DEVELOPMENTS IN THE CASE MARKET

David Levy
Pyramid Technology Ltd
United Kingdom

SUMMARY

This paper addresses a series of trends in the UK computer systems development market. It addresses the and technologies used to develop business application systems in the commercial market as opposed to technical/scientific or CIM.

It looks at the Structured Systems Analysis & Design Method (SSADM), the UK Government mandated development methodology and compares it with Oracle's variant of information engineering; and looks at the strengths and weaknesses of these two methods. It also examines the cultural requirements and ethos of these methods.

The second strand of this paper looks at the technologies, both hardware and software, in use in these market places, suggests some likely developments and points to productivity benefits accruing to the use of fourth generation languages and smaller inter-disciplinary teams using precise languages of communication (easily lending themselves to machine manipulation).

It then looks at the relationship between development methods, technologies and business culture.

A DEFINITION OF CASE

This paper is primarily concerned with methods and the automated tools that implement systems analysis and design methods. The issues of version control, software maintenance and software proving are not covered. In this arena, CASE is becoming synonymous with diagrammers, dictionaries and code generators.

The purpose of CASE and systems development methods in general is to increase the productivity of software development staff. This can exhibit itself in several ways.

Larger projects can be undertaken with smaller teams, projects are delivered with lower costs, due to cheaper (ess skilled) staff or due to lower numbers involved. In addition, a major advantage claimed by CASE vendors is increased software quality leading to lower software maintenance costs. However, not all projects organisations can take advantage of these products and the projects on which these products are applied must be carefully selected. Many of the advantages that may be argued belong to a CASE product in fact are derived through the use of a method, 4GL or RDBMS.

A REVIEW OF SSADM - STRENGTHS AND WEAKNESSES

Early in the 1980's, the UK Central Computing & Telecommunications Agency (CCTA) commissioned a competition to design a systems development method to complement their highly successful introduction of Michael Jackson's structured programming techniques into government computing. The central features of this method were:

- Rigid stage definition
- A distinction and comparison of logical and physical data
- Bottom up process analysis

The major tools used were:-

- Logical data structuring (entity relationship modelling)
- Data-flow diagramming
- Relational data analysis (third normal form analysis)
- Entity life history diagramming

The method gradually grew through central Government and, as the productivity benefits (admittedly in conjunction with other unexplored factors) led to the government taking on major projects, its demand for external consultancy skills rose. The CCTA's requirement that all projects use SSADM led to the major consultancies adopting the method as their strategic Systems Design Methodology. These consultancies then offered the method and its private sector badged version (LBMS) to private sector clients.

The method itself is now showing its age and the CCTA have commissioned a rewrite. Apart from certain technical deficiencies and the way in which it has been overtaken by certain technologies, such as

4GLs and development methods such as prototyping, SSADM had some cultural attributes which make it particularly appropriate for the UK public sector.

The method worked well with the principles of the Government project management methods (PROMPT) of completing and signing of a stage before the commencement of the next. It also ensured that. a complete systems architecture, costs and implemention plan was available prior to purchasing (what was then) expensive server (mainframe) hardware, and embarking on (even more potentially) software development involving many man years of COBOL code development.

Even its greatest advocates will today admit that it is a document intensive method. An organisation with a tradition of documentation management such as the British Civil Service was able to consider the costs of such an undertaking with equanimity, particularly as the method's own benefits helped to finance this overhead. However, most private sector sites would today be unwilling to take on the method without dictionary and diagrammer support. The method required quite a sophisticated diagrammer since it uses in its full implementation four separate diagramming conventions with several sub types beyond these.

The method and its detailed and rigid documentation standard meant that a project team was no longer reliant on the performance of key individuals since their efforts were documented in the same way as everyone else. It allowed method trained people to fit into teams very quickly, allowing management to cope with high turnover caused either by external labour market forces or by the deliberate use of external labour to overcome short-term skills shortages. The bottom line was that management were able to use the method to manage a skills shortage by directing their training investment and the consultancy purchasing. These advantages allowed them to pursue a policy of (in industry wide terms) employing lower skilled but highly trained staff. (Many of these staff members were recruited internally and had a good understanding of the organisational practices and procedures.)

The designers of the method understood the importance of diagrammatic techniques and feedback to users at all levels. However many staff became over committed to the bureaucratic minutiae of the method. In addition the method and the business culture of the organisation that controls the method (i.e. CCTA) is one that seeks to avoid failure and embarrassment, in particular costly failure. Decisions need to be accountable both to senior authority and to parliamentary scrutiny. In this light it can become more important to find a culprit rather than maximise the chance of success. To emphasis this point, it is a strategic goal of the organisation that its systems development methods has an internal audit trail as well as producing accurate development outputs such as data models or working systems. Even successful systems may be subject to scrutiny to see that greater success might have been achieved. The culture is about identifying the culprit as well as avoiding the mistake. In the private sector, management is, or if it is not, should be more accountable to the success or failure of a project.

A REVIEW OF AN INFORMATION ENGINEERING METHOD (ORACLE)

The Oracle method was originally purchased from its originators CACI, an American owned systems consultancy who had specialised in methods and database systems building. The CACI method was taken out of CACI and popularised in the UK by a number of leading methods industry commentators. It might seem that held against the industry predominance of SSADM, there could be no competition. However, the fact that the James Martin Associates Information Engineering method, the Oracle CASE* Method and the smaller methods vendors recognise that common heritage allows methods vendors to treat "Information Engineering" as a family of methods with a degree of openness.

These methods all use:-

- Entity modelling
- Data-flow diagramming
- Functional decomposition
- Normalisation

There is generally an exercise to ensure that all identified "events" are consumed and that all entities go through a birth, retrieval, update and death. These tools differ, sometimes being simply a diagram, sometimes an action diagram etc.

In these methods, the emphasis placed on data modelling is considerably higher than under SSADM since it is not modified by any concept of validation against a physical data model.

These methods can be characterised as:-

- requiring people highly skilled in IT or business techniques
- supporting prototyping
- possessing no bias towards a specified business culture

The less rigid mechanisms of these methods to more money being available to spend on staff costs themselves or external consultancy skills. The greater importance placed on the data model and the less formal process modelling again place a premium on the skills of the individuals creating the documents.

Perhaps because of the less rigorous approach to the techniques of business modelling, but also because of the changes in software technology, these methods have all been built to permit prototyping. While the management of prototyping is not well understood, these methods (particularly the use of functional decomposition) allow the early identification of suitable components or business areas for prototyping.

These methods are much more a framework than SSADM and are usually sold by consultancy based companies looking to increase their sales beyond the method itself. These factors mean that these methods can be customised and that experience customising the methods is plentiful. The methods themselves are not imbued with a specific culture. A company's own preferences can be taken into account in a way that is difficult for SSADM due to the fact that one of the CCTA's strategic aims from SSADM is uniformity across its purchasing domain.

IMPLICATIONS OF TECHNOLOGY

It has been mentioned above that the developments in technology have changed both the requirements customers put on methods and on the methods themselves. The economics of systems development have been effected by developments within the industry.

The rise of the RDBMS to a position where it delivers the performance, functionality and software development productivity is one such factor. Oracle V6 is in a position to offer mainframe database performance with a relational flexibility. The dynamic nature of catalog tables, and the elimination of data solely concerned with integrity means that a database structure is now easily amendable and allows a massive gain in productivity to applications coders and database designers. The use of SQL as a standard also allows a concise and exact pseudo language with all the non-procedural power that programming managers want, to be used by business and systems analysts even before a DBMS is chosen.

The flexibility of the RDBMS and the growth of 4GLs are the primary technologies that make prototyping possible. The combination of user staff and system analyst allow the use of cheaper systems to identify a user's business and ergonomic requirements more speedily (if well managed) and with more certainty. One of the contributions of this tendency is the fact that Oracle and the other major relational software vendors are hardware independent. A customer can cheaply start prototyping and designing using CASE tools but delay the larger costs of the production hardware machine and software licenses.

Underlying these developments is a growth in the price performance delivered by the hardware manufacturers. The growth in power of microprocessors has been immense led in particular by RISC implementations. Without the fact that power only available yesterday in a mainframe is becoming available on a desk top, RDBMSs would not be able to exploit the market as successfully they can, let alone the ,more complex software applications required to support a methods led development. Additionally, the concept of system scalability, where one hardware vendor can offer field upgradable increments of power, allow customers to buy the cheaper, development computer systems and upgrade them as necessary to take on the production load by essentially adding CPU boards, makes the prototyping approach feasible.

The other software technology development surrounds the issue of graphical user interfaces and video screen hardware that supports them. In leading software development houses, windowed interfaces to computer systems have been delivering enhanced productivity for several years. These benefits are also available to users on production systems. The ability of a user to simply choose a series of windows suitable to the way they do business will allow systems developers to simplify their "forms". This reduces the cost, developing and increasing their performance. The other side of this coin is that the single page form filled to the extent of illegibility could be on the way out. Windows is also an ideal way of mixing data formats such as pictures from an "Imagebase" and quantitative data from a database.

Performance of Leading Edge Workstations

year -1984

WS MIPS = 2

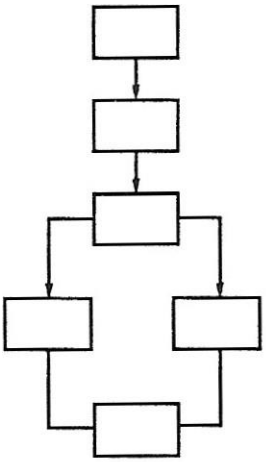
1984	1 MIPS	Sun - 1
1985	2 MIPS	Sun 2/160
1986	4 MIPS	Sun 3/260
1987	8 MIPS	Sun 4/260
1988	16 MIPS	STELLAR MIPS co
1989	32 MIPS	

Figure 1

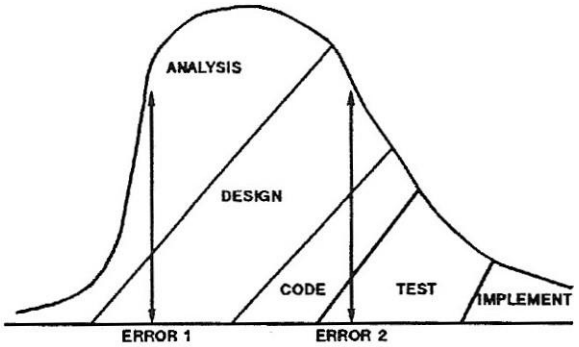
SYSTEMS LIFE CYCLE IMPLICATIONS

The drive for enhanced productivity in the systems development cycle has led to the development of methods, the use of advanced software and high power scalable hardware for the use of analyst workbenches and prototyping. These two developments have a major effect on the desirable life cycle.

SSADM Life cycle



Prototyping Life Cycle



Error 1 Error discovered in design requires re-analysis

ErrM2 Error "ccovered testing reqJIre8 re—design and consequent recoding.

Figure 2

High manpower projects with expensive hardware required rigid stage management in order to maintain quality and ensure costs stayed within project budgets. The rigid staging allowed user management to review the applicability and justification of the proposed systems as the project progressed to ensure it continued to meet a business need and remained cost effective.

The growth of RDBMSs , analyst work benches and prototyping have meant that it has become desirable to be able to amend what have traditionally been the outputs of earlier stages, and immediately cascade those amendments through the intervening stages to the current phase of the project. An error discovered during testing might involve the reassessment of analysis outputs, amendments to design and recoding in order to continue with system testing.

A LOOK AT TOOLS REQUIREMENTS

There is a conflict between maximum management control and development productivity, both for the human resource involved and the capital resources used. The management response is to develop new cost management techniques such as the iterative or throwaway prototyping methods, or the use of prototyping deadlines for cost control purposes.

The developer tool sets require higher levels of integration and a more dynamic interface with the development IY0duct. This level of dynamism and the higher level of user involvement implies a requirement for sharing data between members of a project team.

This management requirement to share information in order to resolve conflicts of interpretation is important since it is members of the project team who will resolve any conflict, not the tool set. This distinction is one that many data analysts and dictionary administrators do not understand or accept. 'Ihe requirements for information sharing, version control and prototyping in an integrated environment all point towards the superiority of a multi-user tool.

MARKET DIRECTIONS

The market for analyst/designer workbenches can be seen as coming from a diagram background often based on PC technology as opposed to coming from a database dictionary based approach starting with the inherent features of multi user support possessed by all databases.

The traditional workstation vendors, all owned by US companies, have picked up the popular methods from the US which owe much more to process analysis than the two UK methods reviewed above. This makes many of the CASE products offered on UNIX based workstations unsuitable for development in a commercial environment. The decision to purchase a development product will be driven by the use of a method and the methods supported have limited application in the UK labour market, as employed staff or consultancy support is easy to find.

The market is divided between the products aimed at an SSADM environment, those aimed at an IBM environment and the multi-user analyst work benches. Almost uniquely, Oracle offers the latter in an open systems environment.

CONCLUSIONS

In conclusion, the development of methods has allowed the use of new technology productivity tools. This in itself has allowed the use of smaller project teams which has led to more effective communication between the user community and the development community which is a productivity gain. These smaller teams are more business oriented.

These are the key productivity gains which successful CASE product vendors and purchasers will address. These systems must be multi-user and support prototyping through its full support for the whole project life cycle.

The choice of method, and hence of tool, must take into account the business culmre of the organisation itself. Enhancing communication between members and groups in the organisation is at the centre of all attempts to enhance productivity within the software development department.