Chapter 8

Empirical Modelling Technology And Data Intensive Financial Applications

8.0 Overview

This chapter discusses the prospects for using Empirical Modelling Technology for data intensive financial applications. The context for this discussion is established in section 8.1, which considers a financial case study on tests for market efficiency. The data collection and financial analysis for this case study was conducted by Keng Yu Ho, while the technical programming work was undertaken by the author. Section 8.2 discusses the broad implication of EM technology on the finance research development cycle. Section 8.3 discusses the prospects towards applying EM technology in connection with data intensive financial applications.

The chapter concludes with the research agenda for using Empirical Modelling technology as a framework for computer-based support of data intensive financial applications.

8.1 The Case Study

8.1.1 Motivation and objective

The computer implementation of different techniques and methodologies used in financial research is rarely documented in the research itself or separately. This can be mainly attributed to the importance of result reporting and analysis in finance, which takes precedence over the computer implementation. Financial experts are not much concerned to reveal their adopted technology support as this will not give any value added to their results and inferences. AI technology is currently a pioneer technology in bridging the gap between financial analysis and computer support to this analysis. Implementation of algorithms, performance, and speed come first in the agenda for use of computers in finance. This gap in the literature on computer-based support in finance has serious drawbacks:

(1) The lack of adequate and complete reporting of computer-based support to financial analysis and research limits the scope of the computer-based support to the financial research activity and hides important aspects of the relationship between the finance research development cycle and its corresponding software system development.

(2) Limiting the computer-based support for financial research to the implementation of algorithms and econometric methodologies using fourth or third generation languages (e.g. spreadsheet and statistical packages, or programming using C or Fortran) diverts attention from providing semi-automated support for financial research and analysis. Providing such semi-automated support relies heavily on the coherent integration of the human and automated activity that makes it possible to infer meaning and conclude results in the course of financial analysis.

(3) The reliance on structured methodological approaches in financial analysis distracts attention from providing computer-based support for experiential knowledge construction that is guided by human insight and subjective understanding.

(4) The lack of concern for appropriate reporting of the computer-based support for the financial research obscures the links between the computing and finance domains and puts disproportionate emphasis on computer-based implementation of algorithms. This inhibits emerging technologies from offering wider support for financial research activity that is more intimately related to the computer-based activity and in particular to context dependent software system development.

This chapter considers a case study that is typical in data intensive financial analysis and computer-based implementation of various methodologies and algorithms. The case study serves three objectives that contribute to the theme of the thesis:

(1) To uncover quantitative and qualitative aspects of computer-based support for the financial research development cycle. Quantitative aspects refer to metrics on the amount of data processing and file manipulation as well as to issues related to the technical implementation of algorithms and methodologies. Qualitative aspects refer to the integration of the human and automated activity throughout the whole of the finance research development life cycle.

(2) To identify the prospects of EM technology in supporting the finance research development cycle and in particular in establishing a closer integration between the finance research development and software system development activity.

(3) To inform the development of the future generation of EM tools in respect of requirements for data intensive financial analysis.

8.1.2 About the case study

The research work described in this section was jointly conducted with Keng-Yu Ho[1]. All the financial analysis and data collection was undertaken by Keng-Yu Ho, whilst the computer implementation of econometric models, web development, and the electronic manipulation of relevant data sets, was carried out by the author. The outcome and progress of the research is framed within a web-based environment (cf. [Maa02-Web]) serving a documentary purpose.

As introduced in chapter 2, market efficiency refers to the extent to which security prices fully and correctly reflect all relevant information [Mal99]. This implies that it is impossible to make abnormal profits on the basis of that information set. All the empirical research on the theory of efficient markets has been concerned with whether prices fully reflect particular subsets of available information. Tests for weak form market efficiency were conducted with past price histories as the information subset of interest [Fam70].

Research on testing market efficiency is broad and encompasses testing the profitability of certain trading rules, testing asset pricing models, adopting event studies or examining stock market anomalies. The research considered in this thesis is confined to long-horizon event studies[2]. Event studies, introduced in Fama, Fisher, Jensen and Roll (1970), provide evidence on how corporate events affect stock prices [Ho00]. Two types of event studies can be

[1] Doctoral researcher, supervised by Dr. Abhay Abhyankar at Warwick Business School, in UK.
[2] Event studies are conducted to test the semi-strong form of market efficiency.

conducted: short-horizon (known as standard event studies) or long-horizon. Standard event studies assume that the response of prices to an event is short-lived, whereas long-horizon event studies focus on the impact of corporate events (such as rights issues, convertible debt issues, mergers, etc.) over an event window of up to 5 years. Further details on the historical overview and evolution of long-horizon event studies can be found in [Ho00].

The following paragraphs describe the financial research development cycle (FRDC), focusing on particular stages of the cycle (cf. Figure 8.1). This description highlights essential aspects of the FRDC such as:

- The size and amount of data processing (this aims at informing the requirements for data intensive computer-based applications)
- The programming paradigms used
- The methodologies adopted

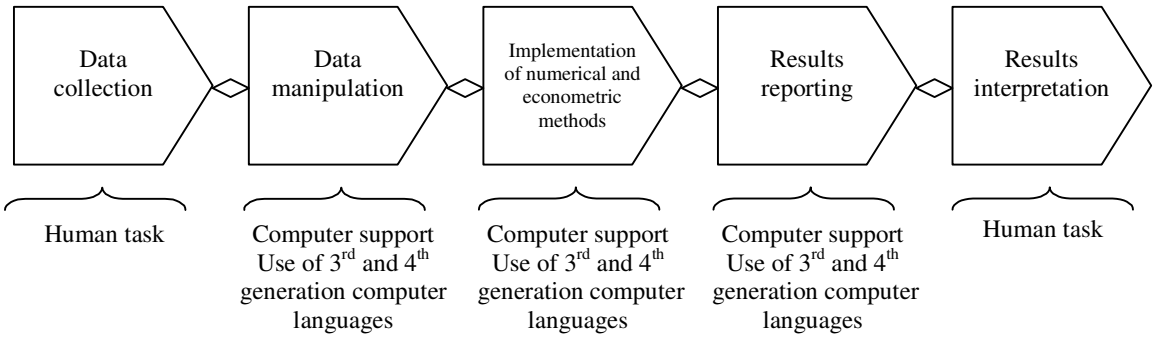

Figure 8.1 The Research Development Cycle

The data collection / manipulation stage:

The research work undertaken by Ho focuses on long-horizon event studies conducted using data from the UK market. The rights issues event (RI) is considered. UK data on rights issues was taken from Extel Takeovers, Offers and New Issues. *Returns* and *market capitalization* data is taken from the London Share Price Database (LSPD)[3] and *Book-to-market ratio* data is taken from Worldscope[4]. Details on the legal and procedural aspects related to rights issues event are provided in [Ho00, AH01].

In conducting the rights issues event study, collected data was stored in three types of file: **`event firm`** data files (including all firms having a rights issue event in a particular year and month over the period of study); **`book-to-market ratio`** data files; and

[3] http://www.lbs.ac.uk
[4] http://www.primark.com

`excluded firms` files (event firms excluded from the matching approach[5]). All files are saved as text and their data corresponds to a particular year over the period (89-99).

The amount of data stored in the files used is of the order of 100MB. The size of the returns file used (RETS1999 from LSPD) is 45MB. It consists of 6911 firms having data from 1955 till 1999. This file holds approximately 16KB of data for every firm.

The **`event firm`** files are relatively small, and include a little information about the event firm (its unique number in LSPD, and the date of the event). Each year, from 1955 till 1999, has its own **`event firm`** file. The maximum number of firms in an **`event firm`** file is around 95 firms.

`Book-To-Market ratio` (BTM) files are also relatively small. They include only BTM data and sedol numbers (unique identifier number of a firm in LSPD) for a maximum of 2000 firms. BTM files are also constructed annually.

`Excluded firms` files contain only the sedol number of a maximum of 400 excluded firms.

The manipulation of the returns file of the LSPD entails filtering selected fields, searching for specific firms, and extracting a specific range of returns.

The implementation of different techniques and methodologies:

Conducting long horizon event studies involves the estimation of long horizon abnormal performance of stocks following or prior to the event date. Different methodologies [AH01] can be used to assess this abnormal performance. These include:

i. *buy-and-hold abnormal return* (BHAR)[6], which computes the difference between the compounding return of the event firm and that of the benchmark over a period of time (3-5 years after the event);
ii. *cumulative abnormal return*[7] (CAR), that simply adds up the returns of the event firm in its benchmark and computes their difference as traditional short-horizon event studies.

[5] Firms that had a similar event in the previous 3 years before the event firm year are excluded from the matching approach.

[6] $BHAR_i = \prod_{t=1}^{T}(1+R_{i,t}) - \prod_{t=1}^{T}(1+R_{benchmark_t})$ where $R_{i,t}$ is the return for sample firms.

The mean buy-and-hold abnormal return is the weighted average of the individual *BHAR*s:

$$\overline{BHAR} = \sum_{i=1}^{N} w_i \cdot BHAR_i$$ where w_i is the weight (either equal-weight or value-weight).

iii. *calendar time abnormal return* which measures abnormal returns relative to the expected returns from a chosen asset pricing model (the Fama-French three factors model[8]) .

Conventional t-statistic[9] is used, and p-values, based on skewness-adjusted t-statistics are recorded.

In assessing the abnormal performance, using CAR and BHAR, three types of benchmark firms are used:

(1) The matched size / industry benchmark: the benchmark is taken as a matching firm in the same industry and having a closest but higher market capitalization than the event firm in the corresponding event year.

(2) The matched book-to-market ratio / industry benchmark: the benchmark is taken as a matching firm in the same industry and having a closest book-to-market ratio to the event firm in the corresponding event year.

(3) The matched portfolio benchmark: the event firm is matched with one of 25 size / book-to-market portfolios. These portfolios are obtained by ranking all firms from LSPD for which market capitalization data is available for the end of the previous calendar year and for which book-to-market ratio data is available at the end of the previous fiscal year. Firms are allocated into a 5x5 grid and the 25 value-weighted

---

[7] $CAR_{i\tau} = \sum_{t=s}^{s+\tau} [R_{it} - R_{benchmark}]$ where R_{it} is the simple monthly return for sample firm *i*, $R_{benchmark}$ is the simple monthly return on the benchmark that corresponds to the firm i.

[8] For each calendar month, form the portfolio that contains all the event firm returns within the 3-5 years period of study, the FF 3-factor model below is used to estimate the abnormal return:

$$R_{pt} - R_{ft} = \alpha_i + \beta_i (R_{mt} - R_{ft}) + s_i SMB_t + h_i HML_t + \xi_{it}$$

R_{pt} is the monthly return on equal-weighted or value-weighted claendar time portfolio
R_{ft} is the monthly return on one month treasury bills
R_{mt} is the return on the return of the value-weighted portfolio of all firms in the market
SMB_t is the difference in the returns of the value-weighteed portfolios of small stocks and big stocks
HML_t is the difference in the returns of the value-weighted portfolios of high book-to-market stocks and low book-to-market stocks
α_i, β_i, s_i, h_i, and ξ_i are parameters of the regression.

[9] $t = \dfrac{\overline{BHAR}}{\sigma(BHAR)/\sqrt{n}}$

where $\overline{BHAR}$ is the mean and $\sigma(BHAR)$ is the cross-sectional standard deviation of abnormal returns for the sample of *n* firms.

portfolio returns are calculated for each month. The event firm is then matched to a portfolio with similar size/book-to-market characteristics.

The following paragraphs elaborate on the programming paradigms used in assessing the abnormal performance of event firms computed using different methodologies and choice of benchmarks. General features of the developed programs are presented and some software metrics are provided:

Third and fourth generation languages (C and Excel) are used in the computer implementation of the different methodologies and benchmarks used in assessing the abnormal performance of event firms. The programming paradigm used in the case study is the imperative[10] one, and the C programming language is used. The C programming language is chosen for speed, ease of use, and compatibility with available financial libraries. The spreadsheet application Excel was used for testing the correctness of generated outputs and for conducting statistical analysis on output results.

All programming languages have their good and bad points, and consequently the choice of a particular language for the implementation of econometric and numerical methodologies has advantages and disadvantages. Modern programming languages have many similarities in types, declarations, expressions, statements and data structures. By using relatively few language constructs and avoiding implementation decisions based on peculiarities of C, we can develop programs that can be easily translatable to other languages. The advantage of C is that it is widely used and has all the basic features needed in various implementations. C has a rigorous high level syntax that allows easy identification of the main features of the program [Sed90].

The imperative programming paradigm was adequate to the program development done so far. The designer was very careful in specifying the program requirements for implementing the econometric and numerical methodologies adopted in terms of input – process – output. This makes it appropriate to use an imperative programming language. Data management was a great concern. The UK dataset was provided in textual format – this made the file processing approach the most straightforward to use.

The choice of a proper data structure is a major decision in the implementation stage. Declaring and using abstract user defined data types was constrained by memory capacity.

[10] An imperative language uses a sequence of commands to carry out the desired operations.

At an initial stage of program development, object oriented programming (in JAVA or C++) and design would not have had a great impact on the analysis and programming stages. Declaring a user defined abstract data type for a firm was quite sufficient.

Arrays (one dimensional and multi-dimensional arrays) and user-defined types (implemented using the data structure **`struct`** in C) were the main data structures used. Examples of user defined data types used data on firms read from different files (LSPD RETS, constructed Book to Market ratio (BTM), and event firms files) are provided in the following table.

Table 8.1 User defined data types used in methods implementation

User defined data type for LSPD RETS firm	***User Defined data type for Event firm***	***User Defined type for BTM firm***

`typedef struct`
`{int ncomp;`
`int sedol;`
`char rname[33];`
`int dti;`
`int freq;`
`int start;`
`int end;`
`int nobs;`
`int samp;`
`int indy;`
`int mval;`
`int ntrade;`
`int ftpr;`
`int dyld;`
`int per;`
`int dmark;`
`int netass;`
`int spread;`
`float zret[540];`
`int dates[540];`
`int mcap[45];`
`} lse_companies;`
	`typedef struct company`
	`{int starting_index;`
	`int event_year;`
	`int event_month;`
	`int sedolnum;`
	`} company ;`
		`typedef struct`
		`{`
		`int sedol_number;`
		`char btm_name[33];`
		`int btm_mcap;`
		`int btm_startyear;`
		`int btm_endyear;`
		`float btm;`
		`int btm_indy;`
		`char marker[2];`
		`} btmdata;`

The *lse-companies* user defined data type represent a company in the LSPD returns file RETS. It has 18 fields, and 540 returns, dates, and market capitalization taken from year 1955 until 1999. The user defined data type *company* represents an event firm. The event year, month, and the sedol[11] number identify the event firm. The user defined data type *btmdata* represents data for a firm in the book-to-market ratio file.

*The standard C libraries (*stdio.h, stdlib.h, ctype.h, string.h, float.h) were used and some user defined mathematical and selection functions were implemented.

The file data management paradigm was adopted and large LSPD text files were read and saved as *binary files* for ease of access and search.

Basic building blocks of programs implementing the different methodologies to compute the abnormal performance using a particular choice of benchmark are:

- *matching:* used when taking the benchmark as book-to-market / industry match or size / industry match
- *sorting* : used when taking a 25 size / book-to-market portfolios benchmark and when adopting an asset pricing model to assess the abnormal performance of event firms
- *bootstrapping :* used to compute the skewness-adjusted t-statistics
- *filtering* : used in the matching and for excluding certain firms

The programs developed are:

(1) Programs to find different benchmarks for an event firm:
   - a matching firm with nearest book-to-market ratio and same industry group
   - a matching firm with nearest book-to-market ratio and same industry group
   - a portfolio with same similar size and book to market characteristics

(2) Program to find compute adjusted t-statistics for computed abnormal returns:

(3) Program to find the portfolios used in the asset pricing model (Fama French 3 factors) to compute the abnormal performance of the event firm

A brief description of some of these programs is given below, specifying each program in terms of input-process-output. In the following description MCAP refers to market capitalization, BTM to book to market ratio, Sedol to a unique 8 digit number identifying a UK firm in LSPD, and RETS to LSPD returns file.

Size / industry matching: This program takes as input event year data (event date, firm sedol), excluded firms data (sedol), returns data (LSPD RETS), and produces an output file showing for every event firm up to 10 matching firms (5 up, 5 down) in the same industry group. The matching between firms is based on market capitalization information.

Book to Market / industry matching: This program takes as input event year data (event date, firm sedol), excluded firms data (sedol), returns data (LSPD RETS), BTM

[11] Every firm in LSPD has a unique sedol number that identifies it

data (BTM, sedol), and produces an output file showing for every event firm up to 5 matching firms in the same industry group. The matching between firms is based on BTM information.

25 size / book-to-market portfolios: This program takes as input a number of years (usually 4 years), starting and ending years, event firm data (sedol, event date). Then for each year between the starting and ending years, it prompts the user to enter BTM data (sedol, BTM) and the file of excluded firms for the given year. The program produces 34 output files. 25 output files are produced for 25 portfolios formed by arranging firms into quintiles according to their BTM (book to market ratio) and market capitalization value. 4 files for BTM firms for year i sorted by BTM. 4 files for BTM firms for year is sorted by MCAP. One file for all event firms with their 3 years returns (starting from event date backward or forward), the portfolio in which the event firm is allocated in the event year, and the return of all portfolios of the same quintile of the event year for 3 years.

Fama-French three factors model:

(1) Portfolio formation: This program takes as input excluded firms data (sedol), returns data (LSPD RETS), and BTM data (BTM, sedol), and produces 10 output files. Two output files are for sorted firms (firms in BTM file except excluded one) by BTM and market capitalization. Six output files (portfolios) are grouping firms allocated as big/small MCAP, high/low/medium BTM One output file for all firms in BTM files (ignoring excluded firms) with all necessary information about the firm. One output file for portfolios returns calculations.

(2) Market returns: this program takes as input a given year and excluded firms in this year (sedol num), and produces an output file of all firms in LSPD (excluding firms from the excluded input file) with their monthly returns for the input year. The output file also includes calculation of equally weighted average returns of all firms and average returns of all firms weighted by market capitalization.

Bootstrapping: this program takes as input a vector (array) of float numbers and produces n re-samples of b sub-samples of the vector of input data. In the program, n is taken

as 1000 and b as 500. Some t-statistics are computed for the re-samples.

Informal statistics on the scale of programs written are summarized in the table below. They refer to the four programs implemented: the BTM and MCAP matching, the 25 portfolio, and the Fama and French portfolios (FF3) programs.

	MCAP MATCHING	***BTM MATCHING***	***25 size / book-to-market portfolios***	***FF3***
Number/ Size of input files	3 input files events firms (2-3kb), excluded firms (2-3kb), RETS (45Mb)	4 input files: event firms (2-3kb), excluded firms (2-3kb), RETS (45Mb), BTM file (30kb)	9 input files: event firms (2-3kb), BTM file and excluded firms files are input for 4	3 input files: BTM file (30kb), RETS (45Mb), excluded firms (2-3kb)
Number/ Size of output files	1 output file (300kb)	1 output file (250kb)	34 output files: * 25 portfolios * 8 files (BTM, MCAP sorting) * event firms and their similar portfolios returns in 4 yrs	10 files * 6 portfolios * sort by BTM * sort by MCAP * return calc
Max no. of times a single file is opened	12 times (RETS)	7 times (RETS)	3 times (RETS)	8 times (RETS)
Running time	~20min	~20 min	~ 1 hour	~15 min
Size of program ***Lines/bytes***	42kb 1585 lines	25Kb 993 lines	34 kb 1415 lines	24kb 1073 lines
Basic blocks	Find min & Sequential search algorithms	Find min & Sequential search algorithms	Bubble sort and sequential search	Sequential search and bubble sort
Rules / Conditions/ data structure	2 user defined types are declared: (1) for a Firm from RETS file - LSE firm, (2) for an event firm and its associated information.	3 user defined data types are declared: (1) RETS firm, (2) event firm, (3) firm from BTM file	four-dimensional array (year, portfolio i, j, nth return)	3 user defined data types are declared: (1) RETS firm, (2) event firm, (3) firm from BTM file
Frequency of usage per study	Used per event year/event study (9 times for 9 yrs) 9*3 =27 for 3 types of event studies	Used per event year/event study (9 times for 9 yrs) 9*3 =27 for 3 types of event studies	Used per event year/event study (9 times for 9yrs 9*3 =27 for 3 types of event studies	Used per event year/event study 9 times for 9yrs 9*3 =27 for 3 types of event studies

Table 8.2 Preliminary statistics on the scale of programs written

These statistics reveal intensive file processing and data manipulation operations. Memory capacity shaped the structure of the programs. The amount of data that can be held in memory as n dimensional arrays of user defined data type is limited. This constrains the way user-defined data types are declared and increases the need for file processing operations. For example, the LSPD returns file was opened 12 times in the matching program to search for up to 5 matching firms (MCAP / Industry matching or BTM / industry matching) for a particular event firm.

The above statistics inform the development of computer-based tools for data intensive financial analysis. Such tools should take into consideration the amount of data processed, speed of processing, data structure used, and data manipulation paradigm.

Results analysis:

The research conducted by Ho and Abhyankar has so far examined the abnormal performance of a large sample of UK firms following rights issues and placings over the period 1989-1998. The research findings revealed in [AH01] show that long-horizon abnormal performance in the UK financial market is highly sensitive to the methodology used to compute the abnormal performance depending on the choice of benchmark. This confirms the view that the apparent anomaly of under-performance[12] may be due to the methodology adopted, and the long-horizon abnormal return may disappear with a reasonable change in techniques. The study concludes with the need to adjust correctly for risk in measuring long-horizon security returns.

The financial research development cycle (FRDC) has been framed within a web-based environment serving a documentary purpose. The PPP (Pre-Publish-Push) technique has been used to develop this web environment. The web environment contains collected UK data, a description of the techniques used, the programs developed, and access to related publications (cf. [MaadWeb02]).

[12] There is mounting evidence, from USA and Japanese markets, of long-horizon security price under-performance following corporate events like initial public offering (IPO) and seasoned equity offering (SEO).

8.2 EM Technology Support for the FRDC

Chapter 4 considered the broad implications of EM for computational activity in an application context, and for software system development in particular. This is manifested in a radical shift that supports openness, situatedness, and context dependent software system development where modelling (as opposed to programming) is central and user-developer-designer situated collaboration is essential. This shift potentially enables a closer integration of the software system development activity and various activities undertaken in real world domains.

EM technology motivates a radical shift in software system development. This gives EM technology a greater potential to establish a closer integration between the Finance Research Development Cycle FRDC and the software system development activity. Such integration can potentially enrich the FRDC with greater computer-based support and a better understanding of the role of the computational activity in financial research. The following sections discuss the relevance of the EM paradigm shifts in SSD to the computer-based support of the FRDC. Three issues are considered:

i. collaboration (user-developer-designer / social network of finance researcher)
ii. the amethodological approach to software system development in EM
iii. the shift from programming to pervasive agent-oriented explanatory modelling

i. Collaboration

Collaboration in a distributed modelling environment is an important feature in providing computer-based support for the FRDC. In the context of the Ho case study, this is illustrated by various observations about the user-developer-designer interpersonal interaction that show that software system development is a social activity that needs computer-based support to meet its objectives.

In the Ho case study, the software system development process involves the designer (Keng-Yu Ho), and the developer (the author). The designer is also the user.

The major challenge faced in the software development process is the gap between the designer's knowledge of the meaning of the financial data set, and the abstract representation adopted by the developer to represent this data set in the program. As an example to illustrate this gap, consider the 12 monthly returns in a given year for a firm. In the designer's mind, specific year and specific month under consideration are important. For the developer returns

are represented as an array of float, and every return is accessed by its index. It is not important to the developer whether this index refers to the month of January or February or any other month within a particular year so long as the returns are abstracted as a fixed size array and accessed through an index. The gap between the designer's and the developer's views is widened as new requirements arise that attach greater importance to the real world meaning of abstract variables in the program. Close interaction between the developer and the designer reduces this gap to a certain extent, and sometimes this interaction takes the form of direct intervention from the designer in the software development to dictate a relationship between variables.

Turning a programming activity into a shared modelling activity and adopting a modelling paradigm that establishes a stronger virtual correspondence between real world entities and their virtual representation helps in reducing this gap and its serious impact on the financial research development cycle.

The following figure illustrates the conceptual gap between the designer (financial expert) and developer (programmer) in providing computer-based support for the financial research development cycle. This gap can be bridged by efficient communication aimed at establishing a greater correspondence between the state of the financial research and its corresponding computer-based model. This state has four components (cf. [BRWW01]): visible external, hidden / internal, mental, and situational. An example of a visible state is the value of financial indicators presented in tabulated output formats. An example of an internal state is the abstract computation that determines the visible state of the financial indicator. An example of a mental state is the meaning and financial relevance of the financial indicator when used in the analysis. An example of a situational state is the context of a particular adopted methodology.

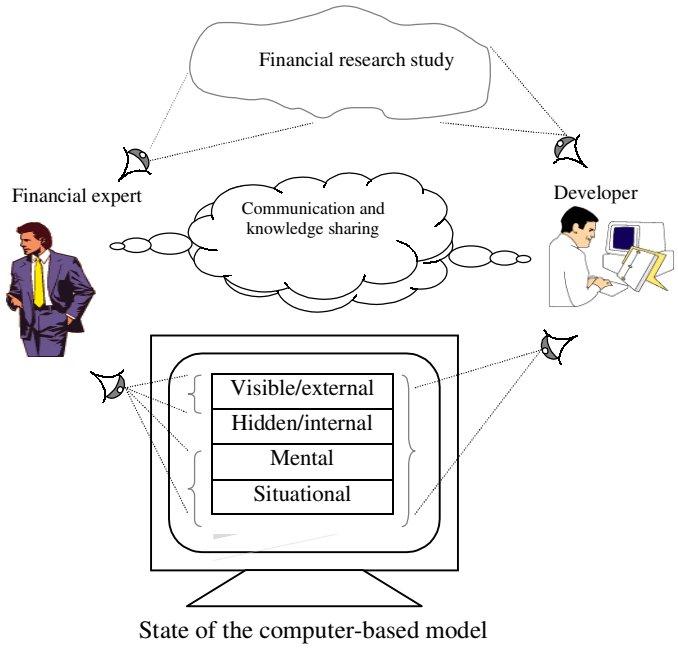

Figure 8.2 Sharing knowledge about the state of the financial research and its computer-based support

ii. amethodological software system development

An amethodological approach to software system development promises a closer integration between the software system development activity and the finance research development activity. In the context of the Ho case study, this emerges from a closer consideration of the FRDC and its associated software system development. A situated account of the FRDC would be better supported by a non rigid software system development approach.

A typical life cycle of a software system may be divided into seven stages: problem definition, feasibility study, analysis, system design, detailed design, implementation, and maintenance. In relating the considered finance research development cycle (FRDC) to a software system development cycle (SSDC), there is no direct one to one mapping between the different stages in the two cycles (cf. Figure 8.3). The financial data collected and the structure, meaning, and uses of this data play a major role in the problem definition, analysis, and design. The appropriateness of a particular econometric approach can only be tested after doing a complete analysis of the results. This is a major concern, as much time and resources

can be wasted on implementing econometric approaches that do not support the expected conclusions effectively. The problem definition is dictated by the designer. The feasibility study considers different algorithms for implementing the suggested econometric approaches. The developer's role is important in the implementation stage, but the designer has more responsibility and influence on the direction and evaluation of the work. The software system is used throughout the whole life cycle of the FRDC and terminates at its end. Unlike a SSDC, where the user enters at the end of the testing stage to use the finished software product, the user in the FRDC is the designer and uses the evolving software product throughout the whole development cycle. Once the final stage in the FRDC is reached, the software product is no longer useful[13]; it serves the user incrementally during the development cycle. The nature of specialized finance research is such that it is difficult to make use of the special purpose software product developed throughout the whole FRDC in subsequent specialized research without much amendment. This raises the issue of re-using general purpose utilities to support future similar research.

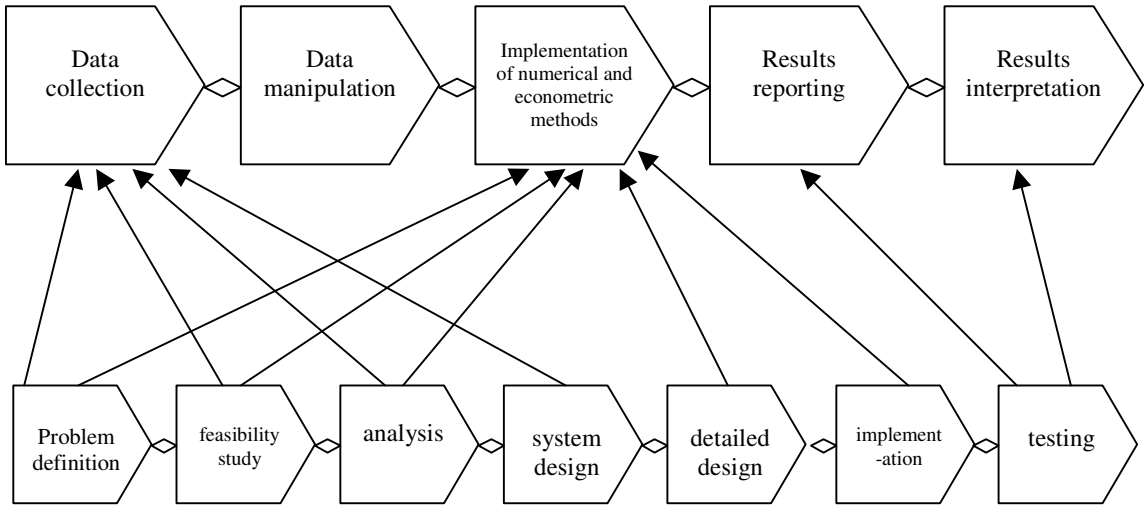

Figure 8.3 FRDC Vs. SSDC

[13] Except as reference to future similar research work, that might adopt different methodologies with similar spirit.

One of the major problems encountered in the FRDC is the need to find reliable techniques for testing. Failure to get the expected results might be attributed to the inadequacy of the adopted methodology, an incorrect implementation of the methodology, or inappropriate processing of the resulting output after the implementation stage.

The above account of the FRDC reflects the author's view, but the designer might well have a different FRDC model in his mind that is not communicated to the developer. This serves to emphasise how the specialized nature of the research makes it difficult to replace the role of the human designer and developer by an intelligent agent. It also serves to highlight how ambitious are the aspirations of logicist AI technology. It is hard to conceive the encoding of specialized financial research knowledge that could inform an intelligent agent capable of collecting, organizing, exploring different methodologies, selecting an optimum methodology, implementing it, analysing the results and inferring conclusions.

iii. The shift from programming to pervasive agent-oriented explanatory modelling

A situated account of the FRDC and a closer integration between the finance research development and software system development activities motivate a paradigm shift from programming to pervasive modelling in providing computer-based support for the FRDC. Whereas programming is limited to the computer-based implementation of methodologies and algorithms using third and fourth generation languages, pervasive modelling contributes to a more holistic view of the FRDC that integrates the human and the automated activity.

Current approaches to computer-based support for the FRDC (provided by third and fourth generation languages, Object oriented, Virtual Reality, and Artificial Intelligence technologies) is limited to the programming / implementation of algorithms and methodologies. The following paragraphs discuss several of these approaches with reference to the Ho case study. The aim of this discussion is to emphasise the importance of providing computer-based support for modelling the FRDC that takes into account the wider scope involving the support of:

- end-user programming
- greater human computer integration
- semi-automated activities
- experiential knowledge construction
- the generalisation of the spreadsheet concept beyond data in tabulated cells
- the user - developer - designer collaboration

The OO approach to support the FRDC

The concepts of modularity[14] and abstraction[15] [Win93] that motivate the object oriented programming approach were little considered in the Ho case study for two reasons. In the first place, many of the operations on the user defined abstract data types could not be preconceived and it would have been difficult to identify all the methods for a particular object at the outset. Second, the programs written are tailored to their highly specialized research purposes to such an extent that addressing modularity and re-use would have been an unnecessary distraction that delayed the delivery of essential results. A good example of the application of object oriented programming in the area of finance is in automating bank operations [Win93]. The operations on a bank account are limited and well defined. This makes it appropriate to declare an abstract object data type **`bank account`**, and to specify the methods of this object as a constructor, a withdraw, a deposit, and a balance method. In the Ho case study, different econometric and numerical algorithms are implemented as the need arises and as particular algorithms fail to provide the expected results. This fact makes it difficult to have an *a priori* description of objects.

The author did not optimize the algorithms used in the implementation with respect to speed and efficient use of memory as the designer's major concern was to get the correct output for a given input and process. It might be that object orientation brings more modularity and manageability to the programming code, but this issue was outside the scope of the author's investigations.

Exploration vs structured methodologies in financial analysis

Adopting structured numerical and econometric techniques to analyse large financial data sets is a popular approach in academic financial research. Uncovering hidden knowledge in huge data sets is a challenging task, and structured analysis has a primary role in gaining knowledge from large data sets. For a single case study, different numerical and econometric techniques can be applied. Each of these techniques reveals a hidden knowledge aspect in the data set. A major problem that might be encountered in financial research is the failure of a particular econometric or numerical approach to deliver the expected result. Some measures could be taken to prevent the commitment to an inadequate approach or to an incorrect implementation. A major concern is that the time spent to explore the adequacy of a particular

[14] Modularity ensures that all elements of the system have well defined boundaries separating each other and clearly defined interfaces through which the elements interact - the modules only interacts through interfaces.

[15] Abstraction is the act of ignoring the detail of a component of a program and reasoning only about the interface. Two types of abstraction exist: data abstraction and functional abstraction.

econometric or numerical approach might be the same as the time for implementing and testing it. This motivates an exploratory approach in analysing large financial data sets.

End-user support throughout the FRDC

In considering the use of end-user programming[16] in the implementation stage of the RDC two limitations are confronted. The first limitation is the large amount of file processing needed to implement the selected methodologies. The second limitation is the need to perform multiple repetitive operations that are not available as built in functions and procedures in end-user programming tools such as Gauss and Matlab. A solution emerges with the evolution and progress of the RDC. This solution would be to develop generic utilities for re-use by other similar research. In the context of the Ho case study these generic utilities might include matching, portfolio creation, market return calculations, and bootstrapping. The possibility of implementing these generic utilities as encapsulated objects is a good solution for re-use and modularity. The use of visual queries techniques to manipulate large data sets might be a complicated approach in specialized financial research such as the one considered, due to the complexity and diversity of special purposes data manipulation techniques.

AI support for the FRDC

In considering the application of an AI approach to conduct the Ho case study, two questions are addressed:

i. Can AI techniques (neural networks, fuzzy logic, visual exploration using self organized maps) replace / complement the econometric / numerical approaches adopted?

ii. Is it possible to encode the knowledge constructed throughout the whole FRDC, and use this knowledge to create an intelligent agent capable of replicating the same analysis in a different context (different country)?

Though the answers to these questions that follow may appear superficial, they raise deep conceptual issues relating to the success of AI technology in actively supporting advanced and specialized financial research.

i. The literature overview conducted by Ho (2000) reveals that different econometric approaches lead to different inferences and conclusions. This calls the viability and reliability of econometric approaches into question and leads the author to conclude that replacing an econometric technique by an AI technique is not a solution. A complementary role is rather advocated to get satisfactory results.

[16] The terms ' end-user programming' and ' programming using fourth generation languages' refer to a programming style that gives the end-user greater computational assistance whilst requiring minimal knowledge of the complex aspects of programming.

ii. It is hard to imagine encoding the designer and developer expertise used throughout the whole RDC in a knowledge base that can be used in a logicist AI approach to automatically re-define a similar problem, explore a solution set and select one optimum solution. The first challenge is the difficulty in encoding the expertise expressed in a group activity (designer, developer collaboration) in defining the problem, selecting an optimum methodology, implementing this methodology and testing the results. Collaborative social activity is rich in interaction and knowledge sharing. AI techniques would be better targeted at supporting the collaborative activity rather than attempting to encode and automate it. The second challenge concerns the context and situation of the financial research. There can be no guarantee that distributed intelligent agents capable of taking autonomous actions to conduct a UK based financial study would be effective in carrying out a similar study in a different market.

VR support for the FRDC

Traditional application of VR technology in finance attempts at merging the mechanism of a trading process with the analysis of the results of the trading process in a synthesised 3D visual environment (cf. http://www.nyse.com/floor/floor.html). This highlights the importance of the human visual system and its role in supporting knowledge construction about an environment.

Where case studies such as the Ho case study are concerned, 3D visualization in a virtual scene may be helpful in navigating large data sets in an exploratory way. Eyeballing techniques potentially have a useful role in conducting or supporting advanced financial research analysis. In the Ho case study, the nature of the data set analysed (event date, monthly returns over a particular window, particular indicators – market capitalization and book to market ratio) and the required operation on the data set (matching, sorting, portfolio allocation, bootstrapping) indicate the need and importance of relying on complex methodologies to conduct the research. This makes eyeballing techniques, that are enabled with VR technology, useful at an end stage. Their role would be in enhancing the visualization of results and in supporting their interpretation.

Another role for VR technology would be to support the description of the event occurrence mechanism. Such a description is found in [AH01], where the sequence of activities followed by UK firms seeking capital through rights issues (a corporate event) is described in the course of reviewing particular aspects of the institutional framework in the UK market.

The above discussion of various approaches to supporting the FRDC reveals that the computer-based support offered by current technologies is confined to implementation and

reporting stages (cf. Figure 8.1). In this kind of support, the computer plays the role of an advanced calculator. The cognitive activity of the financial expert involved in stages 1, 2, and 5 (data collection, manipulation, and results interpretation) of the FRDC enjoys no automated support apart from routine data processing. This is despite the fact that this cognitive activity, if appropriately conducted and supported, would have great impact on the success and usefulness of the computer-based support for stage 3 and 4. This emphasises the importance of modelling the FRDC in a more holistic manner.

With the adoption of an EM approach to SSD in this context, the modeller can bring understanding of human agency and potential computer agency to bear on the construction of a computer-based artefact. Human agency relates to the cognitive activity associated with the choice of strategies for data collection and manipulation, the methodologies for financial analysis to be implemented, the style of result reporting that impact on the interpretation of results. EM can potentially enable the modeller to embody these activities in an artefact without invoking circumscription. This allows the SSD to proceed without premature commitment to a particular data management strategy, analysis methodology and reporting style. This feature, together with the agent-oriented qualities of the artefact and its explanatory character, potentially offer flexibility for experiment that can transform the scope for the modeller both to frame, and to adapt to, new requirements.

By way of a motivating illustration, in the context of the Ho case study, the organization of data files (**`event firm`**, **`book-to-market ratio`**, and **`excluded firms`** data files) and the choice of benchmark (matched size / industry, matched book-to-market ratio / industry, and portfolio benchmarks) used in assessing the abnormal performance of a firm, involve strategic decisions to be taken by the financial expert that have radical implication for the software requirements of the FRDC and are subject to revision.

8.3 The Implications of data intensive financial applications for EM Technology

Adapting EM technology for data intensive financial applications raises issues related to ***referencing***, ***processing***, and ***interpreting*** large volumes of data. Research in progress undertaken by EM technology to tackle these issues is overviewed below.

8.3.1 Referencing large volumes of data

Three approaches have been used to address the limitations of definitive programming within the Empirical Modelling Framework (EMF) in referencing large volumes of data and porting EM concepts to an object and system level:

The first is by applying the concept of dependency maintenance to explicit relationships between abstract objects data types. This is facilitated with the use of JaM[17] the Java Maintainer application programmer's interface (API). As documented in [Cart99], JaM provides a generic dependency maintenance system for use in object-oriented programming. It also provides facilities for organising the information in a dependency structure into virtual directories and manages user and group permissions to access and modify the data and dependencies. JaM' s dependency maintenance can be considered as a generic spreadsheet in which the *cells* of the spreadsheet are instances of any class chosen by a developer that implement an interface of the API, not just conventional types of dates, strings and numbers. The dependency between JaM *cells* is established by the method of a class implemented by a developer that is equivalent to a spreadsheet *function*, such as "SUM". Change the value of any of the values in the *cells* and the values of all dependent *cells* are automatically updated. In contrast to a spreadsheet, items of data do not have to be organised in a grid and are identified instead by a name and directory path. Many *cells* can be changed simultaneously and JaM is fully multi-user and threaded, taking advantage of multiple CPUs if they are available.

The second approach used in addressing the limitation of definitive programming in referencing large volumes of data is by using virtual agency [Sun99]. Conceptually, virtual

[17] Under development by Richard Cartwright at BBC Research and Development, UK. The Java Maintainer Machine API version 1 (JaM) was developed as a demonstration of a generic object-oriented dependency maintenance system illustrating a new parallel method for definition update, the DM model. Developed in 1996 using Java 1.0, JaM is documented as part of the Phd thesis [Car99].

agency provides a way of attaching a definitive script to a particular observer, typically so as to represent the personal perceptions of that observer. Ideally, the association between a script and its observer should be defined and manipulated as a form of dependency, however, practically virtual agency is managed in dtkeden in a procedural fashion. With the virtual agent notion, a definitive script can be used as a pattern to generate different definitive scripts associated with different contexts. This allows reusability of a definitive script in dtkeden. Another solution for re-use introduced in [Sun99] is the introduction of Generic Observables (GO). Unlike those observables that correspond to real world objects in the modeller's external environment, GO are created to correspond to the modeller's experience, which is inside the modeller's mind and emerges from repeated description of certain observables with the same characteristics.

The third approach to addressing the limitations of definitive programming in referencing large volumes of data is distributed modelling. Distributed modelling [Sun99, Car99] helps in spreading the data processing load on different workstations. This increases the capacity of the distributed model to handle larger data sets describing a generic observable.

8.3.2 Processing large volumes of data

The main tool used in EM research so far is the **EDEN** interpreter and its distributed variant **dtkeden**. The **EDEN** interpreter is primarily a dependency maintainer for formula definitions and action specifications [Tru96]. The main weakness in all interpreters, including **EDEN**, is that program instructions and data when parsed line by line, translated and executed, must be held in main memory all the while. In general, interpreters are effective and efficient for applications that requires intensive processing on a small set of data. They are not so effective or efficient for applications that perform a small amount of processing on large volumes of data. Such type of applications are commonly referred to as *data intensive* applications (as opposed to *processing intensive* applications). **EDEN** inherently keeps all of the large volume of data within memory, while performing little or no processing on them. This tends to use up system resources and reduces the execution speed of programs. This creates a serious problem when considering data intensive applications. Attempts at providing a solution to the problem of increasing the efficiency of data intensive **EDEN** applications and at providing a means to manipulate large volumes of data effectively while respecting the EM philosophy of definitions and evaluations based on dependencies, was initiated by Truong (1996). His solution was to design and implement a Relational Language Interpreter **EDDI**

and to use it with ORACLE database. This interpreter forms a front end to **EDEN**, where relational instructions are translated to database manipulations. **EDEN** script can include action specifications that can simulate the storage, retrieval and manipulation of data. The objective is to develop data intensive applications in **EDEN** while maintaining the values of an application variable in a database. When the necessary values are actually needed, then they can be retrieved from the database into memory variables for manipulation. Hence, large volumes of data do not need to be kept in memory, freeing up memory resources. The relational language interpreter is designed with the relational data base model in mind. The model deals with tables of data and operations on data tables.

The extent to which JaM and dtkeden offer enhanced processing capacity has already been remarked. A more radical development, aimed at improving processing efficiency by handling dependency maintenance closer to the hardware level can be found in the definitive assembler machine, as introduced by Cartwright [Car99]. There will always be processing overheads attached to maintaining dependencies that can be eliminated once the functionality of models has been circumscribed for specific applications. This has been the theme of projects directed at extracting conventional programs from ISMs, as discussed in [ABC98]. Translation activity of this nature will always be desirable – if not essential – for the effective application of EM in data intensive and performance critical applications.

8.3.3 Interpreting large volumes of data

Since many data sets fail to satisfy the underlying assumptions of most econometric approaches, unconstrained visual exploration is gaining more importance in the financial industry. EM technology supports the interactive visual exploration of data sets [Roe00][18]. The aim of the research undertaken by Roe (2000) is to overcome the limitations of conventional enhanced database approaches to exploring large data sets. These include: the need to preconceive the field(s) to view before writing a database query; the problem of uncovering hidden relationships between fields when a large volume of data is returned by a query; the difficulty in spotting complex relationships that might exist across multiple fields in numeric data. Roe's proposed visualization is extrinsic (the properties of the data elements to be displayed are mapped to a physical counterpart – shape, color or texture). The visual exploration environment developed is regarded as an interface between the human and a data set. The relationships that may exist in the real world between the data attributes are virtually

[18] The work was inspired by the Attribute Explorer originally developed by IBM, UK, due to an idea of Bob Spence, Imperial College, London.

mediated to the computer environment as visualized dependencies. Human insight guides the exploration of the data set in many personal subjective ways. This helps, at an initial stage, in apprehending the data and its meaning. Compared to statistical regression, visual exploration helps to provide a deeper understanding of complex non-linear relationship between data. The challenge of interactive visual exploration is the difficulty of formulating in mathematical relationships our experiential knowledge of the data as apprehended. This motivates future research on transforming visual and mental understanding into formulated symbolic knowledge.

8.4 Summary and Outlook

This chapter considered a case study related to data intensive financial analysis research. The analysis of the case study was enriched by EM principles that suggested a situated account of the financial research development cycle. At the same time, EM technology can benefit knowledge about the requirements of computer-based support for data intensive financial analysis from the case study.

The future research agenda for EM technology to address the needs of computer-based support for data intensive financial analysis comprises technical and conceptual objectives including:

(a) Further research to integrate EM technology with database technology and VR technology. This is considered to be a long term objective.

(b) The application of the principles of observation, agency and dependency for exploring, in an amethodological way, hidden patterns of relationships in large financial data sets. This can be referred to as "amethodological exploration" of large data sets or "amethodological data mining". Data mining of this nature is led by the modeller's insight in exploring a virtual world of abstract financial indicators. Visualization of state is an important issue. Challenges that may be faced by the proposed amethodical exploration of large financial data sets are: (1) the difficulty of transforming experiential knowledge, gained in the course of exploration of hidden patterns in large data sets, into objective mathematically formulated knowledge; (2) the difficulty in reconciling an amethodological exploration of large financial data set with a well established econometric approach; (3) the difficulty in resorting to statistical analysis on the results of the amethodological exploration.