (http://www.benfieldgreig.com) provides modular environments that can be adapted relatively quickly to different company structures, and that are mainly used for addressing dedicated problems, usually the structuring of complex reinsurance programs or other deals. Another type of DFA software, such as Finesse 2000 by SS&C (http://www.ssctech.com), the general insurance version of Prophet by B&W Deloitte (http://www.bw-deloitte.com/home/), TAS P/C by Tillinghast (http://www.towers.com) or DFA by DFA Capital Management Inc (http://www.dfa.com), is large-scale systems that model a company in great detail and that are used for internal risk management and strategic planning purposes on a regular basis, usually in close connection with other business systems.