

Progressive Renovation

A White Paper by Bruce Hutcheon
Co-Founder and President, Dovetail

Bruce Hutcheon, President and Co-founder of Dovetail and a world-class payments system authority, explains how to replace the old with the new with the least disruption.

The upheaval of replacing an existing back office infrastructure can be daunting and fraught with risk. This paper discusses an approach to the task we call 'Progressive Renovation.' Our approach is appropriate for almost any complex back office environment, but this discussion will focus on the goal of renovating the payments operation typically found in a large international bank.

The Situation

Most payments systems in major banks started life over 20 years ago and were originally built to meet the business model of the time. While they have been greatly modified and extended, there comes a point where the cost of keeping them in sync with changing business opportunities and regulatory requirements becomes prohibitive.

A typical bank finds itself with a diverse, complex payments operation whose architecture has evolved over several decades. The mini-epochs of this evolutionary history probably include:

- Telex wireroom automation
- Introducing direct connections to SWIFT and other clearing and settlement systems
- Hardware migrations between mini, mainframe and various PC and client-server platforms
- Acquisition and merger mania
- Changing standards
- Demands of compliance and regulation
- Various technology paradigm shifts such as batch versus real-time, client-server, middleware and service-oriented architecture
- The bank's own history of product offerings, operations management regimes and renovation attempts

The result is often a dense thicket of tangled technology and convoluted procedures.

Payments back office operations have a particular problem in that many of their systems run on Vax or Tandem hardware no longer supported by their vendors. The application software is typically written in combinations of COBOL and proprietary languages unsuitable for today's network environments and no longer familiar to programmers. There is no simple upgrade path for these systems.

What needs to be done?

There is often a strong desire to retire the old systems and rationalize—meaning 'to simplify', not 'make excuses for'—the whole back office. At the same time, there can be an

equally strong reluctance to take on what might seem to be a Herculean task with the related cost, risks, and operational upheaval.

Approaches can range from 'rehosting'—keeping the existing software while upgrading the hardware and network environment—to wholesale replacement of several interconnected legacy systems including hardware, software, and entire business processes. The appropriate approach usually depends on whether the existing system serves today's business model appropriately and has the technical underpinnings needed for ongoing modification, enhancements, and support.

For a large bank with substantial payment flows to and from many customers, developing or buying a complete new system and implementing it in a single step is too risky to contemplate.

What is "Progressive Renovation"?

Progressive Renovation installs a new system alongside legacy systems, allowing processing functions to migrate gradually from the old systems to the new. During renovation, new functionality is only implemented on the new system. Once the renovation is complete, the legacy systems can be decommissioned or reconfigured into a more manageable services layer. After the renovation, the new system becomes the payments system—the platform for all future operations.

Progressive Renovation's primary goal is to reduce the risks and operational upheaval of a 'big bang' cutover to a wholly new system. It also allows banks to provide new products and services on the new platform while leveraging the functions of legacy systems. Furthermore, renovation can proceed at whatever pace is appropriate. While there may or may not be cost advantages to a Progressive Renovation project, it is usually possible to spread out investment costs to overlap with an earlier payback period.

Business opportunities and requirements often change faster than a bank's technology can respond. A progressive approach allows for adjustments along the way, making a bank more responsive to such opportunities.

What is Needed?

For Progressive Renovation to work properly, there are several important prerequisites.

The Vision

There must be a vision and an owner of that vision.

The vision is a clear picture of how the operation and architecture should be—flexible, elegant, open-ended, and able to be enhanced and maintained over the next several generations of change and innovation.

The owner of the vision is the person within the bank who 'sees the big picture,' can keep the goal in sight, and is the arbiter of which legacy functions get carried forward and which

redundant functions are left behind. Obviously, the owner must have the authority and resources to lead the project.

A vision differs from a strategy. The vision is more of a 'picture on the wall'; something you aim at and can change as the surrounding circumstances change. Strategy is the plan for effecting the vision. A strategy must be reviewed periodically to confirm it is still in sync with the vision. There is no worse recipe for effecting change than setting rigid end targets that may be inappropriate or inadequate by the time they are delivered.

Next in importance is the architect, whose must interpret and temper the vision into a form that can be implemented, managed, and maintained. The architect may be someone with the bank's IT department or an outside consultant or vendor.

The Value of Legacy

'Legacy' has taken on a negative connotation of late. Ultimately, however, 'legacy' is what has best served its purpose and survived the evolutionary process. It may be convoluted and inelegant, but it has also proved effective. It is easy to forget how much didn't survive—outmoded or failed products, initiatives, and projects, leaving only battle stories and scars. 'Heritage' may be a better term to describe these parts of the legacy which should be preserved and recast in the replacement system.

It is possible—temporarily—to harness useful heritage functionality within legacy systems through service wrappers. Indeed, this is a significant tenet of the modern Services Oriented Architecture (SOA). Over the long haul, however, such functions should be cleansed and rebuilt as components of the new architecture.

The Platform and Tools

A successful Progressive Renovation process needs an appropriate platform and various development and integration tools. Many middleware vendors promote application integration tools, rules engines, and such as 'application platforms.' Middleware is like plumbing. While necessary, it is not a platform and will not replace the foundation and other major subsystems which today should include a database, an application server, a transaction manager, and a payments-specific application framework.

The payments framework used to renovate a payments system must be specifically architected as a payments system. It must model the entities used in the payments business: SWIFT messages, payment transactions, payment charges to a customer, FX trades for cross-currency payments, and so on.

It must have functional components built around the processes used in a payment operation: payment entry, payment repair, payment qualification, check processing, reconciliation, payment scheduling, liquidity management, and so on.

Each of these processes and structures must be independent of the others—in old-speak, it must be truly modular—so that the bank can keep departmental processes and systems which work well, replace old functions as appropriate, and introduce new functions needed to meet today's business requirements.

Above all, it must be built on a technical platform which unifies the elements and lets heritage functions coexist and operate with new components in a single new system. This is particularly important in the changing world of a horizontal banking architecture, which dictates one customer database, one investigations system, one reporting function, or one pricing and billing system with which any new payments system must interact. A unifying technical platform is the key element which allows Progressive Renovation.

After renovation, the platform must be an appropriate foundation for the replacement system. Fortunately, the set of attributes needed to support the renovation role also makes the platform best suited to support the replacement role.

Today's software vendor must anticipate market needs and provide a platform that permits a progressive approach. Payment application products, whose origins stretch back over ten or fifteen years, may be functionally rich but will be fundamentally unsuitable for anything other than a 'big bang' cutover approach.

Project Methodology

A Progressive Renovation project will include many different aspects: requirements gathering, reverse engineering, documentation, programming and integration, schedule and resource tracking, testing, deployment, and source control. Juggling these while maintaining 'normal' operations can get very complicated. All standard project management procedures must be followed rigorously. The value of proper design and planning cannot be overstated. This must be a joint effort between the owner, architect, and project manager.

Adopting consistent integration methodologies is also very important to the process. For example, industry standards such as MQSeries and XML for inter-system connectivity have proven very beneficial. Other third-party middleware tools can also be useful, especially when expertise already exists at the bank.

Different Approaches

There is no one correct Progressive Renovation approach. There are, however, several techniques which have proven viable and repeatable. One is the identification and regrouping of functions and services into new components which, as they are implemented, allow the decommissioning of related functions in the legacy environment. Done properly, this usually results in a very clean and manageable system. However, the related reworking of both the old and new systems can result in a large number of intermediate phases, and all parties should be diligent not to let the transition become a mere reinstatement of the old ways of doing things.

Another valid approach is to migrate business flows by user groups such as workgroups, branches, customer and product groups, and so on. This can work well when the required end-to-end functionality exists at the beginning, which is often not the case.

Difficulties and Possible Pitfalls

There can, of course, be disadvantages to a Progressive Renovation approach. Anyone who has tried to live in a house while it is being rebuilt around them is well aware of some of the potential issues. Beyond the obvious, there are costs and complications inherent in the approach's required reverse engineering, legacy system modification, and temporary bridging of various old and new systems.

One of the biggest potential problems involves the tendency to carry forward too much of the old system, thus 'recreating the legacy.'

Because of its multiphase rollout, a renovation project can also take longer. While the goal of completely replacing the legacy systems is real and attainable, there is in reality no 'end state'; Progressive Renovation is an ongoing process, even a way of life.

Many banks use 'SWIFT-like' messages as an internal standard for passing data between components. This is a bad idea. These messages were not designed for this purpose and are not easily extensible to carry the data necessary for payments processing. For example, multi-stage processing requires various inter-process codes and other data, as well as a complete audit trail. Where does this go in a SWIFT message? By what standard? Typically, all kinds of garbage gets put in the header and in field 72. Different systems invariably have their own non-standards, which require special processing and cleanup. This is rarely documented properly, is difficult to reverse engineer, is sometimes mutually incompatible between systems, and often presents unexpected problems during integration testing. XML is a much more effective inter-system message standard.

That said, the passing of XML blobs as the basis of an STP architecture is a bad idea. The payment data should be extracted from the messages and stored in a central database. The subsequent processing should be performed on the records in the database under the control of a workflow or transaction manager.

Summary

Progressive Renovation is not so much a means to an end as a way of life. Even when the original goals of replacing old systems with new and elegant operations are met, new business opportunities and other forces will continue to drive the need for change. The Progressive Renovation process must be integrated with ongoing enhancements and maintenance.

A full Progressive Renovation undertaking can be a long and winding road. Many tempting shortcuts will appear. Such a shortcut's expediency must be judged within the methodology framework. Is this part of a temporary bridge or something that will live on as a component of the long term application? Remember all the maintenance issues; are you building a 'new legacy'? Throw-away code has a habit of outliving expectations and replicating itself in other similar circumstances. Progressive Renovation's vision and methodologies should be maintained for the life of the system.