Alternative Technologies

An Overview of Migration Strategies

David McGoveran
Alternative Technologies
13130 Highway 9, Suite 123
Boulder Creek, CA 95006
Telephone: 408/425-1859
FAX: 408/338-3113

I. Introduction

Over the years, IS (information systems) has developed a number of strategies for managing technology change. I will describe five of these "traditional" strategies here. For convenience, I have given them names: direct replacement, staged migration, anticipated migration, competitive replacement, and extinction.

Among the more recently developing strategies is the notion of integration rather than replacement. This strategy involves a policy of co-existence that can breath new life into legacy applications in a variety of ways. In addition to discussing the key migration strategies for managing legacy database applications, I will introduce the co-existence strategy with which Oracle Corporation is addressing those issues.

II. Existing Environments

For the purposes of understanding RDBMS migration issues, it is useful to classify existing applications into one of three categories. These are legacy applications, pre-relational applications, and early relational applications.

A. Legacy Applications

The term "legacy application" has become a popular industry term without much in the way of definition or understanding. Loosely speaking, it is used to refer to those applications which do not go away, cannot be replaced, and do not take advantage of today's technology. I would like to take a more careful approach to the term and propose a somewhat more precise, perhaps even narrower, interpretation.

By definition, legacy applications are difficult to replace, expand in functionality, or even integrate with newer applications. There are five characteristics which make this true. First, they may be poorly documented. Second, they may be poorly designed. Third, they may represent a very large investment which management is afraid they stand to lose if the application is changed. Fourth, they may be so complex that modification is risky. Fifth, they may be so mission critical that modification is risky.

These characteristics stem from the design and development practices of the past, the absence of standards, and business needs which demanded rapid delivery at any cost.

B. Pre-relational Applications

The second class of applications are those which I would term "pre-relational". These applications depend on non-relational data management such as file systems and hierarchical or network DBMSs. As such, although they may not be poorly designed and may even be well documented, they may well be at odds with relational concepts and practices.

C. Early Relational Applications

The third category of applications might be called early relational. These applications are characterized by the minimal degree to which they take advantage of RDBMSs, and by the limited relational capabilities supplied by early RDBMS products. Both of these characteristics arose because of the limited understanding of both vendors and developers of relational concepts.

III. Traditional Strategies

A. Direct Replacement

The most severe and risky policy is that of direct replacement. The idea is that a new application is written to replace the old one. As soon as the new application is tested, the replacement phase begins. This phase may be realized through either a cutover or parallel operation. Both techniques have benefits and costs.

The benefit of cutover is that the focus of IS and users is never on more than one system. The major cost of this strategy are risks that the new system will does not constitute a full functional replacement, that it contains bugs, or that users will not find it operationally acceptable. Thus, cutover is extremely risky unless the old system is adequately documented (or documentable after the fact), thorough testing can be performed, and the user community can be adequately trained to accept and use the new system.

The key benefit of parallel operation is that risk is minimized. Should the new system prove inadequate for any reason, it is possible to revert to the old system, old data management system, old operational procedures, etc. However, parallel operation is extremely costly to implement. Users and administrators must be prepared to use and manage two systems at once.

Sometimes it is possible to minimize the impact on end users by implementing a dispatch system or message switch which multiplexes the user interface. Both systems are updated via a single source of data entry and a single system is used as the primary source of data. Unfortunately, it is rare that legacy applications permit such architectural changes.

Parallel operation implicitly requires some method of synchronizing data between the two systems. This is a difficult task which has been managed in several ways. The most common technique is batch reconciliation in which a batch program which identifies discrepancies between the old and the new system is periodically run, producing an exception report. The difficulty is in interpreting and responding to exceptions.

One might think that it would be easy to identify either the old or the new system as the correct one and respond accordingly, either modifying the new system to produce identical output or ignoring the differences as indications of problems with the old system. Unfortunately, this assumes that the data semantics of both the old and the new system are well understood, are comparable, and that no systemic errors exist.

If data were maintained in a clean manner and were well-defined, these would be easy criteria to meet. In fact, most data from older applications is in bad condition. It is not uncommon to find data entry errors, semantic errors (i.e., meaning of meaning), and orphaned relationships. These are very difficult to identify and correct. Conversion from one data base to another demands the correction.

B. Staged Migration

Staged migration is a technique similar to direct replacement, but in which separate modules are replaced rather than the entire application. Replacement occurs selectively, forming ever growing islands of newer technology. Typically, these islands are interconnected by an integrated RDBMS data model and possible through distributed computing techniques as well. If the RDBMS supports distributed database management, the databases associated with the individual islands can often be integrated with relative ease.

The risk of staged migration is that no coherent data model is ever developed. In such a case, the application environment becomes fragmented and systemic errors become almost impossible to identify and correct. The key benefit results from the more leisurely and thoughtful pace at which migration can take place.

C. Anticipated Migration

Anticipated migration refers to a technique that is possible if the designers and developers of the older application planned for eventual replacement. In particular, the application architecture must be highly modular, and the interfaces and functionality well-defined. This permits selective replacement of individual modules. Unfortunately, it does not address the problems of data store conversion.

When the data store can be segmented so that a particular set of the data can be identified as being affected by a particular subset of the total set of transactions in the application, it is possible to convert the data store in a modular fashion. The technique of transaction analysis is essential in planning for database segmentation. This is a difficult effect to achieve because most databases contain tables which are highly interdependent and are difficult to segment.

D. Competitive Replacement

Competitive replacement takes a somewhat Darwinian view of the software evolution process. For a particular application requirement, several different groups may be given the task of creating a new system. These then undergo trial adoption by the user community and "compete" to be selected.

Competitive replacement has the strength of providing the user community with multiple options. It assumes that the user community will select the system that is best for them and that this will also be the best system for the business. However, such criteria are rarely successful over any reasonable period of selection; by the time systemic errors are identified, the selection process has ended. In general, data integrity problems do not have a directly visible affect on end-users and, at any rate, are difficult for end users to understand. Competitive replacement also does not work when the application needs to be highly integrated within an enterprise framework.

E. Legacy Application Extinction

Another technique that is sometimes used is that of selective extinction. Certain applications have a naturally limited life expectancy, usually due to changes in business practices. For example, an application may be designed to support specific products sold by a company. Rather than modify the existing application to support new products, an entirely new application is designed. As the company moves away from the older product, the older applications cease to be useful and are effectively "extinct".

This technique, like some of the others discussed above, makes it difficult to create an enterprise-wide, tightly integrated system. Furthermore, it is not always applicable. Many applications are not so closely identified with a system, product, or procedure that is being replaced in the business. Rather, these business requirements change much more smoothly.

F. Productive Co-existence

The newest migration strategy is that of productive co-existence. Productive co-existence is being heavily promoted by Oracle Corporation with the introduction of ORACLE Version 7. In this strategy, new software systems and technology are adopted where possible, leaving the more deeply entrenched legacy applications in place while establishing a framework for cooperation between the systems. The strategy of co-existence attempts to address the deficiencies of older systems, especially their rigidity, by developing highly adaptive new systems. These new systems are made possible by certain new technologies, such as distributed RDBMSs, open and custom gateway technology, and standards-compliant interfaces.

The key benefit of this strategy is that many of the migration techniques used in other strategies can be selectively implemented without introducing significant conflicts. Each legacy application (and indeed individual sub-systems) can be evaluated separately as a candidate for migration. On the one hand, if migration or replacement is not deemed cost-effective, the application can remain in place and new applications can take advantage of it and its data. On the other hand, if either migration or replacement seem to be cost-effective, the transition can be made at any time so long as the application is relatively independent of other systems.

Of course, productive co-existence does place a heavy burden on RDBMS functionality. The RDBMS becomes the center of migration, integration, and interoperability efforts. As such, it cannot be less than the very best product available for the job. Chosen incorrectly, the RDBMS can lead to weakening and eventual disarray of the entire IS infrastructure. Chosen correctly, the RDBMS will enable and strengthen efforts to move into the next century of computing.

IV. Conclusions

In this paper we have discussed various strategies for migrating legacy applications to new technology and have discussed some of the costs and benefits. Productive co-existence, as a superset of the various migration strategies, may well be the only reasonable approach to solving the obsolescence of applications. At the very least, it does not assume that every legacy application needs to be replaced in order to take advantage of newer technology. Ideally, it extends the window of opportunity to recapture past investments by expanding the functionality of legacy applications at a low cost.

References:

D. McGoveran, "Enabling Productive Co-existence," c. July, 1992, Alternative Technologies, Boulder Creek, CA.

D. McGoveran, "Application Migration and Portabilty", Database and Application Development in the '90s Conference Proceedings, October 1-4, 1991, Database Associates, Chicago, IL.

D. McGoveran, "ORACLE7 Database Evaluation", Database Product Evaluation Report Series, c. June, 1992, Alternative Technologies, Boulder Creek, CA.