The Fourth World Conference On
Integrated Design & Process Technology
Incorporating
IEEE International Conference on
Systems Integration
June 27 - July 2, 1999 - Kusadasi, Turkey

Empirical Modelling of Real Life Financial Systems: The need for Integration of Enabling Tools and Technologies

Meurig Beynon
Department of Computer Science
University of Warwick, Coventry CV4 7AL, UK
Email: wmb@dcs.warwick.ac.uk

Soha Maad
Department of Computer Science
University of Warwick, Coventry CV4 7AL, UK
Email: soha@dcs.warwick.ac.uk

ABSTRACT

Understanding, analyzing, and constructing experimental models of real life financial systems is a wide-ranging task that requires an integration of different technologies and enabling tools and calls for a bridging of the gap between theory and application as well as research and development in this area. Business process modelling, intelligent state and agent-oriented modelling, data warehousing and data quality assessment tools, financial analysis tools, and client server technologies should tie up coherently to enhance knowledge acquisition in a global financial market. Players in the global financial marketplace, such as investors, rating agencies, financial service providers, analysts and consultants, are faced with a massive amount of explicit and implicit market information characterized by a high level of dependency and interrelationship. An automated environment which, as far as possible, depicts and captures all aspects of the real-life financial system, is needed to adapt to state changes in global financial markets. Such an environment should provide a flexible human computer interface backed with a high level of interaction, visualization and reporting capability, with minimal overhead coding requirements.

INTRODUCTION

Integrating existing technologies is a difficult task due to the lack of standardization and of open architecture development at the research and commercial level. This paper assesses the major problems and shifts driving the global financial market; surveys the enabling tools and technologies used in the business and financial industry; identifies trends in software development designed to meet the IT business and financial requirements; and finally suggests an empirical agent-oriented approach for the development of integrated software applications in a shared environment. The urgent need to limit the injection of new, yet redundant, technical terminologies crowding out the computer dictionary and scaring the business and financial community is prompted. This paper emphasizes the importance of analyzing the continuous state change in real life financial systems and the potential for creating an automated environment which reflects and copes with this change by maintaining agency and dependency relationhips between interacting entities involved in the system.

1. TRENDS IN THE FINANCIAL WORLD

Over recent years, the world has faced many economic crises. The most important one was the Asian crisis, which had a large impact on the world economy. Recent economic forecasts predict the development of further crises around the world.

Parallel to the economic crisis is the birth of the single European currency, which is accompanied by a high level of uncertainty and an economic slow-down in the short term. The next years will be a period of large-scale change. Competition will be higher, companies should restructure their business processes and operation in face of this competition, and many firms will fail to adapt to changing economic conditions and the new marketplace. Customer requirements will be higher, and greater efforts will be needed to attain customer satisfaction. Players in the global market have to learn, or rather re-learn, some very basic lessons, for example about the pricing of risk or about the appropriate valuation of assets in a zero inflation world.

The speed of technology change is accelerating, and this will have a huge impact on the business and financial industry. The biggest demand is for electronic communication of information and the ability to transact electronically. Consumers will benefit from technology change by getting a wider access to an array of information about products, service and performance. Technology is transforming the way industries operate

and firms can no longer hide poor products and performance.

Technology has also introduced challenging problems. No one can predict whether enough will have been done to prepare for the Year 2000 problem in a particular country or company. Preparing for the EMU and planning for euro-conversion is a business problem with complicated IT issues at its heart. It has been argued that the euro-conversion project is bigger than the Y2K project. The IT implications of the EMU are not limited to the Euro zone only, but will affect all companies worldwide at various levels following the economic environment and the general business outlook.

Today, business excellence is measured by much more than traditional financial metrics. Companies must excel at customer service, employee morale, and innovation. The ability to identify, monitor, and continually improve these vital signs determines how effectively the company will compete. Many organizations have recognized the need to look beyond traditional measures of enterprise success. As a result, performance measurement has evolved over the last few decades from simple financial measurements to measurements of more global scope. For instance, the metrics used in the 60's related to earnings, earning per share, and return on investment. In contrast, today's "balanced scorecard" is an approach that takes account of the financial position, customer satisfaction, learning, growth, and internal business performance. A major challenge in this area is linking performance measurement to the company strategies and plans (Alexander, 1998).

2. TRENDS IN THE SOFTWARE INDUSTRY

Major advances have been made in the software industry in the area of modelling, intelligent analysis, and information sharing. However, these developments lack integration, standardization, and a common technology base. These problems limit the potential benefit from advances in software development, since business success relies on an integrated and coherent use of many applications. The speed of change in the business and financial environment is accelerating. New government regulations, market requirements, and economic conditions are emerging. These factors compel the software industry to adopt the broad perspective, flexible design, and strong standards that prepare an organization for IT change. The Y2K problem, the euro-conversion problem, and the uncertain economic future create a highly challenging environment for software development. The question raised by us in this paper is whether current technology can keep up with the required huge and frequent amount of change, and enable firms to remain competitive in the global marketplace.

2.1 Major challenges facing software development

Globalization, economic and monetary integration impose new regulations on markets, and the highly competitive environment forces firms to re-engineer their businesses. There are many obstacles to exploiting computer-based technology in meeting these business challenges:

- A major reason why the benefits of software implementation have been so slow in coming is that organizational change is not adequately managed. IT is allowing us to build ever larger and more complex application systems, which are required by growing interdependent business processes, which in turn makes software development more complex. Introducing changes to complex applications is very difficult or even impossible in the absence of an interactive flexible approach to software development.
- Aligning software development to corporate goals has emerged as the number one concern for top management over the last five years. Alignment is associated with generating an adequate return on investment in information technology. Measuring the return on investment in technologies is a difficult and challenging task, given that many of the benefits derived from technology are both intangible and long term (Strassmann, 1998).
- There is currently a wide gap between research and development, and target goals and current achievement. Bridging these gaps is not only a technical issue but also entails management and communication issues. Heeks (1998) argues that the failure of most IT systems is due to a conception-reality gap between the rational conceptions of information system initiatives and the behavioural/political realities of organisations.
- Current pressures are obliging the software industry to build faster applications with more control and higher quality at lower cost and effort. However, as complexity increases, cost and development time escalate.
- Many software products launched in the market are developed using traditional programming paradigms that are not well-suited for describing the interaction in a multi-user environment with many applications.

2.2 Available tools

Many tools are available to meet the requirements of the business and financial community. These tools have been developed by many different commercial computer firms, and are tailored to suit particular business and financial needs. Although these tools are available in the market separately, business success relies on their integrated use. It is useless to develop business process models if they are not appropriately linked to the software development process; a re-engineering project might fail if the data quality in the databases is poor; a project might not finish on time if appropriate scheduling and planning is not held dynamically and is not linked to performance measurement. The following paragraphs overview representative tools and discuss their current limitations.

a) electronic document management (EDM) tools: Electronic document management tools are used to control, store and retrieve documents such as faxes, invoices, pay-slips, etc.. Reduced storage cost, security, high speed and shared retrieval, are important features of

these tools. Electronically stored documents can deliver value-added information if appropriately analyzed. Topic classification, and natural language understanding tools, which are a category of artificial intelligence tools, can serve this purpose if adequately integrated with electronic document management tools. Integrating EDM tools with the datawarehouse tools is another important, yet challenging task.

b) modelling tools: A model is a static (cool) or dynamic picture which illustrates a particular concept (a business process, a workflow, a state, a database, an interaction with an environment, an application, a logical or mathematical relationship, a situation, a solution, etc.). An important feature of a model is its ability to communicate the underlying concept to different people with various backgrounds. This assumes that all the relevant details and characteristics of this concept can be captured. Models used to document business processes and operations are often subject to standards (the US Federal Information Processing Standard IDEF0, the ISO9000 documentation for the manufacturing industry, etc.). Modelling languages have been developed for many kinds of process modelling (flowcharts, Rummler-Brache-LOV, decomposition-tree diagram, decomposition-fishbone, event process chains, action workflows, Unified Modelling Language, etc.). Modelling tools with various repository, diagramming, analysis, simulation, documentation, presentation, help, and support features are available. Most of these tools are enhanced graphical packages with a large library of icons. The models built using these tools are cool sheets which offer a very limited user interaction, typically menu-driven. Translating a model into appropriate actions, programs, and applications is a challenging task for which translators are yet to be fully developed (Long, 1998).

c) simulation tools: Today's organizations are constantly changing, and the effect of change on business efficiency is highly unpredictable and risky. Simulation tools are developed to take the risk out of change by evaluating the impact of adding a new product line, re-engineering the business processes, modelling the working environment and foreseeing the implications of different business decisions. Current limitations of these tools lie in the degree of user-interaction and in the scope of their application to different business sectors and operations.

d) tools for the datawarehouse: A datawarehouse (IBM, 1998) is a consolidation of data from disparate operational systems and sources that provides an integrated and complete information base on which to make strategic business decisions. For many companies, the process of merging and integrating data to build the datawarehouse is very difficult. Obstacles include: missing metadata; lack of standards; lack of common keys to build consolidated views; and conflicting data values. The process that merges and integrates data to build the datawarehouse is called data re-engineering. Without it, the value of the datawarehouse is diminished, often to the extent that queries to the datawarehouse yield inaccurate results. Many commercial tools are developed to build and maintain the datawarehouse, as well as to help the business user locate, understand and analyze business data. Datawarehousing tools include data quality, data cleansing, and database integration tools. Standardization, portability, platform independence, global functionality, customization, and ease of use, are important features in datawarehousing tools. A major limitation in the datawarehouse integration tools is their operation at the data level only. Integration at the conceptual level remains a challenging task for these tools.

e) business intelligence tools: Business intelligence tools consists of a range of tools simulating human expertise and reasoning, and operating on massive amounts of data. Within this category we can identify OLAP (Online Analytical Processing) tools, data mining tools, and financial analysis tools:

- General purpose OLAP tools are one common example of business intelligence tools. Data rich industries (such as the financial, marketing, consumer goods production, services, and transport) have large quantities of good internal and external data available in their databases to which they need to add value in order to gain competitive advantages in a global marketplace. OLAP tools have been developed in response to this need, to help conduct sales and customer behavior analysis, budgeting, financial reporting and consolidation, tailored and intelligent management reporting, and performance measurement. However, the success of these tools relies on a high analytical functionality, large data capacity, and an underlying integrated database. The efficiency of OLAP tools might be reduced drastically in firms operating with multiple databases and low quality data (lacking consistency, reliability, accuracy, and integration).
- Many databases contain valuable information that is not readily obvious such as patterns of high-risk companies within a financial database, types of customer in a direct mailing list who are most likely to make a purchase, etc.. The search for valuable, yet hidden, patterns and relationships within a database is known as data mining. Data mining infers rules that can guide decision making and forecasts the results of the decision. A number of different types of data mining tools are in use today, including: data visualization, neural networks, decision trees, and rule induction programs. These tools help in visualizing, detecting patterns of relationship, and inferring business rules hidden in the underlying database.
- Financial analysis tools rely on mathematical and financial models implemented mainly using artificial intelligence algorithms. These tools satisfy a range of financial analysis needs such as investment classification, financial asset classification, cash flow prediction, stock market indices estimation, international portfolio management, credit risk assessment, corporate failure prediction, rating and screening, trade decision, market prediction, risk management, and asset pricing. The main objective of financial analysis tools is to simulate human expertise in trading and investment. However, the

underlying mathematical and financial models, which are based on a set of assumptions and take into consideration a limited number of economic and financial factors, narrow the use of the financial analysis tools and their reliability in different economic environments. These tools can even prove useless when additional economic and financial factors are to be taken into consideration.

f) scheduling and planning tools: Scheduling and planning tools are used mainly in the manufacturing industry to help in better serving the customer (due date delivery of orders), eliminating the waste in over-production, minimizing inventory, and reducing the costs of resource utilization. Other areas such as daily business operations, research, event organization, and timetabling university activities benefit from this range of tools. Coping with partial knowledge and uncertainty and maintaining a dynamic scheduling and planning model is a challenging task with the available tools and technologies in this area.

g) web management tools: The huge amount of information over the internet can give firms competitive advantages if they succeed in transforming this information into knowledge and share it throughout the whole enterprise at various levels. Internet technology also offers a great potential for creating new forms of organizational links (co-operation, co-alliance, market alliance), pointing the way toward virtual integration between firms. Electronic commerce is an emerging industry with a great impact on trade promotion and mobilization, but at the same time it poses a lot of unresolved issues such as regulation of online trade, security of transactions, and control over the operations.

Many tools and technologies are required to extract information from the internet, process this information, integrate it with the data-warehouse, turn it into knowledge which increases the firm's value, and spread this knowledge throughout the enterprise. These include search engines, web-enabled databases, web intelligence, web publishing, e-commerce, video-conferencing, and information sharing tools.

The challenge for web technology is to offer a higher degree of user interaction, a greater analytical capability, and to integrate with the tools and systems used in managing the datawarehouse. Launching applications over the web is another important concern.

3. SYSTEMS INTEGRATION ISSUES

In its broad sense, integration refers to the coherent merging of two entities, having different behaviours and attributes, to obtain a unified entity that can realise the behaviours of its components and whose attributes are derived from but are not necessarily the same as those of its components. Integration might be necessary to accommodate a new style of relationship between different entities, or it might be optional with the aim of enhancing performance and gaining value-added advantages.

The term integration is used in many different contexts; it can refer to economic integration, horizontal or vertical integration of companies, software integration, system integration, etc.. In this paper, we concentrate on the integration of IT systems, focusing mainly on software integration over a distributed hardware configuration.

3.1. The background to IT integration

The first precedents for IT integration are to be found in the databases of the late 1960s and early 70s. At that time, databases typically displaced suites of application programs based on different file record formats and assumptions about physical storage (Korth et al., 1991). They also reduced the need to commission new programs or re-engineer existing programs to achieve new functionality. The key principles that emerged in this process were *logical data independence* and *physical data independence.* Arguably it was also established, whatever limitations of relational databases have subsequently become apparent, that relational theory and the analysis of functional dependencies in data are an essential aspect of maintaining successful data independence.

It is now clearer than it was at the time that relational theory supplies a database solution well-suited to a particular kind of business process model. The success of relational databases in areas such as banking depended on the highly routine nature of the transaction processing and the uniformity of the data representation demands. Subsequent developments in computer applications have exposed the need for greater flexibility in database technology than current commercial relational products have been able to deliver. The problems of integration in contemporary applications such as financial systems are further complicated in a variety of ways:

- whereas in the 70s data input was typically manual, and there were no critical real-time issues to be considered, automatic data acquisition via programs or sensors is now common;
- information is now accessed and processed for interpretation in far more complex ways, e.g. through graphical user interfaces, Business Intelligence tools and report generators;
- there is now a demand for large-scale integration of what were formerly quite distinct divisions of business activity, as in the trend to concurrent engineering, datawarehousing and comprehensive business process models.

Where state-of-the-art financial systems are concerned, these problems are compounded by the factors discussed above that promote volatility and instability of the business process model.

The most significant element here is the plethora of ways in which data is accessed and processed. This can be interpreted as a need for better models of data and agency. The focus is no longer on abstract data alone, but on the state-changing activities that surround that data, to include the protocols and interfaces of all the agents that operate upon it. In this context, an *agent* can refer to a computer program or procedure, a human agent, or an

electronic device that mediates between the internal representation and the external world. With this interpretation, agency is manifest at many levels of abstraction, in low-level data exchange, in internal business interactions, and in the external business environment (cf. Figure 1). In general, the problems of integration cannot be resolved without taking account of the multiple views imposed upon data through different types of agency. Only in quite exceptional circumstances, when there is an unusual degree of consistency in the ways that data is addressed and modified by agents, is integration of data representations sufficient.

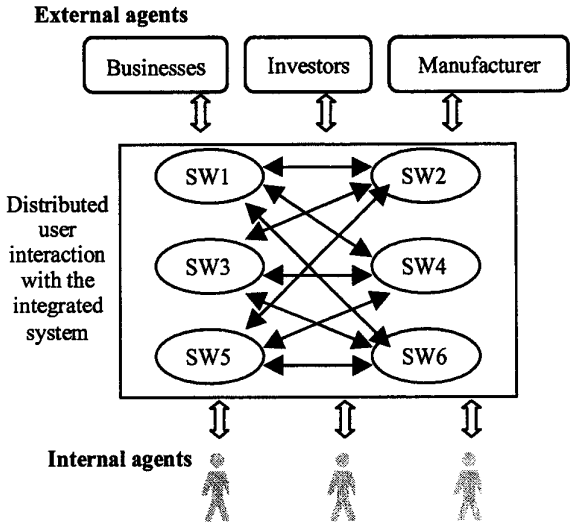

Figure 1. An integrated agent oriented system

Agency confounds the naive view that standardization of data representations and process can necessarily resolve the issues of integration. This is particularly well-illustrated by the problems encountered in commercial off-the-shelf (COTS) development of software. Relevant issues include:

- conceptual aspects of data: consider for example the very different kinds of analysis and control over geometry that are afforded by different representations (e.g. constructive solid geometry, boundary representation or implicit function);
- the character of interaction at the interface: e.g. as in seeing a picture or a postscript file, identifying the spoken or written word, accessing persistent or transient data;
- data integrity in distributed state: meaningful interaction with an aggregate of co-operating components often demands synchronized update e.g. where there are functional dependencies in a relational database, or levers in a mechanical system;
- timing issues in agent interaction: the speed at which data is processed and communicated can have a critical impact on global system behaviour.

A key problem in this context is that COTS typically have to be used in the absence of knowledge about their requirements specification and design, and without any scope to modify their behaviour. This is particularly problematic when integration is unrealistic without some element of re-engineering.

3.2. Classical approaches to software integration

Classical approaches to software integration are typically based on combining two paradigms: the use of relational methods to construct integrated data models, and the use of object-oriented methods to describe the operational processes surrounding business data at many levels of abstraction. Both relational and object-oriented modelling methods can be seen as addressing data and agency to some degree.

Functional dependencies in a relational model express the way in which a change to one item of data has to propagate change to other items in a way that is conceptually indivisible if data integrity is to be respected. In the relational theory of database design, they also supply the framework for organizing data to meet the needs of different users. The characteristic mathematical abstractions of relational theory don't capture the distinctions associated with different kinds of agency on the part of the user, however. A relational algebra expression can be used to express how one relational table is derived from others, but this may be used to evaluate a one-off query, to construct a virtual table or view, or (e.g. if associated with a spreadsheet interface) be subject to a process of continuous update in response to independent interactions with its operands. Where the human agent is in full control - as in traditional database applications - distinctions of this nature can be effectively managed without reference to explicit models of agency. Where the database is interfaced directly to computer programs, or to electronic devices with an external interface, the need for an explicit way of expressing and analyzing agency becomes evident. For instance (Carney et al., 1998), two COTS programs that access the same database table may operate quite effectively in isolation, but fail when integrated because one locks the entire table in order to access a single tuple.

Object-oriented models focus upon modelling collections of data together with the fundamental operations that can be applied to them. In this respect, they are well-suited to representing the components of distributed systems, as is appropriate in a typical context of virtual integration. What is lacking is an effective way of dealing with the complications that arise from concurrent interaction between objects. This problem has both accidental and essential aspects in the sense of F P Brooks (1995). It may be that appropriate re-engineering and the use of frameworks such as CORBA and DCOM can resolve the accidental complexities introduced when processing over several machines in a network, using different programming languages, or running on different platforms. To apply these techniques effectively, it is still necessary to resolve paradigm differences, as when trying

to convert a standalone legacy system designed without object-orientation into a modern three-tier architecture. Even supposing that this can lead to a homogeneous collection of objects communicating seamlessly, essential problems remain. It is necessary to account for active objects and autonomous agents and for the perspective that subject-oriented programming provides (Harrison et al., 1993). There are challenging and well-established problems concerned with the operational semantics of concurrent object-oriented systems. A strong body of evidence from relational database theory also argues against unsupported object-orientation as a solution to data and application integration (The Committee for Advanced DBMS Functions, 1994).

Issues concerned with agency are amongst the most difficult to represent and the most subtle to analyze and resolve. Operating systems provide the setting in which such issues have generally been encountered hitherto. It is hard to prescribe automatic solutions for the common problems of contention, synchronization and conflict that can arise when many applications are integrated. Experimental activity and insight specific to the particular situation typically rule out full automation. This is acknowledged in the design of GENIO, a software tool for data integration that has been described as a "databroker" (Porter, 1999). GENIO gives the user the means to control the scheduling and propagation of change between data representations either through direct personal intervention or by supplying parameters for automated data conversion agents.

3.3. IT integration in financial systems

Financial institutions are amongst the largest investors in computer technology and therefore appreciate the importance and significance of such technologies for their growth and survival. Financial institutions have often exploited technologies to create innovative products and services, capture market niches, and better serve the customer. The use of IT in the business and financial sector has evolved from the simple electronic data storage and limited computational capability of large business and financial databases to electronic data analysis and interchange. In the process, new forms of electronic data storage have been introduced, so that relational database management systems have been complemented by object-oriented databases and hyper-based storage. Turning information into knowledge is a major corporate challenge and the rate at which organizations learn and accumulate knowledge may become the only sustainable competitive advantage. With the growing complexity and competition in the global markets, the effective use of scarce resources and new technologies is of strategic importance. Enabling tools and technologies, such as business process modelling, intelligent analysis, data warehousing, and web technology should integrate coherently to establish corporate intelligence networks for intelligent information gathering, dissemination, and decision making. This intelligence network is critical for enabling global knowledge and information consolidation and distributing heterogeneous data relating to local markets (cf. Figure 2).

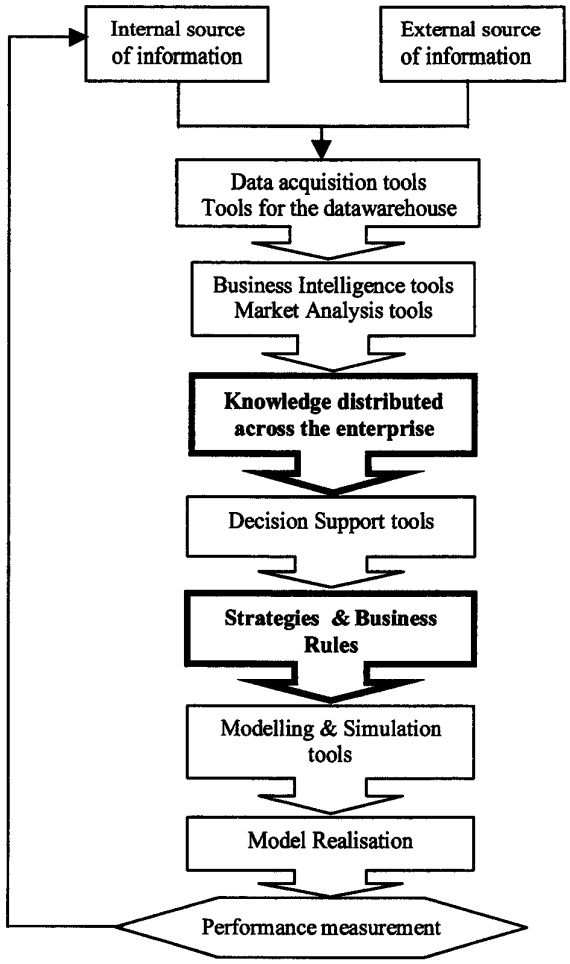

Figure 2. The integrated application chain

4. EMPIRICAL MODELLING FOR INTEGRATION

4.1. Framing the challenge of system wide integration

The above discussion has identified several key issues that have to be addressed for successful IT integration:

- dependency and the indivisible propagation of state change;
- the association of data with operations upon that data;
- the modes of agency by which state-changing activities are mediated and synchronized.

Successful integration gives users and programmers concurrent access to many software applications in a distributed environment. It must respect the integrity of data in relation to user views and the external environment. It involves combining interfaces both at the user level and at the machine level, where it provides shared access to raw data stored in system files, taking account of priority and effect. Whilst each of these issues

has been addressed individually by current approaches to IT integration based on a relational, object-oriented or data-broker models, there is a need to combine the qualities of all three.

The kind of integration achieved in the first commercial databases was tailored to a limited range of transactions within the context of a stable business process model. Financial systems, in contrast to typical customer facing systems, have to operate on a business model that is directly responsive to environmental change - rapid adaptation of the business model is essential if such a system is to take appropriate decisions in an environment of great uncertainty and flux. To meet this challenge, it will be necessary to find ways of combining software and business process re-engineering within a rapid feedback loop.

Current approaches to software development, such as Object Advantage (Jacobson et al., 1995), put a particular emphasis on integrating software development with business process modelling. These approaches aim to circumvent protracted software re-engineering processes through re-use, providing libraries of business objects for this purpose. The effectiveness of these development methods is a matter of topical investigation, but it seems plausible that they can support rapid re-engineering of software only in a context where the existing software systems are appropriately engineered and documented, and the semantics of the business process model is appropriately constrained. There are several reasons for supposing this. The overheads of present software development methods are largely incurred in the early phases of system requirements capture and specification, and in providing effective documentation and support for the activities that connect the business process model with object-oriented software specification and design. There are many difficulties in sharing and disseminating insights associated with these phases, as evidenced by COTS development in particular. There is also the possibility that some legacy components defy re-engineering, and it is unclear how to incorporate these within the object-oriented software engineering framework.

Generic and stable business models promote a vision for software development and of IT integration that is not well-suited to the requirements of financial systems. When the business process follows a preconceived pattern, it is appropriate to seek a high degree of standardization, and to make a circumscribed model of the interaction between communicating human and software agents. Current techniques for business process modelling are best suited for use in this context. Consider, for instance, the cool sheet representing a static view of the system, created using a graphical tool supported with a library of icons representing different nodes in the business process and facilitating the documentation of the business process according to enforced standards.

An entirely different framework is needed to sustain the dynamic relationship between the business process model and the intelligent analysis model that is required in the context of financial systems. The integration that is ideally required is based not on stasis, but on a dynamic equilibrium established through negotiation between a family of agents representing vectors of change associated with many different viewpoints and at many levels of abstraction. At a low level, these agents should include automated or partially automated procedures for data conversion (resembling the data brokers in GENIO) and for data acquisition. At an intermediate level, they should represent the diverse viewpoints of human agents: including non-IT-specialists, with a strategic and managerial agenda, as well as the IT users and programmers. At the highest level of abstraction, there should be ways to represent the external agency attached to the environment of the entire system. All these perspectives need to be taken into account in creating an integrated system that can be adapted in response to developments at the micro level (e.g. new hardware platforms with faster performance, new protocols for interaction), in the business process (e.g. new reporting or accounting procedures), and in the external environment (e.g. transition to a zero inflation economy, changes in share prices).

It would be unreasonable to expect integration in a system of this nature to be fully automated. Human monitoring and intervention will be required in its successful operation. As in GENIO, it is envisaged that the operation of data brokers will have to be scheduled, or even directly invoked, by a human agent. The interfaces between applications will typically encounter singular conditions for which special unspecified procedures are required. The effects of external agency will in general have to be mediated through human intervention. The degree to which integration can be achieved will depend upon the context, and upon the degree of insight that can be gained through systematic experiment and observation. Computer-based modelling techniques that are suited to the empirical development of a distributed multi-function system of this nature are the subject of the next section.

4.2. Empirical Modelling: an overview

Empirical Modelling (EM) is an approach to computer-based modelling that has been developed at the University of Warwick. It combines agent-oriented modelling with state representation based on scripts of definitions that capture the dependencies between observables. Unlike conventional modelling methods, its focus is upon using the computer as a physical artefact and modelling instrument to represent speculative and provisional knowledge. This section gives a brief overview of the main principles and tools of EM. Their potential relevance for system integration will be discussed in the following section. More details of the EM project can be found at http://www.dcs.warwick.ac.uk/research/modelling.

The central concepts behind EM are: definitive (definition-based) representations of state, and agent-oriented analysis and representation of state-transitions. In broad terms, changes of state within a system are interpreted with reference to a family of observables

through which the interactions of agents in the system are mediated. The term *observable* is used in a very general sense to refer to any conceptual entity that an agent can be construed as apprehending in a direct manner.

Empirical Modelling is a situated modelling activity that conceptually involves two aspects that are developed concurrently:

- an analysis that is concerned with explaining a situation with reference to agency and dependency;
- the construction of a complementary computer artefact - an *interactive situation model (ISM)* - that metaphorically represents the agency and dependency identified in this process of construal.

There is no preconceived systematic process that is followed in analysing and constructing an associated ISM. The modelling activity is open-ended in character, and an ISM typically has a provisional quality that is characteristic of a current - and in general partial and incomplete - explanation of a situation. The identification of agency and dependency often exploits previous knowledge and experience, but can be thought of as being derived in essence through observation and experiment.

In the analysis of a situation, key themes are:

- the identification of observables, and of patterns of correlated change to observables;
- the identification of agents as the instigators of state change;
- the classification of observables with respect to each agent, associated with the identification of those observables that are presumed to account for stimulus-response patterns in their interaction.

Working with an ISM resembles the interactive construction and concurrent use of a multi-user spreadsheet to which suitable interfaces for visualisation and possibly direct manipulation have been attached. In such a context, each user-modeller can explore and adjust their mental model of a situation through redefining cells and carrying out *what-if?* experiments. As in conventional spreadsheet modelling, the primary emphasis is not on specifying circumscribed behaviours, but on representing the modeller's expectations in the particular context that they currently have in mind.

The principal software tool that has been used to construct ISMs in Empirical Modelling is the EDEN interpreter. In the ISM, the viewpoint of each modeller is represented by a script of definitions (a *definitive script*) resembling the system of definitions used to connect the cells of a spreadsheet. The variables on the LHS of such definitions are intended to represent observables associated with the external situation. There is typically some form of visualisation attached to them, so that for example a variable can denote a point, a text message or a window displayed on the computer screen. Different *definitive notations* are used to formulate scripts according to the semantics of these visual elements.

In Empirical Modelling, all state-changing activity is attributed to agents. An agent can be a human actor, a state-changing process or component. In the ISM, the role of an agent can be played by a modeller, or by the computer. In the process of explanatory analysis of a situation, many aspects of the agency and dependency that is being identified will be implicit in the interaction between the modellers and the ISM. In understanding the behaviour of a system, it is also typically important to identify explicit protocols and stimulus-response patterns that are characteristic of agent interaction.

A special-purpose notation - the LSD notation - has been introduced to describe such agency. An *LSD account* is a classification of observables from the perspective of an observer, detailing where appropriate:

- the observables whose values can act as stimuli for an agent (its *oracles*);
- which can be redefined by the agent in its responses (its *handles*);
- those observables whose existence is intrinsically associated with the agent (its *states*);
- those indivisible relationships between observables that are characteristic of the interface between the agent and its environment (its *derivates*).
- what privileges an agent has for state-changing action (its *protocol*).

In constructing an ISM, a distributed version of EDEN can be used, where appropriate in conjunction with special-purpose auxiliary tools for animating LSD accounts. A distributed EDEN model is implemented on a client-server architecture, in which the viewpoints of individual modellers are represented by independent definitive scripts executed on different client workstations. State changes are communicated between clients by sending redefinitions across the network via the server. Communication strategies can be specified via the server to suit different purposes. In concurrent engineering, for instance, the server can play a role in negotiation between clients, resolving conflicts between design decisions, or dictating the pattern of interaction and privileges of design participants (Adzhiev et al.,1994).

4.3. Empirical Modelling for IT integration?

Software integration in general combines two kinds of activity: integrating existing separate software products and applications, and constructing new multiple-purpose software components. In this exercise in software re-engineering, the principal agenda is:

- interface design and interaction;
- shared access to data;
- synchronization of extraction, transformation, and loading of data;
- creation of a coherent and unified suite of functions.

Current proposals for software integration involve creating a metadata repository comprising profiles of each of the different applications to be integrated. Each profile is compiled from existing documentation and from the results of manual or automated code analysis. When combined with suitably engineered Common Information Models, the metadata repository supplies the resources from which the integrated system is to be developed.

The creation of a metadata repository is realistic only in certain contexts. It ideally requires the source code, if not the requirements and design documentation, for each component application. The analysis activity is an exercise in program comprehension that can be very challenging if it involves a mix of programming paradigms, or ill-documented legacy code. At best, the repository provides such documentation as is associated with requirements capture and specification in a modern software development method. This provides a static view of the system supported with static workflow models. Even where this information is sufficient to specify the behaviour of individual software components in detail, this does not address the agenda for software integration identified above. Successful integration requires crafting of the corporate behaviour of component applications, to include the specification of interfaces, strategies for shared data access and synchronised data processing, and the design of a coherent functionality.

This section briefly examines the prospects for using EM to complement orthodox approaches to integration (cf. Figure 3). Our proposals draw upon previous research into the role of EM in software development that, in particular, examines how:

- agent-oriented analysis combined with ISM construction assists program comprehension and the understanding of requirements (Sun et al., 1998);
- software modules can be extracted from ISMs and, if necessary, optimized by translation into conventional procedural programs (Allderidge et al., 1998).

simulating the execution of agent protocols, for example by using EDEN. In this way, two or more modellers can play the roles of agents within the system independently.

Definitive scripts provide a powerful means of data integration that can be used in particular to express the way in which low-level redefinition can entail high-level change. Consider for instance how the interest rate can change when a balance crosses a threshold. Data conversion agents that are empirically tuned to particular patterns of synchronization can serve as databrokers. In EM terms, what has been described in Figure 2 as the conversion of data to knowledge is merely one aspect of pervasive mechanisms and processes that mediate between the viewpoints of one agent and other.

In EM, constructing an ISM addresses requirements understanding for software development in a way that circumvents the limitations of normal documentation. An ISM represents knowledge in an implicit and experiential manner. The modeller can develop, access and explore understanding through interaction with the ISM, and share this insight by presenting the ISM to another modeller. ISMs constructed using EM principles are so general as to encompass traditional engineering or scientific artefacts that are devised to capture empirical insights. Experimentation with legacy components of a software system can be used to develop ISMs in a similar way.

In applying EM to software integration, the key idea is to understand each software application in agent-oriented terms with reference to the particular observables that mediate its interaction with other agents in the system.

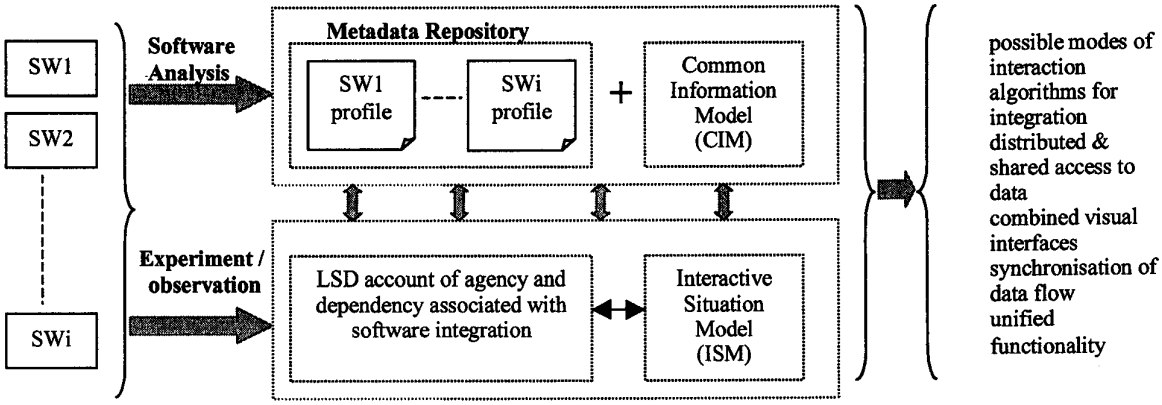

Figure 3. The use of EM to complement the conventional approach to software integration

EM principles to software integration are well-oriented towards the key issues as set out in section 4.1. Definitive scripts deal explicitly with dependency and indivisible propagation of change. The definitions in such scripts can be grouped in many ways, as their order is not important. By collating the state observables of an EM agent, as specified in LSD, an object-like abstraction is obtained. The permissible operations on such an object in general include redefinition of its state observables through the direct action of other agents. The operational effect of concurrent agency can be empirically established by

This analysis is not simply concerned with abstract inputs and outputs, but with the way in which interaction is embodied at the interfaces to other applications. This embodiment of inputs and outputs is metaphorically represented by an ISM and its associated LSD account, as developed in parallel. Such ISMs enable an experimental study of the modes of interaction between both existing software applications and those yet to be developed (cf. Figure 3). They can also help to address issues of scalability and customization.

The development of ISMs for software integration can draw upon ideas and techniques introduced in previous research. These include:

- the construction of ISMs based on animating conventional static artefacts (such as object models and statecharts), both to introduce models to the data repository and to refine and to exploit them;
- collaborative interaction of potential users of the integrated systems (such as the internal and external agents in section 3.1) through networked ISMs in a distributed environment;
- the intervention of the modeller in the role of an all-powerful agent (e.g. to shape the synchronization of interaction, to resolve conflicts between viewpoints and to compensate for incomplete rules for data brokering).

The EM approach also promises to deliver a model of an integrated system such as is envisaged in section 4.1. This model is typically neither static nor comprehensive in character, but comprises a loose association of ISMs constructed from the viewpoints of different agents within the system, each reflecting different observables, dependencies and types of agency. Integration is achieved through a dynamic empirical process of negotiation between these viewpoints. This process in general entails compromise, and may require intelligent intervention by a human agent acting in the role of an arbitrator or broker.

5. CONCLUSION

This paper questions the effectiveness of current approaches to software development for the integration of enabling tools and technologies. It advocates the development of EM as a new computer-based modelling paradigm that promises more direct ways to revise systems in response to changes in real-life requirements. This revision can be guided by experimental interactive activity, and in some circumstances may be successfully applied to incorporate legacy components for which no requirement and design information is explicitly known.

The software industry is becoming highly volatile due to the introduction of many products, technologies, and terminologies which give minimal added value and introduce confusion and controversy. Successful integration and standardization is the key to creating a stable environment in which innovation can have a cumulative rather than divisive effect.

From an academic perspective, it is clear that the relational models for data integration that Codd introduced in the 1970s are insufficient to meet the needs of data integration for the year 2000. There is an urgent need for theoretical principles that can help to classify and discriminate between software tools in a context where new technological developments are rapid and commercial hype is rife. There is a particular danger of automating unprincipled decision-making beyond the control and comprehension of human agents. A most important aspect of the EM agenda is to seek foundations for new modes of human-computer collaboration that can help to realise the potential and recognise the limitations of emerging technologies and tools (Beynon, 1999).

ACKNOWLEDGEMENTS

We are indebted to many members of the Empirical Modelling group for relevant ideas, and to Pi-Hwa Sun in particular for his work in developing a distributed version of EDEN for constructing interactive situation models.

REFERENCES

1998, "IBM's Visual Warehouse and Data Cleansing and Reconciliation", Technical Brief, IBM & Vality Technology Incorporated.

Adzhiev, V.D., Beynon, W.M., Cartwright, A.J., Yung, Y.P., 1994, "A Computational Model for Multi-agent Interaction in Concurrent Engineering", *Proceedings of CEEDA'94*, Bournemouth University, pp. 227-232.

Alexander, R., McCann, M., 1998, "Balanced Scorecard Measurement", *Proceedings of BPR Europe 98 Conference*, London.

Allderidge, J., Beynon, W. M., Cartwright, R., Yung, Y. P., 1998, "Enabling Technologies for Empirical Modelling in Graphics", *Proceedings of Eurographics UK*, 16th Annual Conference, pp.199-213.

Beynon, W.M., 1999, "Empirical Modelling and the Foundations of Artificial Intelligence", *Proceedings of CMAA'98*, Springer Lecture Notes in AI, to appear.

Brooks, F.P., 1995, "The Mythical Man-Month Revisited: Essays on Software Engineering", Addison-Wesley.

Carney, D.J., Wallnau, K.C., 1998, "Pro/COTS:Process Issues in Building COTS-based Systems", *Proceedings of ICSEA Conference*, Vol. 2, Paris.

Harrison, W., Ossher, H., 1993, "Subject-oriented Programming (A Critique of Pure Objects)",OOPSLA'93, pp. 411-428.

Heeks, R., 1998, "Why Information Systems Succeed Or Fail: Conception-Reality Gaps", *Proceedings of BIT98 Conference*, Manchester.

Jacobson, I., Ericsson, M., Jacobson, A., 1995, "The Object Advantage: Business Process Reengineering with Object Technology", Addison-Wesley.

Korth, H., Silberschatz, A., 1991, "Database System Concepts", Second Edition, McGraw-Hill.

Long, K., 1998, "Process Modelling Tools, Demo's, and Evaluations", *Proceedings of BPR Europe98*, London.

Porter, R., 1999, "Integrating Databases", *Genio Seminar*, Leonard's Logic.

Strassmann, P.A., 1998, "What is Alignment?", *Cutter IT Journal*, Vol. 11, No.8, pp.4-9.

Sun, P.H., Beynon, W.M., 1998, "Empirical Modelling: A New Approach to Understanding Requirements", *Proceedings of ICSEA Conference*, Vol. 3, Paris.

The Committee for Advanced DBMS Function 1994, "Third Generation Database Systems Manifesto", from Readings in Database Systems, 2nd ed., Morgan Kaufman.