INTRODUCTION

The result of leveraging, successfully done, is a continual reduction of both the up-front costs associated with acquiring and installing software applications as well as the long-term support costs. These cost reductions are achieved through economies of scale realized as common elements of a software application are repetitively applied and costs are prorated across a larger set of installations.

This article examines the issues of leveraging application software in a manufacturing context. Leveraging, or the reduction in application life cycle costs (illustrated in Exhibit 1), can be discussed in either a context of spanning multiple manufacturing locations within a single business unit or spanning multiple businesses within a single product type.

MAINTENANCE AND SUPPORT COSTS IN MANUFACTURING ENVIRONMENTS

The process of acquiring automation, process control, and plant information systems has traditionally involved one-of-a-kind development projects. The needs of each site were individually assessed, software was selected and purchased, and local resources (often in the form of system integrators) were contracted to provide services. This business paradigm is easily executed, but it has resulted in a profusion of unique, site-specific systems. By multiplying atypical applications, manufacturers discovered that the costs of supporting these unique sets of applications increased because of the lack of commonality or no economy of scale. Compounding the problem is the plethora of applications serving different needs or functions of the manufacturing user communities.

Business structure and organization are also issues to consider. Manufacturing sites are usually autonomous, with at least cost center responsibility if not profit-and-loss responsibility. A large manufacturing site may make multiple products or product grades. Production areas within a single site are usually product defined. A single product may be made at multiple sites for sourcing, distribution, and marketing reasons. Business units are formed around one or more product lines. A business unit may span multiple products on multiple sites. Hence, any given site may support many business units.

PAYOFF IDEA

Leveraging is the reusability or portability of application software across multiple business units. Leveragability is the extent that the application can remain unchanged as it is installed and made operational at each location. Because leveraging can reduce the cost of acquiring and maintaining application software, it should be part of the IS strategy.

Information and control systems (e.g., process control, product control, and quality control) are usually required to be in-plant systems. Enterprise systems (e.g., material requirements planning or MRP,
warehousing and distribution, and order management) are most often shared across multiple sites as supply-chain functions.

Manufacturing applications that directly effect the manufacturing process have historically been contained on site. This situation has resulted in a physical architecture different from a central mainframe with dumb terminals at remote sites. Whereas manufacturing applications are often compute-intensive, few simultaneous users are served.

Unique site manufacturing applications crowd the IS landscape, owing their existence to differences in product or area requirements. Factors that have contributed to this condition include:

- The need to exchange process control data with other manufacturing systems.
- The autonomy of manufacturing sites to make IT investment decisions.
- The lack of a vision for future integration of direct manufacturing systems with other plant IS applications and systems.¹

Historically, manufacturing applications were constructed in a purely vertical sense with automation, not integration, in mind. Shared applications, while functionally isolated, were often interfaced with other applications through a variety of means. The resulting set of disparate and unique legacy systems has driven support costs higher, even as support resources are shrinking.

One textile fibers manufacturing business estimates that for every dollar invested in development, $0.25 per year is incurred for maintenance and support, including both direct and indirect costs.² Taken over a 10-year anticipated life of an application, this amounts to a present value for support of about 2.5 times the total costs of the initial development.
ECONOMICS OF LEVERAGING

As leveraging is foremost a business objective, it is important to note the economic effects of leveraging as a capital decision process. As leveraging occurs, the costs of application software go down per site. Assuming benefits from the software are constant, the net present value (NPV) of the per-site investment increases. Exhibit 2 shows a sample discounted cash flow (DCF) curve from an initial investment in an application.

The costs associated with bringing a software capability online at a site include the following cost categories:

- Initial planning, analysis, consulting, and specification.
- Construction/implementation, staging, integration, and factory acceptance.
- Hardware, system software, and networking.
- Installation, database population, commissioning, site acceptance testing, and training.

Taken together, these cost categories can be counted as the investment toward an anticipated improvement in manufacturing operations. Most investments are driven by an anticipated series of positive future cash flows. This is, of course, standard financial analysis for capital decisions. The word “benefits” is frequently used to describe enhancements to the manufacturing operation that result from using an application. For an investment to yield a net present value, the sum of the present value of the future cash flows resulting from the initial investment must be greater than the present value of the costs associated with realizing the benefits.

The investor (plant site) expects benefits (future cash flows) from the investment (initial costs) at an appropriate discount rate. One method of organizing and analyzing benefits in manufacturing seeks to maximize net present value of a set of information system projects identified through a strategic planning activity.iii
Exhibit 2 illustrates the shape of the curve of cash flows over time when costs are assigned as negative cash flows for an application project and benefits as positive cash flows. Development costs (cash outflow) initially cause the curve to go down. After commissioning and allowing some time for use of the application to reach maximum effectiveness, cash flows become positive as benefits (cash inflow) begin to be realized. Support costs (cash outflow) continue but should be small compared to the benefits accruing per period. Discounted cash flow causes the net cash flow over time to steadily decline, assuming constant support cost for the application. A break-even point occurs when cumulative DCF is equal to zero. That is, the present value of application benefits is equal to the present value of costs.

The principal business drivers for leveraging are economic, not technical. A successfully applied program of leveraging an application or capability across multiple manufacturing sites reduces the installed costs per site while minimizing the ongoing support and maintenance costs of the delivered applications. Assuming manufacturing benefits result from the application, the result of leveraging is a maximum net present value of the investment across one or more manufacturing sites.

Exhibit 3 illustrates the economy-of-scale effect as a measure of the resources required per installation. Leveraging has the effect of driving the total costs per site to some base level that is set by the costs of off-the-shelf system components plus resources required to install and make operational the system at each respective site.
Leveragability, then, can be economically measured by the extent of the costs associated with planning and implementing each site’s respective requirements. Exhibit 4 demonstrates the effect of leveraging as the number of sites to receive the application increases. Leveraging is therefore an economy-of-scale effect. The greater number of sites in the leveraged effort, the greater the net present value of the investment across the collection of target sites or installations.

**CONSISTENT DATA**

Because the business drivers for leveraging are clear, it is reasonable to ask why leveraging is not a pervasive business practice. Some of the barriers to achieving leveragable software include:

- Misperceptions of leveraging.
- Absence of a long-term manufacturing applications migration plan.
- Lack of consistent architectural framework.
- Corporate culture (“not invented here” thinking).
- Ad hoc approaches to applications development.
- Conflict between corporate IS/engineering and the manufacturing sites’ objectives.

Whereas application leveraging does not have to mean “one size fits all,” some consistent framework must exist so applications can be designed consistent with this framework. Although the data content of applications varies between businesses or sites, applications should fit into a common architecture across the domain of sites or businesses over which the application is implemented. Without such a common framework, leveraging does not occur.

The common architecture must not stop at the point of defining hardware, communication protocols, or even data base types. A consistent way of understanding the types of data to be stored and the type of repository for those data should be planned. If a real-time database is to be employed for process...
control and monitoring, for example, what are the valid types of data to be managed by this portion of the plant manufacturing application architecture? What are the valid functions to be addressed by this part of the architecture?

Often, the answers to these questions are blurred by misconceptions. Sometimes companies select a particular data base vendor and perceive they have accomplished leveraging. On the contrary, the applications must be able to plug and play in the target database existing at the sites if they are to be leveragable. This degree of leveraging implies standards for the development of applications that use the selected data base, ideally determined before development and acquisition of applications.

**Data Standards.** Data standards go beyond a textual specification of functionality, however. Textual specifications usually emphasize desired functionality of the application and fall short of defining a model of the data to be employed. Information engineering, which is now rapidly becoming a standard for data base design, is an improvement over narrative specifications. This methodology incorporates data modeling and functional modeling using the entity-relationship diagram (ERD) and activity hierarchy diagram (AHD), respectively.

To prepare for leveraging applications, IS management must answer several questions:

- What are the standards of technology to support applications?
- What are the standard tools for building and maintaining manufacturing applications?
- What are the standards for modeling and describing requirements?
- What types of data will be stored and operated on by these systems?
- What are the standards for screen design and use interaction?

**EXHIBIT 5—Typical Manufacturing Information Processing Requirements**

**TECHNICAL ARCHITECTURE ISSUES**

It is often impractical to know every aspect of each plant’s or site’s technical IS architecture before commencing development. It is important, however, to have a common way of viewing existing and proposed systems in the context of the types of data they are to manage.
A data-centered architectural framework can be used for information and control systems in manufacturing. Exhibits 5, 6 and 7 provide an overview of the framework. Data is viewed in three distinctive categories: “in-area” for process control, “manufacturing operations and control” for plant-wide product control, and “production history” for plant-wide and business-wide decision support. This architecture places such functions as MRP, order management, and inventory management as business level or supply-chain functions. Exhibit 6 decomposes these three categories of data stores into further detail. Exhibit 7 maps a specific example set of data concerning the subject of quality to the architecture of Exhibit 6.

These diagrams portray a topological way of viewing manufacturing data. They also reveal a fundamental obstacle to leveraging. Without some type of agreed-on taxonomy of data, leveraging becomes difficult. An application framework should recognize and position the desired suite of application software during the planning or analysis phase of the project.
Many workable approaches can be applied for modeling manufacturing applications, but both data types and functions must be central to any discussion of modern plant IS architecture. In the end, it is the types of data to be managed that must be understood between applications if cost-effective integration is obtained.

Given optimum conditions of similar or like plant logical and physical architectures, the ability to leverage an application is, nevertheless, still influenced by a variety of factors. First, it is important to note that the closer the application is to the manufacturing process equipment, the less the application will lend itself to leveraging. Process control applications are inherently tied to the site process control systems, which often vary greatly. It is a rare process control application that is totally independent from the field/equipment instrumentation that is measuring and controlling the process.

A valid approach to designing leveragable applications is to place only real-time applications at the process control level of the architecture. These are applications that must, by the nature of the data they manage, be positioned such that they can access the real-time process control database.

In contrast, applications that can be logically positioned farther from the front-end process control database tend to offer greater leveragability. Such applications are usually database applications related to the product rather than the manufacturing process. The integrated operations or manufacturing execution level of type 3 applications in Exhibit 6 (previously referenced) lists some of these products or subject-based applications.

Leveragability, then, is limited to the extent that it is specific to the site’s manufacturing processes. It is therefore incumbent on the applications designer or purchaser that applications be properly positioned within an IS architecture and, further, be constructed to be as generic as practical with respect to the process control system.

<table>
<thead>
<tr>
<th>EXHIBIT 8—Taxonomy of Leveraging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
</tr>
<tr>
<td>Core Capability</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Auxiliary Functions</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Site-Specific Functions</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

MEASURING LEVERAGABILITY

The ultimate measure of leveraging is the resulting business benefit – the reduced cost of delivering a working capability from site-to-site across an enterprise. Can leveraging be quantitatively analyzed before commencing a rollout in order to forecast the required resources/costs per incremental site? The following paragraphs illustrate one approach.

It is helpful to map the various functional elements of the application against an expected quantity of its leveragability. A spreadsheet can be used to view the various components according to three categories:

- Core functionality.
LEVERAGING DEVELOPED SOFTWARE:
AN ECONOMIC PERSPECTIVE

- Auxiliary functions.
- Site-specific functions.

The data types and functions comprising the application set can be decomposed into these three categories. Exhibit 8 depicts such a decomposition.

As the data model underpinning the application is foundational and can be understood independently of user screens and reports, it is likely the purest representation of core capability. Functions of one or more applications can be separated by some rationale into independent sub-functions. Sub-functions can be assigned to one of the three leveraging categories.

Site-specific functions are those that must be customized to a particular site and therefore offer the least leveragability. Auxiliary functions may be items like reports, specialized functions, or perhaps particular user screens. Such functional items may be leveragable across a subset of sites but perhaps not the whole of the target domain. Core functions are those that can be leveraged across every site in the target domain without modification.

Analysis of Target Domain. There are two possible sources for meaningful information on which to base estimates for assessing leveraging. The best information source results from an exhaustive analysis across the target domain. An analysis of the subject areas across multiple sites is able to calibrate the data/functional model according to the extent of changes likely as the application is moved from one site to the other.

Pilots and Prototypes. Another valid source of information is from completed projects, pilots, or prototypes. These assets are excellent, because they provide a test bed for quantifying leveragability. In any case, usually more than two sites must be sampled to have a meaningful representation of the whole.

Weighting Factors. Assuming one of these two possible sources of information, it is possible to assign leveraging weighting factors to each function or process of the application. Weighting factors should reflect, more or less, the extent of variability in each sub-function or capability of the application with respect to target sites. Weighting factors can also be used to assess resource requirements as measured in development resources or costs. This discipline reinforces an effort to maintain as much of the application as practical within what may be called the core, thus driving greater leveragability.

Breakpoint values for leveragability can be defined, quantitatively, by what is in the core, auxiliary, or site-specific categories. One hundred percent leveraging means that no modification is required in the application to make the application operational across all sites in the target domain. Zero percent leveragability implies unique tailoring of the applications to each site. The end state of the leveraging analysis should portray the incremental costs of moving the complete application across all sites in the target domain.

CONCLUSION

Leveraging is driven by management’s recognition of the inherent costs of unique site-specific solutions. An effective business practice will explicitly declare leveraging to be part of the IS manufacturing strategy. The goal of leveraging is to maximize the next present value of the IS investment.

As manufacturers continue to invest in information technology, significant assets are being created in plants. The management of these assets should be performed with the same decision process as applied to other important capital assets.

This article presents a financial perspective on leveraged software. Article 3-01-80, “Leveraging Developed Software: Organizational Implication,” describes an effective environment for leveraging software development and support.
LEVERAGING DEVELOPED SOFTWARE:
AN ECONOMIC PERSPECTIVE

1 R.L. Sloan, Executive Briefing, Maximizing Return on Plant IS Investment (Houston TX: November 17, 1993).
2 Sloan, Executive Briefing.