

# The Net Effects of Product Lines

Scott Tilley



One of the most promising aspects of net-centric computing is its ability to aid the incremental migration from stove-piped legacy systems to product lines. Operational for many years, legacy systems are often viewed as liabilities, in part because of the difficulty involved in understanding their structures and upgrading their capabilities to meet new business requirements. Net-centric computing can play an important role in converting liabilities into assets through a three-step process of decomposing the legacy system into separate functional units, developing these artifacts into components using distributed object technology, and deploying these components as core assets in the product line.

As described elsewhere in this edition of *SEI Interactive*, a product line is a group of products sharing a common, managed set of features that together address a particular market segment or fulfill a particular mission. Product lines offer several advantages over more traditional application-development approaches, not the least of which is the ability to reuse corporate assets across a number of separate but similar products. However, in order to adopt a product-line approach to software engineering, the existing base of legacy systems must be dealt with first.

## Legacy Systems

By definition, legacy systems are resistant to change. Although legacy systems vary in their individual features, they all share several characteristics. They are old (10–25 years) and large (100 KLOC–1 MLOC). Such age and size typically means the system is poorly structured, often due to traumatic maintenance over its lifetime. This contributes to a poor understanding of the system’s essential functionality, a situation that is exacerbated by personnel turnover: each developer is forced to relearn aspects of the system that previous developers spent significant time and effort to recover. Nevertheless, legacy systems represent substantial corporate knowledge and cannot simply be discarded and rebuilt with a “green field” development effort.

Instead of viewing legacy systems as liabilities that are difficult to understand and hence difficult to evolve, a better approach is to turn them into valuable corporate assets. Product lines offer the ability to leverage existing assets to provide a significant return on investment over the long term. This can be accomplished by providing the infrastructure for the strategic reuse of corporate assets, such as

mission-critical systems and associated business rules. This process is illustrated in the following figure.

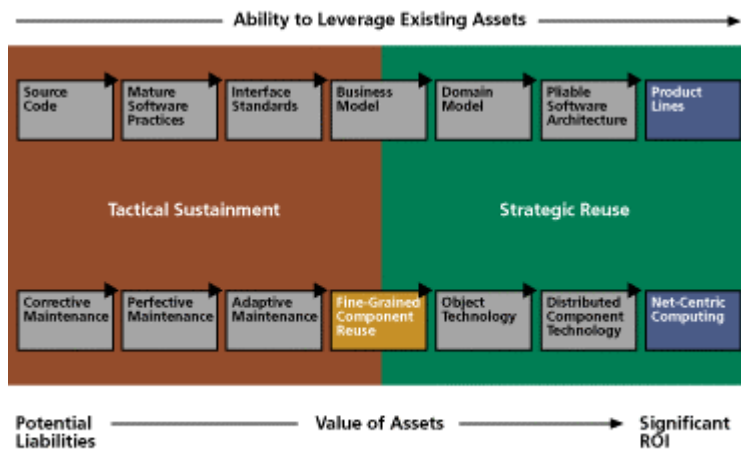


Figure 1: Leveraging Assets

## The Migration Process

Migrating legacy systems to product lines can be accomplished using a three-step process. The first step is to decompose the legacy system into separate functional units. The second step is to develop these extracted artifacts into components using distributed object technology. The third step is to deploy the new components as core assets, making them available to the product line.

The result of the migration process is a three-tiered set of interacting components, as illustrated in Figure 2. Each component may reside on a client or a server, or it may migrate between them (in effect, serving multiple roles simultaneously). Data can also be made similarly mobile. These components can provide the basis for a product line as the software continues to evolve to meet changing business requirements.

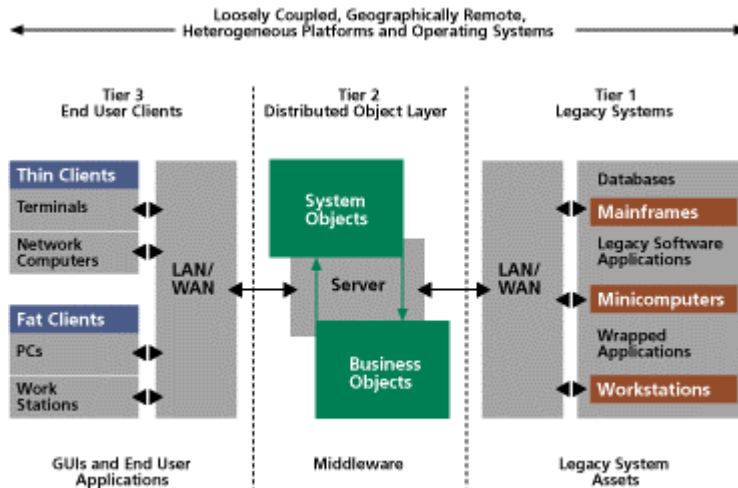


Figure 2: Three-Tiered Architecture

## Decomposing the Legacy System

The first step in the migration process is to decompose the legacy system into separate functional units. This can be accomplished using two variants of system-understanding technology. The first variant is a white-box approach that relies on traditional program understanding. This is typically done using a reverse-engineering tool that parses the legacy system's source code, populates a repository with the data it gathers, and performs analysis on this data to aid the user in understanding the system.

The second variant is a black-box approach that uses newer methods to understand the system. These methods can include binary reverse engineering, interface analysis, and behavioral probing. The black-box approach is used when the source code to the legacy system is not available for white-box analysis.

The result of this procedure is the first tier of the new product-line infrastructure. At this point, the artifacts can be considered to be virtual components. They represent essential functional artifacts of the legacy system, but are not yet in a format suitable for use as a true component in the software engineering sense.

## Developing the Core Assets

The second step in the migration process is to develop the product line's core assets from the artifacts extracted from the legacy system in the first step. This is done by wrapping the virtual components with object interfaces. This results in true components that become available for use in the third step of the migration process.

Wrapping the virtual components can be accomplished using distributed object technology as middleware. There are several different middleware offerings to choose from for this task. One of the most popular is the Object Management Group's Common Object Request Broker Architecture (CORBA). Microsoft is a proponent of the Distributed Component Object Model (DCOM) and related technologies under the COM+ rubric. Sun Microsystems espouses an approach relying on Enterprise Java Beans (EJB), possibly augmented with its Jini networked-appliance enabling technologies.

The result of this second step is the middle tier of the new product-line infrastructure. The new business artifacts are encapsulated with object interfaces, turning them into core assets and making them available to the third tier. The final step is to deploy the assets.

### **Deploying the Core Assets**

The third step in the migration process is to deploy the core assets using the network infrastructure. The core assets are accessed by product-line developers, who in turn use the assets to construct variant product instances. The result of this third step is the final tier of the new product-line infrastructure.

By deploying the core assets across a network, client applications can access and browse the collection of core assets. This facilitates the strategic reuse of business-critical core assets, which has two benefits. The first benefit is the ability to leverage existing capabilities—in other words, to convert legacy liabilities into valuable product-line assets.

The second benefit is the ability to compose new functionality from the newly mined business objects. Because the virtual components that were mined from the legacy system in the first step of the process have been given object-oriented interfaces in the second step of the process, the resultant artifacts can be combined to provide functionality that would have required significant writing of new code in a non-product-line approach. This composition of assets into new functionality facilitates the evolution of the legacy system to meet new requirements in a disciplined and cost-effective manner.

### **The Net Effects of Product Lines**

Today's newly developed system is tomorrow's legacy system. As such, legacy systems will always be with us, and will remain vitally important for the foreseeable future. The key to successfully adopting a product-line approach to software engineering is converting existing legacy systems from liabilities into assets. This can be accomplished using the three-step process described above.

In essence, net-centric computing provides the same application and asset portability to the client that distributed object technology provides to the server. The result is a flexible deployment infrastructure for core assets of the product line. The three-tiered architecture can be used by both end-users of the derived products, and by developers who rely on the core assets to construct instances of the product line.

The net effects of product lines are that they make software engineering more akin to a manufacturing process, with predictable attributes such as cost, schedule, and quality. The key is the strategic reuse of corporate assets—assets that are usually considered liabilities. Product lines promise to pervade software engineering in the new millennium, resulting in the development of better products in shorter time at lower cost. Underlying the success of product lines is the infrastructure provided by net-centric computing for migrating legacy systems from stove-piped systems to deployed core assets.

### **About the author**

Scott Tilley is a visiting scientist with the Software Engineering Institute at Carnegie Mellon University, an assistant professor in the Department of Computer Science at the University of California, Riverside, and principal of S.R. Tilley & Associates, an information technology consulting boutique. He can be reached at [stilley@sei.cmu.edu](mailto:stilley@sei.cmu.edu).

---

Copyright © 1999 by Carnegie Mellon University

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

<sup>SM</sup> Architecture Tradeoff Analysis Method, ATAM, CMM Integration, CMMI, IDEAL, Interim Profile, Personal Software Process, PSP, SCE, Simplex, Team Software Process, and TSP are service marks of Carnegie Mellon University.

® Capability Maturity Model, Capability Maturity Modeling, CERT, CERT Coordination Center, and CMM are registered in the U.S. Patent and Trademark Office.