

A Transformation Approach to Smarter Core Banking



Redguides
for Business Leaders

Alex Louwe Kooijmans
Rishi Balaji
Yasodhar Patnaik
Saket Sinha



- Why transformation?
- Transformation methodology, framework and tools
- Core banking systems infrastructure



Executive overview

The way today's economies are functioning, plus growing competitiveness between banks, demands a very flexible IT environment. Banks must be able to quickly comply with new regulations, introduce and bundle products, enter new marketplaces, and make swift business decisions to increase profitability and revenue. An agile IT environment that can easily and continuously be adapted to changing business needs is key.

However, many core banking systems have been modified, extended, replaced and customized over time, resulting in a vast and complex web of customized code, especially when this is done without strong and structured enterprise-wide governance. Maintaining this code can involve significant operational cost and risk. Core banking systems were initially designed to be product-based and hinged on loans, accounts, and savings. Applications and processes followed this model, driving the duplication of services, for example opening an account, across multiple product silos and customer touch points. This also led to a product-based approach to systems and governance, in which each operational silo made its own decisions without considering the requirements of other business units.

Currently, changing business models, increased regulatory requirements, and the need for better risk management are forcing banks to integrate systems across siloed product lines, further increasing the size and complexity of legacy systems. Moreover, transaction volume and related data have exploded during the last decade and have exponentially compounded complexity by causing a significant overhead in bank operating costs. This trend is expected to continue in an accelerated manner because of the growth of the mobile channel.

The inflexibility of core banking systems in adapting to new approaches is becoming a major business issue as business and operating models focus more closely on customers and markets. Innovation, by means of exploiting new architectures and technologies, is necessary to keep the IT systems responsive to the business needs of today. In most cases a transformation will be required to create a foundation that is sustainable and flexible enough to accommodate today's and future business developments. This IBM® Redguide™ publication explains how banks can use the IBM Banking Industry Framework along with industry models and software and hardware products and solutions to progressively transform and modernize their banking systems and achieve better straight-through processing, synergy, and integration between product-based systems—and better time to market, while staying in check with costs and minimize overall transformation risks to business.



The case for core banking transformation

For many banks, the critical complexity gaps that hinder profitable development are embodied in their core banking systems, application architectures, and project development capabilities. While core banking systems are not actually broken, they are becoming a major impediment to effectively executing application and product development. This need for a more flexible core banking environment is accelerating because of some major trends in the banking industry that are taking place right now. We examine these in the next section.

Trends in the banking industry affecting core banking systems

The banking industry has been evolving and searching for new business models since the very beginning, but never before have the needs and trends been as strong as today.. The following are the three most significant developments that call for a deep examination of the core banking systems environment.

Customer insight and pro-active relationship management

Increasing insight into a bank's customer's buying behaviour and needs is putting the bank in a position to offer products at the right time and in the right place, and perform cross-selling and bundling of products. This could even extend beyond the traditional bank products, for example into insurance products. For example, a customer who just took out a car loan may also be interested in an insurance product for the car just purchased. Many of the bank's transactions touch the core systems at some point, whether it is just an inquiry or finalizing the actual sale of a product. Opening up the core systems and making them more flexible will allow the bank to access core systems more easily from across the entire bank, allowing for product bundling and cross-selling.

Risk management and compliance with regulations

In today's economic climate there is a lot of focus on the banks and their health. A bank must be capable of determining its health at any point in time. Also, a bank has to comply with numerous regulatory requirements, many times different per country, and also changing through time. A bank today needs flexibility in creating reports and calculating indicators. The

core systems contain many of the source data for these reports and indicators and should be up-to-date and easy to access at all times.

Driving down IT cost, while gaining flexibility

For many banks, core systems consume a significant portion of the IT budget and out of this portion a large part is spent on “keeping the lights on”. So, there is money spent on IT without really adding business function. With a continuous pressure to lower cost, including in IT, this is an issue. It would be much better to have an almost “maintenance-free” core banking environment, where money is only spent on actual new business functionality. With the adoption of modern service-oriented architecture (SOA)-style application architectures and business process management (BPM), the ratio between money spent on maintenance and money spent on new business functions will improve dramatically. By applying more straight-through processing, banking processes become less labor-intensive as well.

Key growth imperatives for banks

The inflexibility of legacy core banking systems is making it increasingly difficult for banks to meet challenges and to address key growth imperatives, including the following:

- ▶ Address market opportunities in a timely manner
- ▶ Increase profitable customer acquisitions
- ▶ Tap high-margin new markets
- ▶ Improve the ability to target investments
- ▶ Drastically reduce the per unit cost structure and minimize operating overheads
- ▶ Create sustainable differentiation
- ▶ Increase volume without compromising on risk
- ▶ Regain consumer trust

The business problem

Banks continue to undertake extensive ad-hoc integration efforts to get a total customer view across siloed business lines. Given financial and time pressures, many banks have chosen to take short-cuts and apply quick fixes rather than to address the underlying product-based architectural and governance issues.

Given the sheer size and scope of the challenge, using short-cut options has so far been more appealing than tackling a full modernization project. Typically, a large core banking system can contain millions of lines of code across multiple, distinct applications, written in many versions in various programming languages, some of which may have become obsolete. Such a system will depend on different database technologies and will communicate, both internally and externally, with other systems via many unique interfaces.

Figure 1 illustrates the key concerns banks have with their current core banking IT systems.

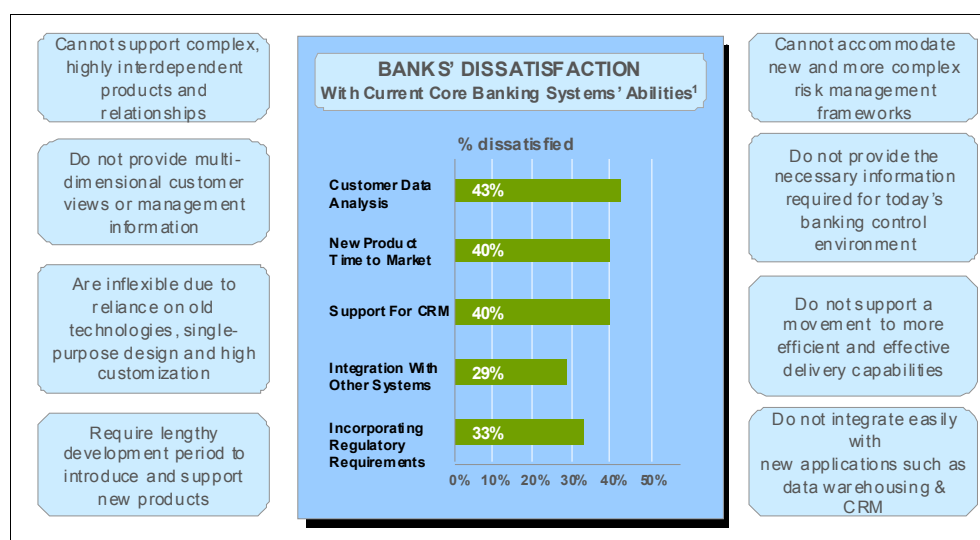


Figure 1 Dissatisfaction with current core banking IT systems as perceived by banks

The key issue is the underlying architecture of the core systems. Legacy core banking systems were developed on a product-centric design that delivered operational support for a single product line. At the same time, contemporary design principles allowed applications to hard code user interfaces, business logic, business rules and data, all in millions of lines of code and in monolithic programs. Without any holistic architectural strategy, thousands of tactical changes carried out over time have left banks with inflexible and enormously complex legacy systems. Most of these tactical changes spawned a series of code and program changes that cascaded across the length and breadth of the application portfolio. Failure to adopt modular design principles, to properly document code changes, and the limited availability of systems skills and knowledge all added to the overall complexity and further weakened the bank's ability to thrive in today's competitive banking environment.

The result is that the legacy core banking system at many banks has become a tangle of hundreds of point-to-point interfaces, dead or unreferenced code pools, poor documentation, duplicated or obsolete functionality, and inconsistent, redundant and duplicate sets of data. All this spawns proliferations of applications, programs, codes and data causing compounding impact on infrastructure, network, and computing power.

This complexity makes it extremely time-consuming and difficult to make modifications, add new functions, and repair the existing systems. Maintaining the existing systems in good working order consumes a major portion of IT budgets and also causes time-to-market issues. Consider that a typical new core system enhancement consumes a huge part of a bank's IT budget merely by demanding integration and testing across the jumbled mess of infrastructure and application platforms.

This complexity is true both for banks that built core banking systems and for banks that customized a software package, though more and more customization was required to accommodate explosive growth and changing business models. Many existing software packages were designed around product-centric architecture; in time, they suffered from the same design challenges as custom-built solutions. Thus, the levels of complexity and inflexibility that inhibit growth and differentiation are similar regardless of whether the legacy systems were custom built or adapted. Additionally, front-end systems with tight dependencies on the underlying core systems created another layer of cost, inflexibility, and complexity.

Current business expenditures

Banks typically spend between 10 and 20 percent of their total operating budget on IT. More than half of this amount is spent on maintaining aging core banking systems, and of that, about 90 percent is spent on maintaining day-to-day business. This leaves limited resources for discretionary projects or innovations. In fact, legacy IT systems can account for as much as 30 to 40 percent of the bank's total operating expenses as shown by the need to maintain larger numbers of employees due to system inefficiencies and poor front-to-back-office integration or limited straight-through processing. Figure 2 illustrates such spending.

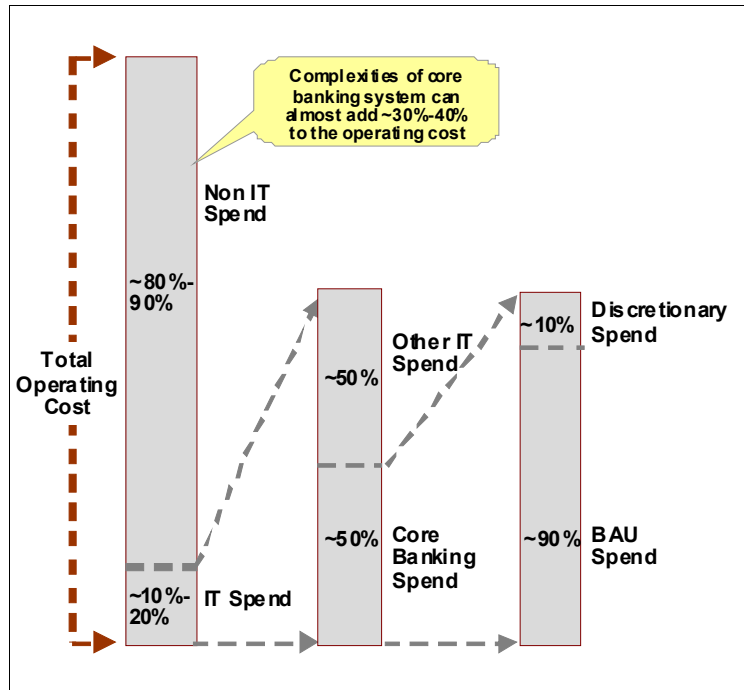


Figure 2 Core banking IT spending analysis

Unmodernized legacy systems can reduce revenues in various ways, including the following:

- ▶ Lower customer retention
- ▶ Lower success rates in new opportunities in limited cross-sell capabilities
- ▶ Lack of customer centricity
- ▶ Increase time-to-market for innovative products

Another cost is related to the changes required to accommodate regulatory and risk management guidelines. This cost is exponentially compounded by the complexity of the legacy systems and the need to gather data across numerous siloed applications.



Core banking transformation process

Addressing core banking systems' complexity issues is a top priority, and many banks have started to modernize core systems already or at least started to take a look at it. But this modernization is seldom done at systemic levels and therefore does not deliver optimal results. Modernization must involve a complete overhaul of the core application portfolio, IT architecture, and infrastructure. Such an effort is expensive, time consuming, and risky but, if successfully done, may significantly benefit the shareholders and also provide a sustainable competitive differentiation in the marketplace. However, since benefits far outweigh risks, some banks have already begun, or plan to begin a long-term systemic modernization process. Their approach is to achieve a high degree of architectural modularity and speed-to-market benefits while reducing overall complexity and unlocking non-discretionary funding.

IBM is a leader in the modernization effort and offers deep industry experience amassed by helping banks worldwide to successfully modernize core systems. The IBM experience shows that a successful modernization project includes the following elements:

- ▶ Institutes transformational governance.
- ▶ Aligns business and IT stakeholders to commit to successful modernization.
- ▶ Includes key design principles of modular business and IT architecture.
- ▶ Selects proven methodology and technology.
- ▶ Optimizes infrastructure to deliver maximized value of the modernized systems.
- ▶ Professionally manages the project.
- ▶ Bases the delivery capabilities on an industrialized factory model.

We discuss how these elements drive a successful modernization in the following sections.

Getting started with the transformation

Successful organizations follow a structured approach to begin the transformation process. A structured approach includes the following key principles:

1. Establish transformation goals and objectives.
2. Align the business and IT stakeholders.
3. Set up organizational governance.

4. Define the “to be” architecture.
5. Create the business case.
6. Design the transformation roadmap.

These topics are discussed in the following sections.

Establish transformation goals and objectives

A core banking transformation project must have the goal to eliminate inhibitors of growth to an acceptable extent. Now, of course, the issue is that there are many different ways to transform and one way may reach the goal better than the other. So you need to define what “acceptable” means and you need to be able to measure the improvements that the transformation project brought. Examples of easily quantifiable measurements are:

- Cost to develop a new function

For example, upgrading from a traditional programming model to a Java-based programming model will reduce this cost significantly. This means that with the same IT budget more functions can be added, leading to more business value.

- Time it takes to deploy a new function

For example, by using a better application architecture, better management of IT assets, and end-to-end application lifecycle tooling, deployment speed will be much improved. Because of this, new business functions become available faster, helping to remain competitive in the marketplace.

However, some improvements may not be so easy to measure. A transformation project may lead to an overall better quality of the IT system and thus impose less risk.

Align the business and IT stakeholders

It is critical that the objectives of both business and IT are aligned and that both participate equally throughout the modernization process. Common business goals are growth, profit margins, new products and services, and improved customer centricity. Common IT goals include lowering nondiscretionary cost, rationalizing portfolios, modernizing systems, improving efficiency, resolving complexity, shortening time to market, and enhancing skills. A modernization effort will not be successful if it is driven from a single perspective, that is, IT or business.

Business and IT need to create a common forum where business priorities and objectives, and corresponding IT options—cost, time and risks—are evaluated, discussed and agreed upon. Even though management may be aligned on common business and IT objectives, these common goals do not necessarily percolate through the organization unless management has committed to communicating and enforcing them. One management technique is to incorporate modernization objectives into individual performance reviews.

Both IT and business teams must balance the interdependencies of agility, flexibility and time-to-market against system complexity, development process, and technology architecture. These interdependencies must be communicated through a common vocabulary that will be adopted across the board. Both teams must conduct an ongoing dialogue during the entire modernization process. Successful banks carry on such dialogues using vocabulary from banking industry reference models. All these efforts result in a better understanding of business requirements, and save time and reduce costs, especially during the downstream testing and integration processes.

The business and IT areas need to jointly define and agree on transforming functional, application, architecture and infrastructure levels. Further, they need to express this agreement in quantifiable business benefits that have both near term and long term tactical, pragmatic and strategic realizations in line with the business case. To achieve a successful core banking transformation, banks need to define an enforceable mechanism of periodic review and common governance. This will align the benefits and the timing of realizing those benefits as common goals directly tied to the financial metrics of the bank.

Set up organizational governance

Transformation governance is the decision-making process involved in modernizing a core banking system. Typically, transformation governance shares the rights and responsibilities of governance with the bank individuals participating in the modernization, especially management and stakeholders. Transformation governance entails leadership buy-in, commitment, collective ownership, and accountability in order to mitigate risks and deliver business benefits. In many cases, upper management mandates the need for modernization and communicates that goal throughout the organization.

Management must be completely committed to the modernization process to ensure success and prevent failures; further, management needs to be continually aware of potential pitfalls throughout the process. For example, the cost of transforming an enterprise is often beyond an IT budget, which means the bank must identify additional capital to cover those costs. Because the modernization investment horizon is long-term and involves significant risks, the bank must have a clear roadmap to avoid unpredictable outcomes. If the methodology chosen for the transformation is not correct, the process can stall and create critical gaps, further adding to overall costs. Management needs to focus on the process to make sure that unforeseen challenges do not force the bank to abandon the transformation midway. Successful banks have addressed transformational governance in some of the following ways:

- ▶ Collectively agreeing and committing to the transformation roadmap at the onset of the modernization process.
- ▶ Clearly understanding the business case.
- ▶ Agreeing and enforcing the core tenets of the transformation process.
- ▶ Supporting the modernization case with statistics on potential savings, improved efficiency, and so on.
- ▶ Quantifying and demonstrating direct and indirect benefits.
- ▶ Communicating how the transformation will improve operations while reducing costs.
- ▶ Analyzing operating costs against component-based business architecture to identify points where operating costs are not aligned with strategic imperatives.
- ▶ Translating accrued benefits and cost savings to the bank's balance sheet to demonstrate the modernization value to shareholders.
- ▶ Expressing benefit statements as a direct impact on the cost-to-income ratio, as an increase in operating leverage, as an increase in profit margins, and as a direct impact on operating costs.

Another key requirement of transformational governance is to identify and assign modernization roles and responsibilities in the bank and, further, to set accountabilities for those roles. Management needs to be empowered not only to make key decisions but also to enforce responsibilities and accountabilities throughout the entire process. The bank achieves optimal results only when the entire organization is aligned with the modernization team and committed to the same goals.

Lastly, transformation governance must interact seamlessly with the day-to-day business of the bank while simultaneously monitoring the modernization process. Often banks create forums and steering committees to enforce the decisions necessary to drive the transformation process; banks can also seek support and guidance from external advisory services.

Define the project organization for the transformation approach

Obtaining organizational commitment from the top is a vital first step in the transformation. Successful banks have the support of their board, C-level managers and business unit leaders, all of whom are directly responsible for driving the transformation and achieving business objectives. The entire team begins by clearly defining goals, developing the roadmap and a pragmatic business case to achieve those goals.

Figure 3 shows an example of an organization structure for a transformation project.

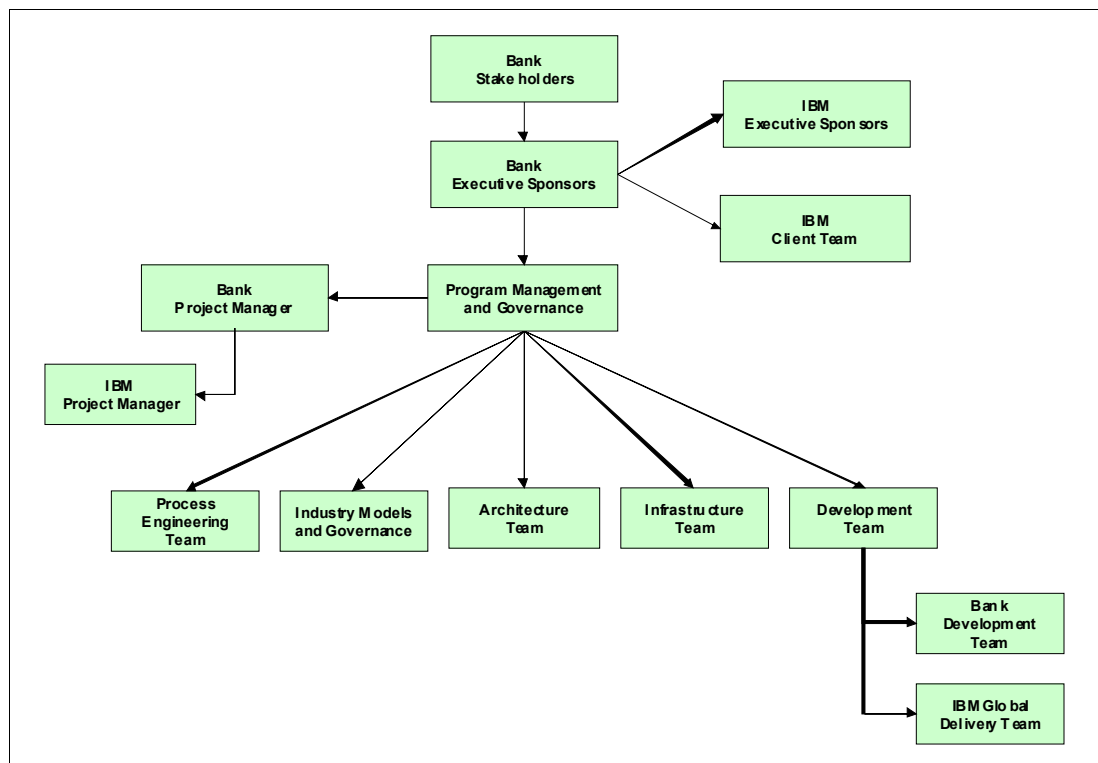


Figure 3 Example of an organization structure for a transformation project

This team must also consider the following:

- ▶ Governance, the change-control framework, and communications plans
- ▶ High-level details of business and technology initiatives, with prioritization
- ▶ Investment justification, benefits, cost and risk factors for the entire program
- ▶ A roadmap that takes into account project strategy, dependencies and limitations
- ▶ Thorough understanding of how these changes will affect the bank

Define the “to be” architecture

Conduct an envisioning phase with stake holders to level set the expectations and achievements that need to be targeted in the transformation. The envisioning phase begins

by focusing on the strategic imperatives that the bank has to deliver and the current pain points that challenge that delivery. Use this input to create the strategic priorities.

Develop the *to be* architecture that will be used with the strategic priorities to assess current legacy assets, to identify gaps and the initiatives needed to fill those gaps. These initiatives can fall into technology, application, functional and non-functional requirements, and governance capabilities. How these initiatives are sequenced and plotted on the roadmap depends on business priorities and inherent dependencies.

Create the business case

Prepare a business case that estimates the overall cost of the transformation plus derived benefits. The merits of the business case may depend not only on the effort itself but can also influence a broader transformation scope that addresses business requirements. It should be more than a technology transformation, but one that also influences a broader business transformation. This is necessary because, if the business case is viewed from a pure technology aspect, it can be difficult to justify costs. Once the envisioning phase is finished, the bank has several options to continue with the process: follow the roadmap, adjust the process to fit the business direction, or select a replacement based on the envisioning phase deliverables. Regardless of the chosen option, the bank can leverage all of its investment to this point.

Design the transformation roadmap

Once the business case is made, the transformation roadmap can be created, which basically describes how and in which steps the transformation will take place. In a transformation process you typically go through the following phases, and we discuss each of these in more detail now:

1. Componentize business and IT architecture
2. Map components
3. Delink legacy systems

Componentize business and IT architecture

A goal of componentization is to separate architectural concerns and modularization. This makes it easier to develop, enhance and troubleshoot the system efficiently and more quickly.

Componentized applications allow users to install only those software components that are necessary for their specific systems. These components can be reused by all appropriate departments of the business. IBM has led the way in moving away from monolithic, product-oriented development to creating componentized software that is more responsive to customer demands.

During componentization, banks re-engineer and redesign their business architecture, separating it into discrete building blocks that are then mapped to the underlying technology services that support them. This allows a bank to view its business as a set of discreet functional components, each exhibiting unique capabilities. Components can then be classified under larger components such as strategy, planning, measure and control, and execution. Using this method, banks create the ability to resolve problems in isolation and to deliver opportunities in targeted areas. In short, componentization enables banks to make required changes without the cost and risk of changing the entire environment. Further, it also improves the ability to use a fine-grained mechanism to better address customer and operational needs such as targeted product bundling, dynamic pricing, offers management, and risk management. Figure 4 displays a typical componentized architecture in a bank.

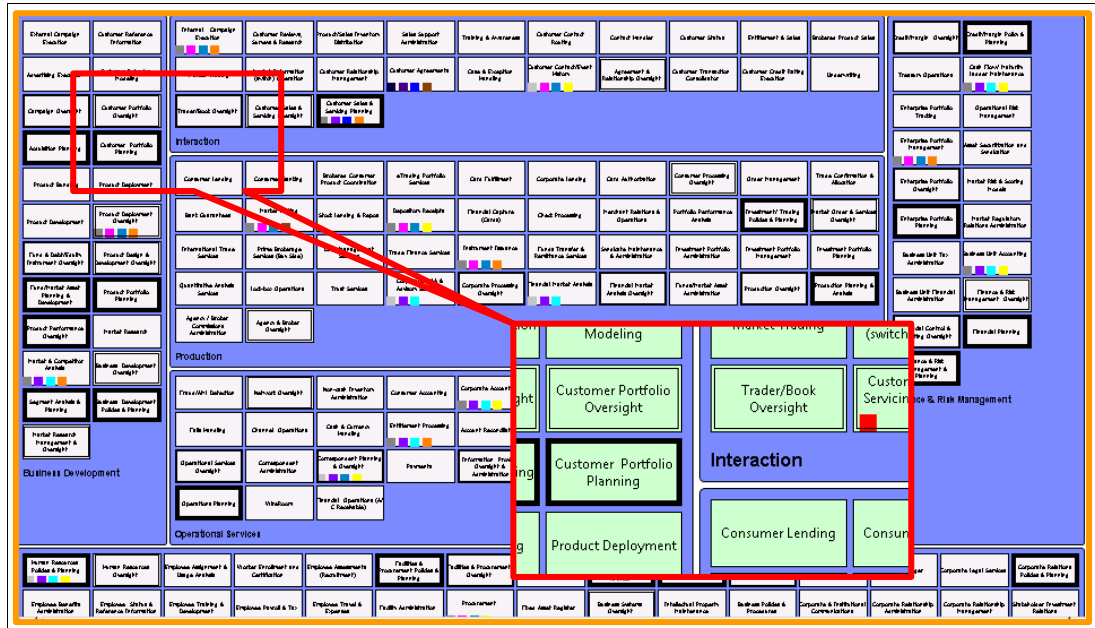


Figure 4 Typical componentized architecture in a bank

Componentization typically involves the following key stages:

- ▶ Establishing a resilient, agile architecture that separates architectural concerns from business logic
- ▶ Using a master data concept
- ▶ Breaking monolithic applications into manageable blocks
- ▶ Extracting business rules and logic for easier access, reference and control

Architecture componentized in this way can be reused by multiple business lines while providing discrete business services that can be orchestrated to satisfy new business requirements.

Basic components of bank operations are as follows:

- ▶ Collecting insights for customers, markets and products
- ▶ Distributing and servicing products through channels
- ▶ Manufacturing products
- ▶ Supporting bank operations with operational services
- ▶ Managing financials and risks
- ▶ Maintaining infrastructure and services necessary to operate the bank

Map components

Business architecture can be depicted as groupings of discrete business components on a component map, a powerful tool that facilitates constructive and focused dialogue between business and IT. The bank's component map allows a business to attribute its strategic imperatives and to identify points to differentiate itself for a sustainable competitive advantage. The map also highlights capabilities and gaps that must be addressed to deliver strategic outcomes. For example, business can use the map to determine areas of high cost consumption and inefficiencies such as current operating expenses, resource consumption, and labor, as well as revenue generating areas.

IT can use the component map to trace the existing application portfolio with capability descriptions, a process that identifies redundancies and gaps in the portfolio. Business and IT

units can work together to identify areas of concern by comparing and contrasting the business overlay of strategic imperatives, resource consumption, and the IT overlay of the technology portfolio. This joint effort drives bank growth while reducing costs.

Delink legacy systems

The transformation efforts that result from business component analysis are most effective when paired with componentized technology architecture. Typically legacy systems do not have well-defined componentized architecture and operate from hard coded elements. Functions such as business process orchestration, functional capabilities, business rules, business logic, and data access are hard coded, which makes them difficult to isolate and separate. Part of the modernization process is to modularize architectural concerns so that the complexity in the related technology can be broken apart. Next the legacy applications need to be re-architected into a new architectural paradigm. The result is a transformed legacy system whose core systems are delinked, meaning that a change in one component does not necessitate changes in other components; see Figure 5.

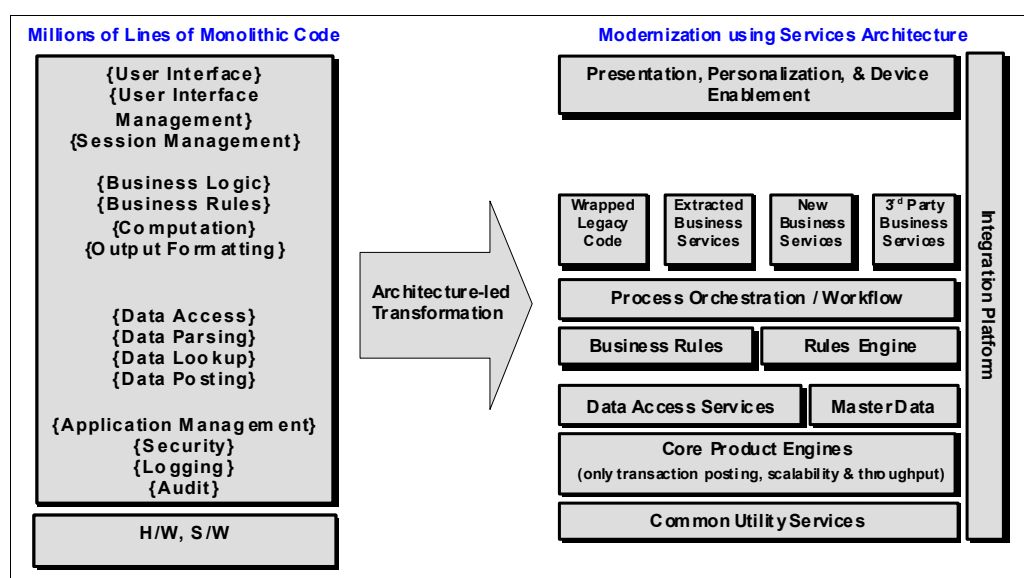


Figure 5 Delinking legacy systems during transformation

Delinking components eases the modernization process because the resulting modulated architecture means that changes are both less time consuming (there are no cascading changes) and easier to manage (less time is required on system integration and testing). Thus, changing a business rule that manages product terms and conditions can be done without requiring the cascading of changes to other parts of the system. In addition, common functional capabilities can be leveraged and shared without duplication. For example, opening an account is a single discreet function that is defined, built, tested and deployed once, then reused multiple times across the organization. Many system capabilities such as security, auditing, and logging are now available in modular software products that can easily be shared rather than replicated across multiple programs.

Another transformation advantage is that legacy applications can be delinked from data, thus exposing the related business logic and making it available to multiple programs. This is accomplished by removing hard coding between application logic and the data. In addition, data can be grouped into master data that can serve read, write and update requests from multiple programs.

Banks have been introducing the following elements, among many others, to separate architectural concerns:

- ▶ Presentation layer: user interface rendered over multiple channels and separated from the business and data access logic
- ▶ Business logic as exposed services shared by multiple programs
- ▶ Rules engine and repository to house common rules
- ▶ Shared common programs, for example fees and charges
- ▶ Master data for customers, products, and contracts
- ▶ Common services to manage complex events and processing
- ▶ Banking process orchestration engine that handles both simple and complex events processing

Separating architectural concerns is an important step because multiple point-to-point integration, either internally across different components or externally across multiple systems, makes system enhancements more difficult and costly. Point-to-point integration tightly couples computing logic at two ends of the system. Multiple point-to-point interfaces create a cascading dependency on the soundness of the system integrity whenever an interface is modified. This increases both speed-to-market and delivery costs.

Further benefits are realized when each architectural construct is exposed as a service, a process that eases making changes and adding new capabilities. Using the principles of Service Oriented Architecture (SOA) enhances the benefits of separating architectural concerns because as a bank transforms legacy systems into SOA, it reduces change impact while increasing development speed. This in turn affects speed-to-market and system development costs.

Banks typically use SOA where core banking components, for example user interfaces, business rules, business logic and data, are developed or exposed as single or multiple service components. Recent SOA advancements allow developers to link different system components through an integration or mediation layer. System components created under SOA have defined input and output interfaces that link with other components via the integration layer rather than through hard coded interfaces. System components, therefore, communicate with each other by publishing and consuming service requests mediated by the integration layer.

Components designed with SOA principles can be changed and still continue to function as long as they are able to publish and consume service requests. User interfaces, data, business rules, business logic, security, audit, and transaction processing should be separate system units that collectively deliver a business function. The key objective is to reduce architectural complexity and minimize the size of system components so that they can be modified, upgraded, enhanced and customized without affecting the remaining components. The banks most successful in their transformation efforts are those that have focused on creating architectural constructs in the beginning of the process.

Transformation approaches

Banks can select from various approaches once they have decided to embark on transforming their core banking systems. Some banks prefer to focus on a few architectural issues and then retrofit the existing systems; others prefer to complete limited transformation of a few systems, while still others prefer to undertake a total systems replacement. The following transformation approaches can be used:

- ▶ A packaged solution that replaces all, or parts of, the legacy systems
- ▶ A rewrite in which all, or parts of, the legacy systems are rewritten from scratch

- A hybrid approach, which is a combination of the packaged solution and the rewrite approach
- A customized progressive approach, in which existing applications are transformed and enhanced in an iterative manner

Deciding on the approach is a complex thought process and the IBM core banking transformation solution can be customized to accommodate any of the approaches. IBM strongly advises to weigh all possible decision factors, with a strong focus on architecture. This is why IBM is also using the term *architecture-led* transformation. It is the architectural foundation that determines agility and sustainability of the core banking systems. Figure 6 illustrates the factors that play a role in this decision process.

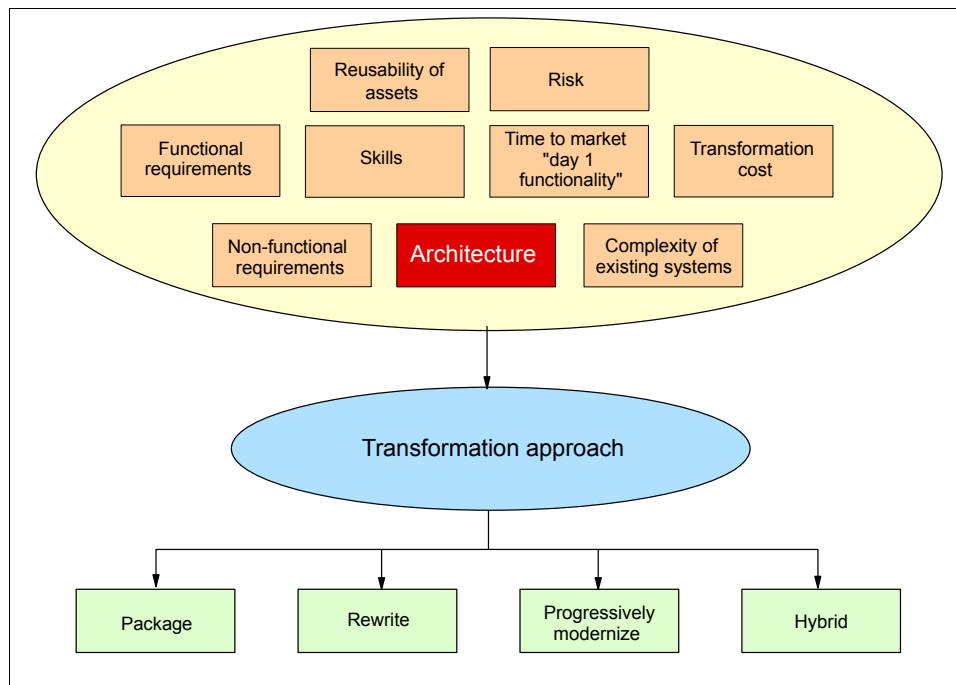


Figure 6 Factors determining the core banking transformation approach

Selecting the best approach

A bank's objectives and strategy should determine the approach, together with transformation objectives and the scale of the agenda. Although there is no one answer for all banks, in Figure 7 on page 16 we provide some guidance for selecting the right style.

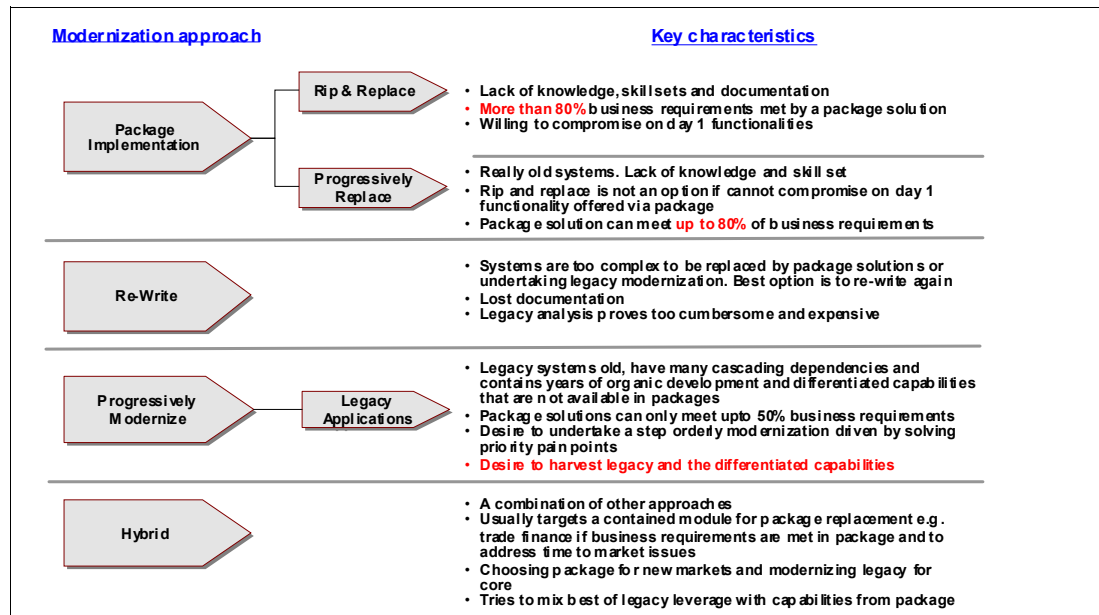


Figure 7 Transformation approaches

During the selection process, banks must refer to the following elements of a successful transformation:

- ▶ Adopt architectural design principles and core architectural constructs before committing to an approach.
- ▶ Mature the existing architecture, integration and SDLC environment before beginning the transformation.
- ▶ Exercise due diligence to align business with IT, create the business case, design and optimize the roadmap, and implement governance and organizational structures before beginning the transformation.
- ▶ Give due diligence to the legacy system, making sure it is documented and streamlined; also unravel any complexities wherever possible.
- ▶ Factor scenarios for experimentation, failed roadmap attempts and business cases.
- ▶ Factor resourcing and skill issues before beginning the transformation.

Packaged solution approach

Banks can customize packaged solutions to accommodate their business processes, operating structures, customer and product structures, and local country regulatory and compliance formats. The process can be completed all at once or by progressively moving legacy systems functionality into the new packaged solution. One advantage of this method is that a bank can leapfrog generations of development and thus accelerate the transformation. However, there are several drawbacks to this method, such as the following:

- ▶ Need to customize

Since every package must be customized to meet operational needs, the bank must consider how much customization will be required. Extensive customization can result in systems that do not remain in synch with future releases, and that are costly to maintain.

- **Package integrity**

This is a major challenge as the bank integrates packaged software with legacy systems. Packaged solutions contain specific architectural constructs, for example a customer information file (CIF). If the bank's strategy is to integrate the package with systems that are not being transformed, the integration needs to link the package with its architectural constructs to those systems and their architectural constructs. The more complex the legacy system, the more integrations will be required.

These types of integrations can be costly, time consuming, and unpredictable, making them a leading cause of budget overruns, frustration, and even failure. But keep in mind that even a total replacement strategy is not immune to such challenges. In addition to integration issues, another challenge is incorporating package capabilities to mimic bank operations: as the process continues, a gap can grow between what functions the package provides contrasted with capabilities that must be customized to meet bank requirements.

- **Package architecture**

Architecture should determine the choice of the package and is an important consideration when using this approach. The higher the architectural modularity inherent in the package, the fewer the customization and integration challenges. Using a package that requires significant changes to the existing infrastructure can increase the complexity and risks of the transformation. Packaged solutions should adopt the bank's technology architecture and infrastructure. If they do not, the bank will need to deal with two levels of complexity, one in the package and the other internally. The bank must also evaluate the downstream net benefit against total cost of ownership and the level of future performance.

Many packaged solutions are based on product silo design, for example deposit or loan modules, and are closely knit front to back. The challenge here is to integrate the package horizontally across business lines to support a customer-centric business model rather than a product-centric one.

Packaged solutions that contain substantial hard coding and deeply embedded user interfaces, data, business logic, and business rules in a product module (for example a deposit module) are more difficult to customize than packages that provide modules as separate architectural constructs. Also, packages that provide clearly defined hooks to accommodate customization are better than more rigid packages.

Separating applications—in specific business logic and business rules—from data is emerging as an important determinant to transformation complexity, cost and risk.

- **Current bank architecture**

Banks must have a mature technology architecture and a well-defined technical governance model to successfully transform core systems. Architecture limits, or red flags, include non-matured integration architecture, lack of master data, too many point-to-point interfaces, lack of an integrated development and tooling environment, lack of technology governance, and limited availability of skilled staff. All of these can dramatically increase the difficulty and complexity of integrating a package solution because the existing architecture is a poor foundation for the transformation.

- **Open solutions**

The packaged solution must not limit a bank's ability to adapt to future changing business conditions. Banks must choose a packaged solution that allows them to deliver business capability in an adaptable environment after the transformation. Agility and flexibility can be compromised downstream if a bank selects a package that locks architecture and infrastructure.

- Financial commitment

The bank must remain committed to the transformation process because there are no exit points without economic consequences. Capital investment is part of the financial commitment required to drive the transformation and, once made, may be difficult to recover in case of roadblocks in the process.

Rewrite approach

Banks may try the rewrite approach if they feel legacy systems are too old and complex to be harvested and transformed. They conclude that available packages are not suitable for their complex business processes and anticipate better results from rewriting the core system in newer technologies and architectural paradigms. Typically these banks have substantial IT discipline, a large IT staff, and have developed a deep pool of IT talent.

Hybrid approach

The hybrid approach combines both the packaged solution and customized progressive approaches and is often used with complex legacy systems. Banks can use a package for contained or compartmentalized domains such as trade finance, treasury, cash management, and risk management. Then they use a customized approach for their core market, or home country, while reverting to a package approach for international subsidiaries. The hybrid approach requires the same architectural discipline as all other approaches.

Customized progressive approach

This approach involves progressively adding, modifying and enhancing capabilities to core systems and is the one used by most banks. A customized approach requires a roadmap that outlines the master transformation plan, using an agreed upon governance process. The roadmap ensures that there is strong alignment between the business and IT and that everyone uses a strong governance process. All this ensures that the transformation successfully achieves its objectives.

Figure 8 on page 19 shows this progressive approach.

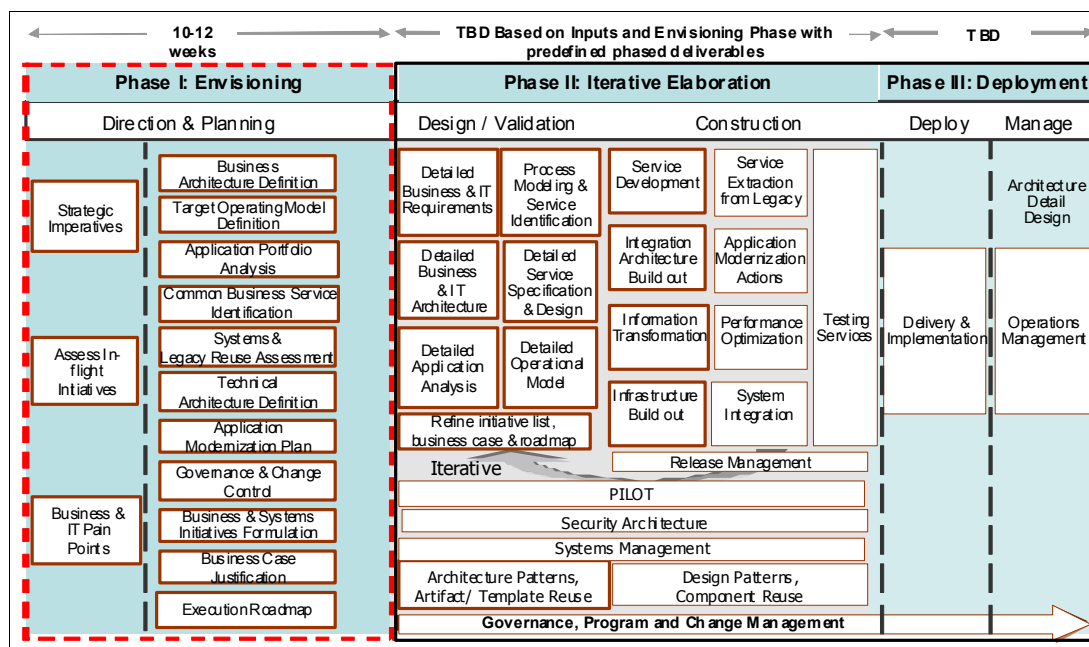


Figure 8 Customized progressive approach

Banks using this approach continually reprioritize projects and adapt transformation schedules accordingly. Guided by their roadmap, banks can continue to leverage assets that are still effective and change only those assets that are not performing optimally. While similarities exist, progressive transformation differs from the piecemeal modernization of the past in that it leverages technological innovations and adopts holistic architectural principles to fundamentally transform the system's building blocks. This architecture-driven approach provides tremendous agility, flexibility and time-to-market advantage while improving overall efficiency and deliverability of business changes.

Challenges

Even though banks using the customized progressive approach have achieved great success, there can be challenges along the way, namely the following:

► Roadmap

Progressive transformation requires a well-thought-out strategy, a roadmap with a long horizon plus a strong commitment of senior leadership to realize the roadmap. The roadmap should detail the progressive build of key architectural constructs and then retrofit the existing environment to use those constructs; see Figure 9 on page 20. The roadmap must align with the business objectives and capabilities necessary for the bank to successfully compete in the marketplace.

The developed roadmap is a mixture of tactical and strategic initiatives that are executed within the architectural discipline laid out by the bank. Executing the roadmap should be supported by a business case where the investment dollars are spread out throughout the transformation journey, though banks can change the transformation pace to accommodate their investment spending.

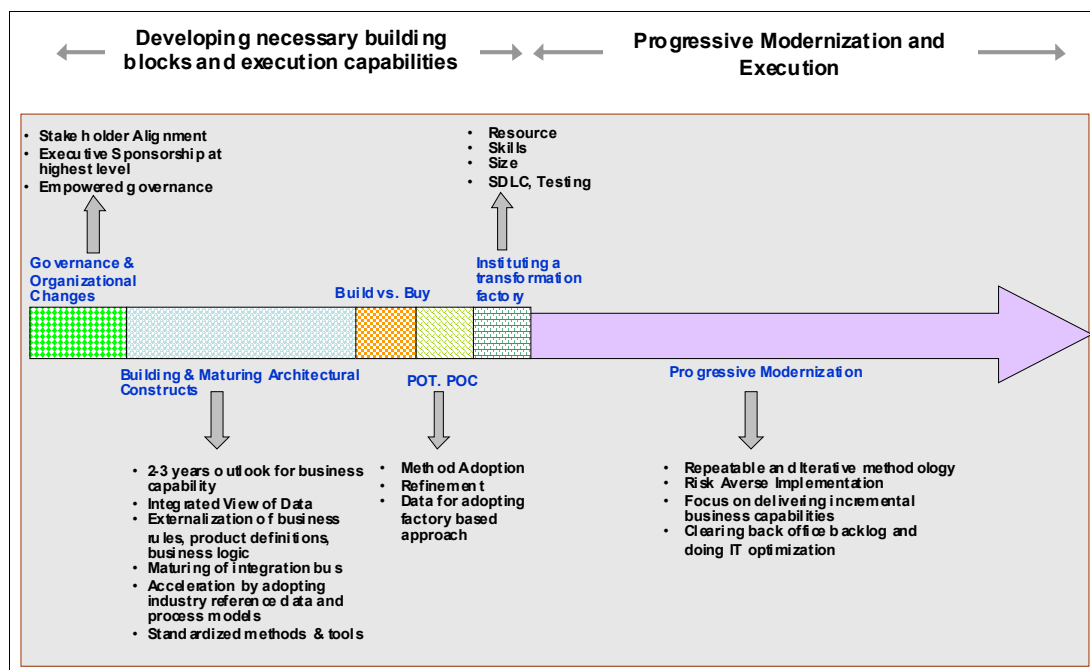


Figure 9 Example of a roadmap

► IT maturity

Banks need a matured IT organization to pursue the customized approach. Strong architectural discipline and technology governance must ensure that there is minimal non-compliance to standards. Banks also need an experienced pool of talent, including adept architects who can translate business objectives into architectural considerations in the roadmap.

► Integrated development environment

Banks can set up an integrated development environment to drive an end-to-end transformation. This environment allows the bank to define componentized business architecture, to identify critical components that will drive strategic imperatives, to model underlying processes that support component capabilities, and to derive the top-down service definitions that will be built to deliver the business capability. The integrated development environment also allows for tools to analyze the legacy environment and to identify assets that will be leveraged as well as those used for top-down service definitions. These elements are included in the design phase of the environment, which in turn provides the technology environment with the tools and technologies to build, integrate, and test processes.

The integrated development environment should provide all of these capabilities through a common framework with a tight governance structure. A similar framework should also be created to provide tools and technologies that will support the involved workforce in building skills and competencies.

► Governance

It bears repeating that thorough governance controls the transformation process, provides a preemptive view to risk and allows for proactive measures to mitigate those risks. Thus governance plays a crucial role in a customized approach. Limited governance with unenforceable oversight can damage progressive renovation, especially where a piecemeal approach is used. Further problems arise when business and IT are not aligned. For best results, such governance should be centralized. Banks using federated governance, where pieces of architecture or application portfolio are owned and managed

by different stakeholders, can have difficulty driving the transformation. Governance priorities must be aligned, set up and coordinated by a cohesive team. Banks can control priorities and achieve favorable outcomes by placing accountable enterprise governance at the top of the transformation effort and then defining a clear business-aligned strategy and roadmap.

► Strong architectural discipline

Banks driving a customized transformation must adopt a strong architectural discipline and its guiding principles. The architecture must be aligned at three levels of specificity: business, technology, and infrastructure. At each level, changes to the architecture must be guided by architecture principles such as separating architectural concerns from data, and externalizing business rules and logic. The bank must enforce these principles across the organization using a strong technology governance discipline.

Current legacy systems are unduly complicated as a direct result of compromised architectural discipline that occurred over time. To prevent repeating this mistake, banks must develop an architecture with clear definition and purpose. Many banks are using SOA to drive their transformation efforts. As we have discussed, when systems are componentized these components are then exposed as a service. Service components are then orchestrated to deliver a business outcome that is simpler to deliver than through hard coding. Common architectural principles adopted by banks are outlined in Table 1.

Table 1 Common architectural principles

Construct	Description
Separate channels	Separate core channels so that one can be changed without changing the other.
Separate data	Separate data so that embedded data and hard coding between application logic and data is removed from the application. Data is externalized as a set of master data that represents a single truth. Master data is further defined at a more specific granular level, for example customer, product, and contract in order to make data versatile and to support combinations of different data sets. The goal is to achieve the best level of granularity to orchestrate data elements to deliver the optimum business outcomes. For example, separating customer, product, and contract data elements provides flexibility and agility to permute a customer's contractual relationship with bank products.
Externalize business rules	Banks use rules engines and repositories to externalize business rules outside applications. This provides deep flexibility since a rule can be changed once in the master rule engine and that change can cascade and be automatically read by dependent systems, as opposed to changing each dependent system.
Externalize business logic	Embedding business logic, for example logic to compute fees and charges for loans and mortgages, will require application maintenance each time a business requirement changes. This is expensive, time consuming, and affects the time to market and competitiveness of the bank. By externalizing business logic outside the application, banks can significantly minimize the need to cascade changes across the systems.

Construct	Description
Adopt middleware and integration technologies.	Use sophisticated integration middleware to avoid multiple point-to-point interfaces in connecting discrete units of technology architectures. This ensures architectural separation by allowing pointed development and changes without changing the whole system.
Use industry reference models.	Use these to standardize banking processes and data models. Models contain predefined service definitions that can accelerate adopting a services led approach. Industry standard definitions also help to establish a common vocabulary between business and IT, help with service definitions by eliminating granularity questions, and provides process, data and use cases for faster implementation. Banks can scale infrastructure to support interconnection between components without compromising scalability and throughput. With this architecture, banks can componentize their technology into granular components, each with a defined interface through which it communicates with other components. As long as service definitions and interface points do not change, the component itself can be changed without impacting other components. Componentization also allows better sharing of common components through the middleware, resulting in valuable return on investment and increasing speed to market. Banks using middleware and integration technologies invest in the process of identifying the business and technology blocks that can be componentized—developed once, then used multiple times by multiple systems. Using middleware also allows banks to separate front office channels from back office systems, to publish commonly used business and technological services, and to use services to mediate the underlying complexity of the legacy systems.

Continuing the process

After completing all the steps required to select the best transformation approach and creating the roadmap, and after considering all the interdependent IT and business issues, banks can undertake the actual transformation process. However, as we have stressed, governance, accountability, continued analysis and flexibility are paramount as the process continues and as the bank manages the complexities of integrating legacy and transformed systems.

Managing and delivering transformation projects

Banks need professional project management and delivery capabilities to successfully drive the transformation process. A large, complex, multi-year transformation journey can be taxed with risk, disruption to the business and staff changes, all of which require a robust program management structure that will withstand various internal and external issues. In addition, the transformation is often politically charged, which makes it difficult to accommodate the involved staff at the same level of priorities.

Successful banks address these issues by creating a separate unit to drive the transformation or by bringing in transformation professionals, or vendors. This frees the transformation team

from organizational politics and fosters neutrality and objectivity. A bank can also create a separate transformation company with full responsibility for providing advisory consulting, design and development. Separating the staff involved in the transformation from the staff running the bank brings more defined purpose, separates responsibilities and helps to avoid conflict of interests. Banks look to transformation professionals for independent opinions, objective guidance and experience.

Banks need to consider the following when creating the project management and delivery options:

- ▶ Vendor experience

The vendor must demonstrate the total experience and capabilities necessary to drive all aspects of the transformation, from strategy shaping, infrastructure and runtime environment decisioning, to scaling large projects. The vendor should also have solid experience in running large transformation projects. Banks should evaluate the vendor's balance sheet to ensure that it can absorb transformation risk. Smaller vendors can present risks to managing a large scale transformation program.

- ▶ Transformation initiatives

Banks need to determine which transformation initiatives will be handled internally and which will be outsourced to the vendor. Many banks focus on creating the architectural constructs and engage vendors to provide advisory consulting or testing and data migration services. A consulting vendor can provide an outside view to help develop the transformation strategy and roadmap. Outsourcing tasks such as testing and data migration can leverage the economies of scale provided by reputable vendors.

- ▶ Factory based approach

Banks must plan for the large scale deployment of resources necessary for a successful transformation. They must evaluate the vendor to ensure that it can provide all the necessary resources to deliver the transformation, especially if the bank is using a vendor-specific package. Successful banks invest heavily in setting up a transformation factory that hires, trains and updates the skills of the delivery resources managing the transformation. When vendors participate in the transformation factory setup, banks have the flexibility to scale the resource mix and to adjust the number of factory resources to align with the transformation pace. The transformation factory also manages moving the work effort and makes sure that it is developed, integrated, tested, and delivered efficiently.

Project-by-project method

As they continue in the transformation process, many banks at this point proceed with a project-by-project method. This avoids the high risk rip-and-replace approach and achieves a steady stream of return on investment (ROI) results. A project-by-project method includes the following elements and steps:

1. Mature bank business and IT architecture.

After securing organizational commitments, the bank should focus on maturing key aspects of its business and IT architecture. Matured architecture results in the cleaner slate needed for a successful transformation. The alternative is spending resources to clean up existing architecture or to transform inferior design.

Some aspects of architecture maturity include the following:

- Adopt a componentized business architecture that supports a common vocabulary in all bank dialogues.

- Use industry reference models to create standard definitions that can be used to accelerate the transformation.

- Clean up the application portfolio.

Banks should create a clean state before beginning the transformation by standardizing legacy applications and by collapsing, consolidating, or sunseting applications that are not aligned. This step saves transformation time and effort since the legacy system will be smaller and consist of components that remain valuable. Be sure to document the legacy code—you cannot build a renovation plan without knowing what you have. Higher productivity is another benefit of this step.

- Build key architectural constructs.

Build key architectural constructs such as master data, rules engine repository, process servers, services repository, defined messaging interfaces, middleware implementation, and maturity. All these constructs are necessary to derive maximum value from the transformation.

- Build the integration hub.

Build the hub, then begin to aggregate channels, decommission P2P interfaces, componentize legacy systems, and serve up web services.

- Organize data.

Develop master data management for client, product and account to help deliver on the business demands for a holistic client view, product bundling, and single account.

2. Select the transformation vehicle.

Determine whether the transformation will use a software package approach, a customized approach, or a hybrid of both. A bank will have a clearer view of the best method after cleaning the legacy environment and building key architectural constructs. Choosing the best method will guide technology investment and minimize transformation risks.

IBM suggests a concurrent top-down and bottom-up view of the bank and its supporting technology. These views meet in the middle and define the *to be* architecture that includes IT, application, information, integration, and infrastructure architectures. This method, done in a progressive manner, offers the greatest flexibility while mitigating risk. Because the approach is progressive, the bank can scale the transformation pace as business conditions and the appetite for investment change. A bank can better manage cash flows and realize tangible results in a shorter time frame, usually six months, rather than the three to five years necessary to complete the entire transformation. More banks are adopting the progressive approach as the transformation process becomes more widespread in the industry.

- Top down view

Review medium-term—three to five year—business objectives and focus on business levers required to achieve financial objectives. This evaluation should result in a roadmap that will include strategic initiatives, as well as the business and technology pain points that need to be addressed.

Develop a component model of the bank's business architecture to identify the basic building blocks of the business, determine the components, and isolate those presenting immediate opportunities for growth, innovation, or improvement. This remarkably efficient discovery helps to prioritize initiatives and to ensure that operational and capital expenses are aligned with the bank's overall business strategy. The discovery process also identifies redundant and missing business components while highlighting new opportunities.

Define a similar component model for the target IT architecture and align it with the business component model. IT components can be a combination of existing assets, modernized assets and new assets to be acquired or constructed. In all cases, the process should separate architecture concerns from user interfaces, data, business logic, and business rules.

- Bottom-up view

Identify the bank's unique differentiators and leverage them to create new value. The deliverable is a transformation plan that uses a SOA, retrofitting the applications and systems to fit into a new modular design that provides better flexibility, resiliency, and agility benefits.

Using the top-down, bottom-up view enables the bank to:

- Build common services and decommission them from the legacy environment: EDM, BPM, BI, statements, needs analysis, pricing engines, audit, compliance.
- Restructure the legacy code, in line with business priorities.
- Leverage package solutions that are aligned with the architectural principles adopted by the bank.

3. Create an integrated development environment.

Select the appropriate method and create tooling and assets to integrate the development environment that will drive the transformation. An integrated environment provides strong governance and manages risks better.

4. Conduct a Proof of Concept (PoC) and Proof of Technology (PoT).

Conduct a PoC and PoT to test the transformation approach and technology. Both the PoC and PoT provide insight into the risks, necessary skills, methods, tools and technologies necessary for the transformation.

5. Create the program office.

Establish a program office to oversee the transformation. The program office must be empowered to make decisions and allowed to interconnect with business-as-usual ventures of the bank.

6. Create a transformation factory.

Use the PoC and PoT experiences to create the transformation factory, where the required resources, skills and training needed to drive the transformation will be housed. Use external vendors to provide requisite skills such as testing and data migration at lower costs.

7. Use the progressive approach iteratively.

Continue with the transformation iteratively and progressively. It is critical that the transformation process includes architectural constructs, a defined business and technology roadmap, strong governance, integrated development environment, and a transformation factory.

Keep funding liquid.

Use a powerful business case to obtain internal financing. Usually business and IT stakeholders develop the business case and then get approval from the board or management. However, financing may be difficult to obtain even when the business case is effective. Some banks find the cost of transformation daunting and are not able to take the plunge. It is possible for banks to secure funds externally from vendors or by raising additional capital. Vendors can fund the bank's transformation as a financing vehicle. They can also contribute transformational elements to offset the bank's investment and defer the payment until realization of benefits.

Transformation benefits

Banks that have modernized their core banking systems have seen dramatic improvements in their overall operations and profitability. Benefits include the following:

- ▶ Improved time-to-market for new products and services
Banks have reduced time-to-market for new products from months to weeks: this increased market share and improved customer retention.
- ▶ Implemented product bundling and relationship pricing
Bundled accounts result in increased cross-sell opportunities and results, increased customer balances (lower cost of funds), reduced account maintenance costs, and reduced risk. Customer satisfaction and retention also increased.
- ▶ Decreased business-as-usual expenses
Lower development and maintenance costs have reduced business expenses.
- ▶ Reduced operating expenses
Reduce costs by eliminating labor intensive activities while providing significantly improved visibility of operational data.
- ▶ Quicker entry into new markets
Banks with a technology and operations template that can be reused effectively—with minor localization costs—can open new businesses in emerging markets within months, leaping over the competition and achieving first-to-market advantages.
- ▶ Accelerated project delivery
Reduces the cost and duration of systems projects that roll out new functions and features.
- ▶ Increased productivity
Banks have leveraged the IBM application development model and methods to drive consistency across development teams, centralizing key development activities and driving delivery excellence.
- ▶ Reduced testing effort
Banks that leveraged common tools for automating testing processes, including regression and performance testing, have realized lower infrastructure costs.
- ▶ Reduced time-to-market
Banks have reduced launch time for new products or product bundles.
- ▶ Reduced costs by consolidation
Banks have successfully balanced resources across product teams, increasing the available pool of resources to minimize staffing peaks and lows.
- ▶ Reduced costs by outsourcing
Banks used global capabilities to provide planning and execution activities in lower-cost locations.
- ▶ Improved quality
Banks identified defects earlier in the development process and reduced the overall number of product software defects.
- ▶ Improved decision making and saved costs by applying a process
- ▶ Banks applied best practices and innovation to standard application process model and methods.

- Improved customer retention and implemented relationship pricing using a single customer view

Single customer view and enhanced customer detail information enabled a bank to improve cross-sell capabilities, increase wallet share, and maximize customer profitability, resulting in a four-fold increase in customer loyalty.

When to measure benefits

Banks should analyze projected versus actual benefits over both the long term and short term. IBM suggests that banks adopt the Earned Value Analysis approach where each project segment is held accountable for projected benefits, and course corrections are applied until each goal is achieved.

Banks that use a progressive approach to drive modernization have achieved tangible results in shorter time frames than banks that waited until the end of a long transformation period.



Transformation methodology

The Banking System Modernization Methodology (BSMM) is a structured approach developed by IBM based on decades of experience in transforming systems to match the evolving business needs and industry technology advances. BSMM helps banks plan and execute a progressive transformation of their core banking systems.

A banking transformation engagement deals primarily with migrating legacy systems to modernized systems using service oriented architecture (SOA). Typically, such engagements begin with business modeling and IT architecture definition, which lead into the design and development of the IT solution. Component Business Modeling (CBM), a technique defined by IBM, is used for business modeling.

The IT portions are delivered through Service-Oriented Modeling and Architecture (SOMA), another method defined by IBM. SOMA is an end-to-end delivery process for developing complex service-oriented applications and solutions. In addition to CBM and SOMA, a core banking transformation requires additional processes and methods that define activities related to governance, program and change management. IBM provides these methods, along with accelerators such as reusable models, transformation tools and work product templates, as part of a framework called Core Banking Transformation Framework (CBTF).

BSMM phases

BSMM takes a phased approach towards modernization that is based on the types of activities performed during the project life cycle. The modernization consists of three phases, each further divided into sub-phases with specific activities and assigned roles. In addition to activities required for modernization, BSMM includes references to IBM products and reusable assets, ranging from specialized development tools to work product templates, reference architecture, and code assets. These help to execute the steps outlined in the methodology. Major BSMM phases are as follows:

Phase 1 - Envisioning

A high-level analysis focusing on the client's vision and strategy that involves the basic building blocks of modernization.

Phase 2 - Iterative elaboration

Designing and constructing the modernization solution outlined in the envisioning phase. Phase 2 expands on the outputs of the envisioning phase and details requirements, design and architecture of the modernized system, followed by the implementation and testing activities.

Phase 3 - Deployment

The modernized system is deployed and managed. Phase 3 includes activities necessary to deploy the modernized system to production, followed by maintenance and support activities that continue through the end of the project lifecycle.

BSMM approaches to transformation

The main approaches that BSMM uses to modernization are the top-down and bottom-up, and the model-driven approach that is illustrated in Figure 10.

Top-down versus bottom-up approach

In the top-down approach, current business and technology architecture are analyzed and a modernized system is designed based on Service-Oriented Architecture (SOA). In the bottom-up approach, existing legacy systems are analyzed and refactored to fit into an SOA design. All these activities are distributed across the three BSMM phases. This BSMM approach is illustrated in Figure 10. Completing the phases results in a system that has the modernized versions of the existing business processes identified as business services with certain service implementations referring to existing legacy code.

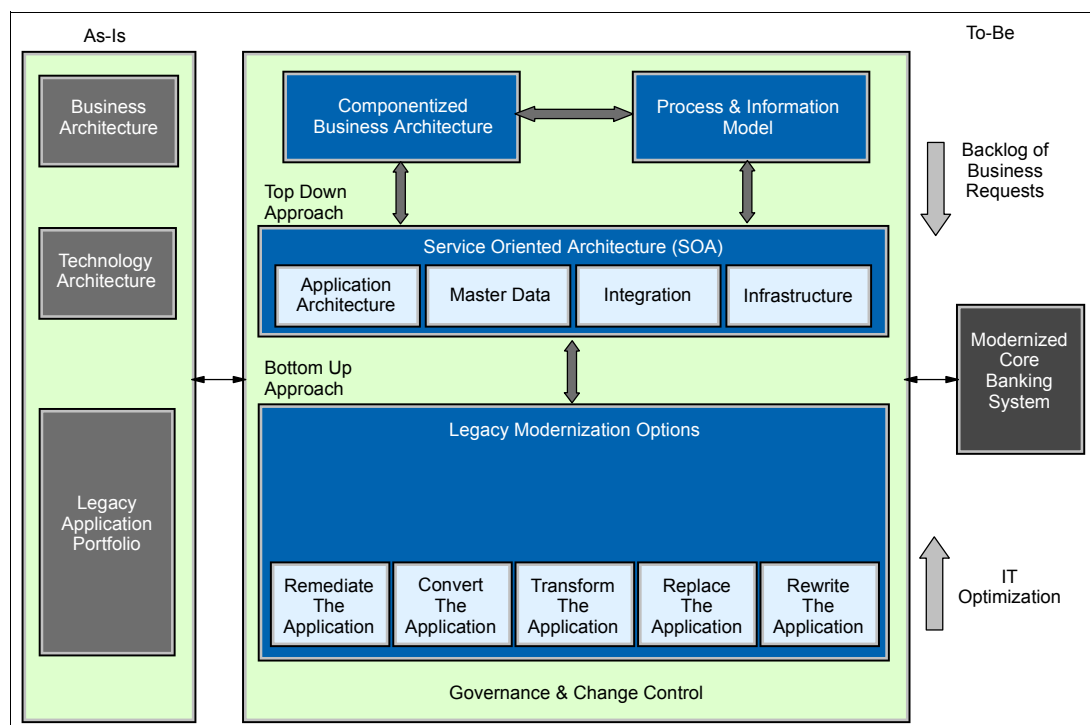


Figure 10 BSMM approach to modernization

Model-driven approach

BSMM model-driven approach to modernization is performed by defining the business and IT systems as models that are eventually converted to source code through a series of automated transformations. At each stage in the transformation, additional IT details are added to the business models, thus enabling a smooth handover from business to IT. The business architecture is modeled through *component business modeling* and *business process modeling*. Business services are identified and attached to the process models. The process models and services are then transformed into a *solution model* containing the *service model*, business entities and their state machines in UML. Figure 11 illustrates the model-driven approach to modernization.

The solution model is enhanced by adding additional services identified through legacy system (bottom-up) analysis and by adding detailed design specifications for the identified services. The UML content in the solution model is then transformed into source code artifacts such as Java classes, WSDL interfaces, SCA state diagrams, and BPEL code. Logical data models are created and transformed to physical models.

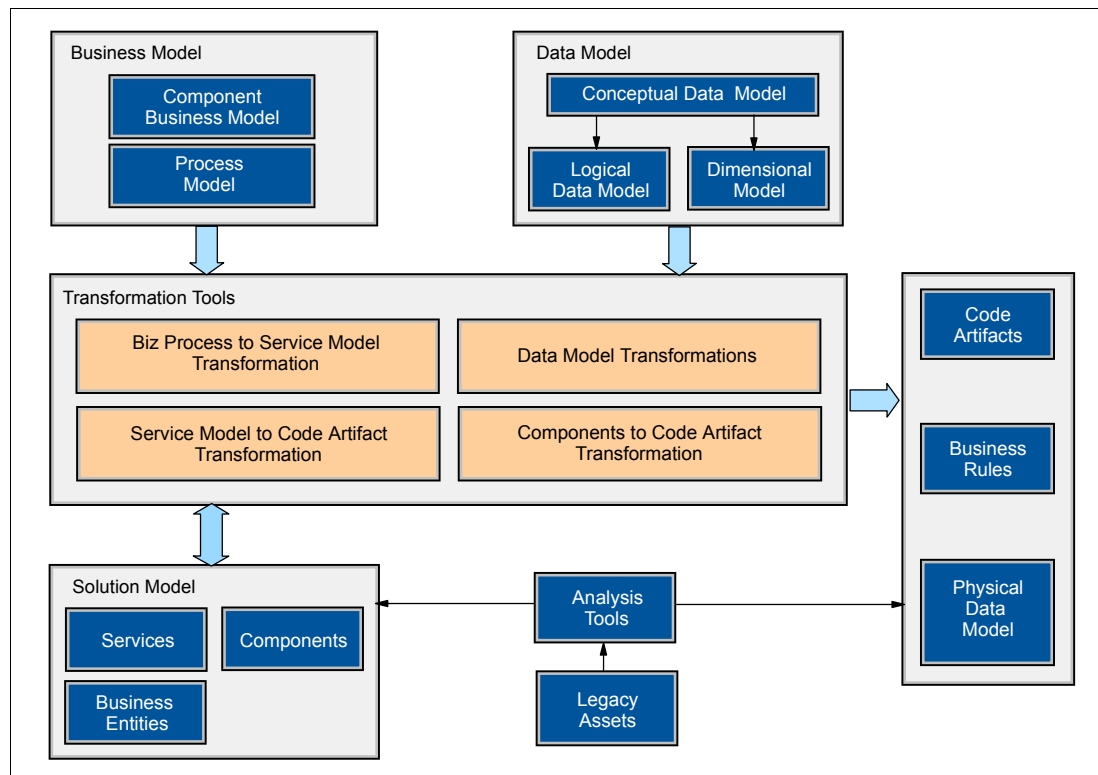


Figure 11 BSMM model-driven approach

Governance

Governance activities are performed throughout the modernization engagement beginning with the envisioning phase onto the deployment phase. Banks that already have a governance model in place at the corporate and IT levels should analyze and evaluate the existing model to determine what facets will be applicable to the modernized system. Based on this analysis, The governance model should be updated with policies and guidelines based on this analysis and as indicated by the architecture and technology of the modernized system.

Questions related to governance as addressed at the envisioning stage are:

- ▶ How to leverage the bank's IT governance model in order to deliver the change?
- ▶ How to handle the funding for cross enterprise services?
- ▶ How to sustain the momentum for this long journey?
- ▶ Who will manage the bank's architecture trade-off decisions?
- ▶ What KPIs will be defined and measured?
- ▶ How to ensure that there is no governance "overload"?

Based on the answers to these questions, the bank's governance team lays out a governance plan that will be used as a foundation for governance activities for the remainder of the engagement. Governance activities are usually driven by a center of excellence that lays out the policies and ensures compliance. BSMM includes guidelines on program governance such as benefits management, stake holder management, risk management, roles and responsibilities.

Since BSMM uses an SOA-based approach to modernization, make sure to consider SOA governance best practices during the modernization. SOA governance is an extension of IT governance, which is an extension of corporate governance. IBM provides methods and tools to help implement SOA governance, for example IBM SOA Governance and Management Method (SGMM), WebSphere® Service Registry and Repository (WSRR), DataPower® and Tivoli® Composite Application Manager (ITCAM).



Using a framework for transformation

Banks are adopting a variety of modernization approaches including progressive transformation. A number of technology advances over the past several years offer viable solutions for a progressive core banking transformation such as the following:

- ▶ SOA helps to decompose the problem space into a set of loosely coupled service components that integrate through a service bus. This provides a new approach to support the agility and flexibility that business is demanding, in both building new core banking applications, or renovating existing applications.
- ▶ Using SOA as a set of architectural principles along with Model Driven Development, IT can build applications that use rich industry models and derive solutions that align with business goals.
- ▶ Legacy environments often contain multiple sources of master data. Key advances in master data management (MDM) enable IT to consolidate the master data into a central repository, providing a 360-degree view of customers, products and accounts. This data can be exposed to enterprise-wide applications and business processes through a service bus.
- ▶ Business process management (BPM) elevates business processes along with their modeling, development and execution from the siloed application-centric approach to a common set of service components that can be choreographed externally to provide flexibility.
- ▶ Advances in business rules management (BRM) centralize management and governance and free them from the application providers. This reinforces the agility that business needs to expand to new markets or to optimize operations.
- ▶ While SOA, MDM, BPM, and BRM provide the architecture, modeling, development, execution and monitoring, we still need the key solution ingredient of legacy analysis and discovery. These technologies allow you to browse through millions of lines of legacy code, and to identify pieces of business logic and rules that can then be elevated into the middle tier. This enables IT to develop business processes that span across applications, such as cross LOB product bundling, cross-sell and up-sell, and customer-centric pricing, and so forth.

The framework

IBM and its business partners provide industry solutions that meet specific strategic business and IT objectives. These solution offerings leverage IBM and partner industry assets and best practices and business applications. To support these solutions, IBM provides industry frameworks to support deployment and integration.

A framework is a repeatable methodology and uses reusable assets with an underlying set of design rules that solve a class of problems. Frameworks give the bank the flexibility to deploy multiple solutions at its own pace while using elements already in the bank's IT portfolio. The result is a faster implementation with less risk than with alternative approaches. The framework includes the following:

- ▶ Next generation foundational architectural constructs and market leading software and hardware platforms that bring modularity, agility and lowered TCO to the banking architecture, for example message brokered enterprise integration, master data, system management and governance services, and SOA capabilities.
- ▶ Proven reference models for best practice banking processes and data and service models to develop differentiated capabilities such as predefined banking service specifications, for example evaluating operational risks, or checking customer status, and so on.
- ▶ Prebuilt solution templates and accelerators to speed time to market in both migration and development of next generation banking architecture, for example bank-specific customer master templates, solution templates for account opening, product bundling, dynamic pricing, and so on.
- ▶ Proven methodology and business-specific usage patterns to lower implementation risk, for example legacy modernization methodology, data migration methodology, and SOA governance.
- ▶ Capability to seamlessly integrate third-party business applications such as the leading ISV packages for front office, payments, risk, and core banking.
- ▶ An industry standard approach to align technology with business needs, for example computing resources and throughput optimization.

Many banks approach transformation progressively, implementing solutions one project at a time. These projects stand on their own and leverage the value created in previously completed projects. Many project areas can be used as points of entry for a framework-based solution implementation. Such project areas, and typical projects contained in each area, include the following:

- ▶ Architectural transformation
This project area enables banks to transform their core banking platform, reduce operational cost and risk, and improve core banking process efficiency. The transformation is accomplished using projects that address fundamental capabilities, including a simplified IT infrastructure, platform scalability, enterprise-wide master data management for customers, contract, and product data, and model-driven development to build business service components.
- ▶ Banking process agility
This project area helps banks to improve profitability by expanding into new markets faster, bringing innovative products to market quicker, and differentiating pricing and terms for maximum customer satisfaction. These projects help banks implement more flexible and efficient processes such as account opening and management, product bundling, and dynamic relationship pricing.

- Banking application modernization

This project area aids in transitioning from existing core banking application silos to service-oriented, componentized architecture in a staged and modular way, with near-term payback and reduced risk and disruption. Banks achieve this by a combination of integrating best-of-breed application components, renovating legacy applications, building new SOA components, all of which lead to a customized, lower cost, and less risky solution.

The four pillars of transformation

Core banking transformation is founded on four pillars built on an industry framework foundation that includes best practices from engagement experiences, industry standards, innovations from customer PoCs and first-of-a-kind projects led by IBM and early adopters. The four pillars are *method and tools*, *accelerators*, *architecture*, and *integration platform* as shown in Figure 12. Together, they offer the following:

- A comprehensive methodology and tooling that aligns business requirements with IT capabilities.
- A set of solution accelerators to jump start transformation engagements.
- A flexible architecture that supports a variety of transformation patterns.
- A scalable and robust integration platform to support the architecture-led progressive transformation.

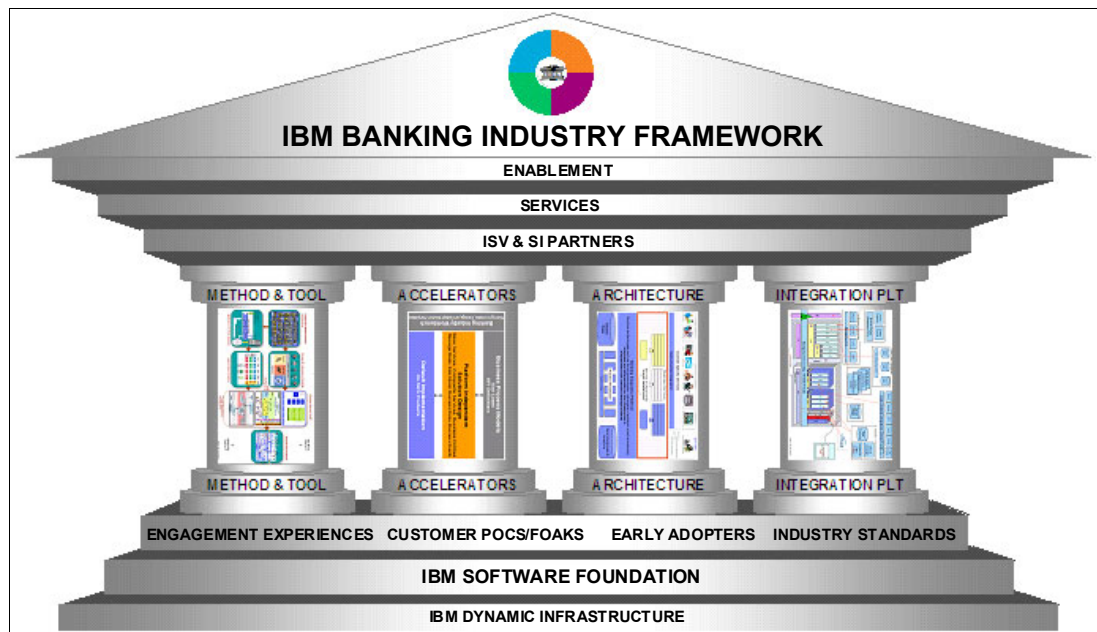


Figure 12 IBM banking industry framework

Pillar 1: Methods and tools

The banking industry framework defines a comprehensive methodology that aligns business and IT, leverages the IBM business architecture assets and industry models, and also integrates bottom-up legacy analysis and discovery with top-down business-driven transformation.

This methodology, based on IBM's experience with complex SOA projects and banking transformation engagements, is a “meet in the middle” approach using both top-down and bottom-up threads. It uses best practices and industry standards to greatly accelerate the top-down approach. These approaches are illustrated in Figure 13.

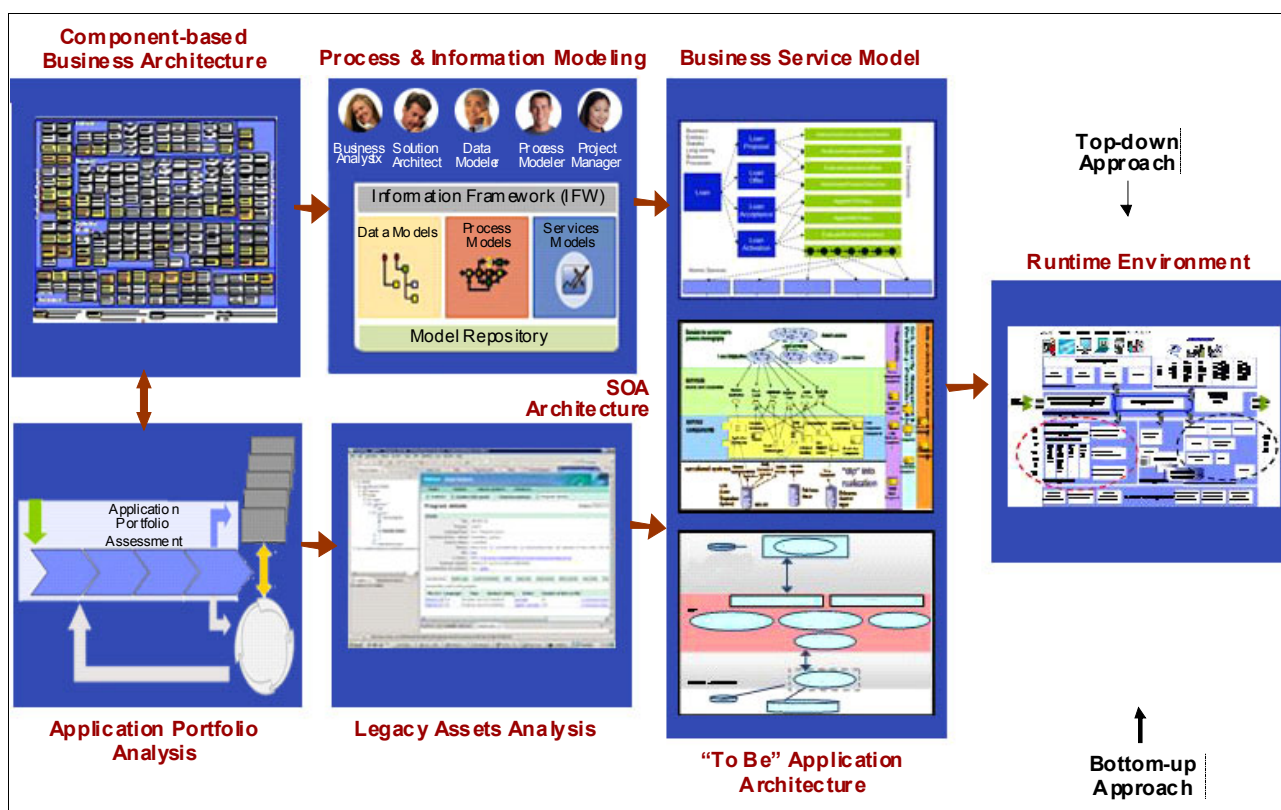


Figure 13 Top-down and bottom-up approaches

On the business side, we define bank business priorities along with the processes and service components required to support those priorities. The *component business model* (CBM) is the starting point used to identify and target the business functions and activities that will be transformed.

Once the business functions are identified, we map the CBM components to information framework models and then identify the information framework (IFW) tasks and processes that support the CBM component. The mappings are used to extract relevant processes and information models from IFW. Obviously, all models are customized to meet specific bank requirements.

We then create an SOA solution design based on the models analyzed in the IFW processes. The solution model is refined to integrate the new SOA components with the legacy systems. The bottom-up approach, or thread, is introduced at this point. Typically, this involves a tool-based application portfolio analysis that is driven from the strategic imperatives identified in the business architecture.

Next, we apply legacy analysis and discovery techniques to identify the service touch points, and message and data requirements in the legacy applications. We then combine this information with the top-down analysis to create a *to be* application architecture and solution design. Finally, we use the resulting solution model to generate runtime artifacts that can be deployed on the integrated middleware platform. This approach ensures that the business

goals are aligned with the IT goals. The top-down and bottom-up approaches are further illustrated in Figure 14.

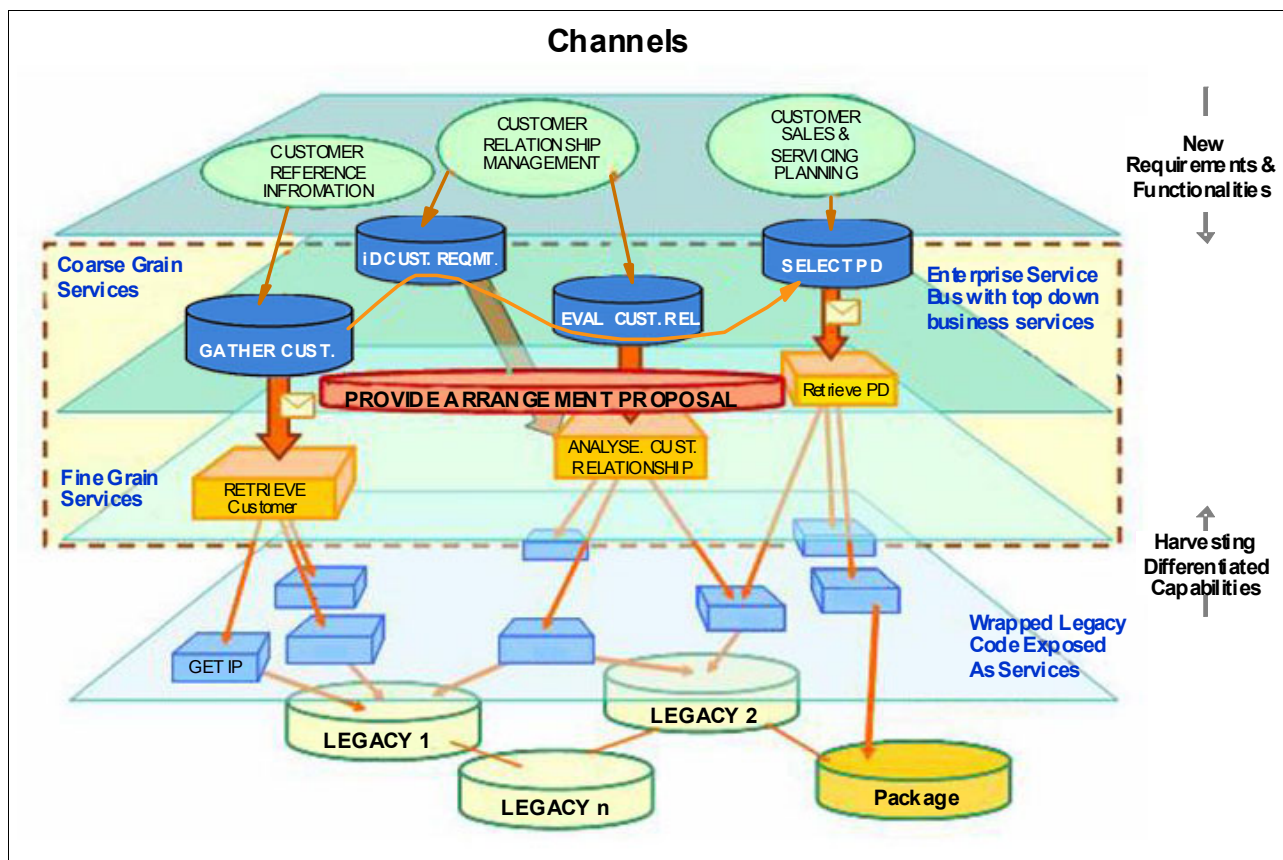


Figure 14 How the top-down bottom-up approach works

In Figure 14, the top-down approach identifies a set of coarse grained services for a specific business process such as “provide arrangement proposal”, driven by analyzing new requirements and functionalities. These coarse grained business services are decomposed into fine grained technical services. Now these fine grained services are newly developed by wrapping existing applications or packaged application components. This is where we harvest differentiated capabilities using the bottom-up analysis. Once the application functionality or components are identified using a bottom-up analysis, they are wrapped as services that are then mediated to the fine grained services as identified in the top-down analysis.

This meet-in-the-middle approach allows you to harvest existing functionality whenever possible and while continuing to address the new business requirements. You can progressively modernize legacy applications and bring in new functionality as packaged application components without disrupting business. In this approach we clearly separate the architectural concerns, which greatly eases the integration and leverage of enterprise capabilities. We provide a reference model for granularity of the services and design principles as part of the industry frameworks and models. By providing a service mediation layer between the core banking services and the channels, we hide the underlying complexity; this gives banks the flexibility to change either the front or back end without either impacting the other.

Figure 15 on page 38 illustrates how the meet-in-the-middle approach works.

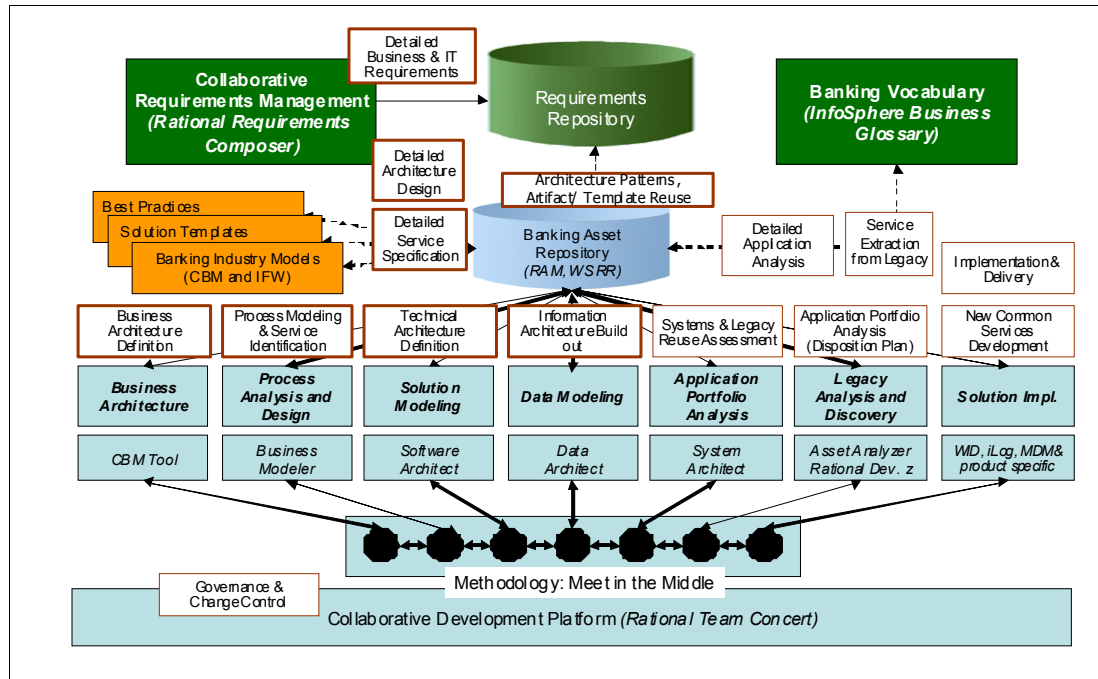


Figure 15 How the meet-in-the-middle approach works

The IBM Banking Transformation Workbench

The integration methodology is supported by the *Banking Transformation Workbench* (BTW), an IBM integrated tooling environment that implements solution templates as part of a core transformation project. BTW further integrates the framework methodology and assets with IBM development and modeling tools from WebSphere and Rational® brands, all on a collaborative integrated solution development platform designed for core banking. BTW enables engagement governance through tasks, work items, and artifact management. It also provides method and assets guidance to enable task-oriented development and automation and assists in creating and transforming engagement artifacts. BTW is a comprehensive, all-in-one tool that provides a multitude of core banking transformation capability patterns. It enables a model-driven development approach that semi-automatically derives and transforms SOA runtime artifacts from CBM maps and industry models.

Pillar 2: Solution templates

The banking industry framework provides prebuilt accelerators to jump-start the implementation, namely banking industry solution templates. The templates are developed through customer engagements and harvested, formalized and generalized to a set of reusable industry software components. Templates are essentially industry software with configurable business processes, business rules and data models, all grounded on the following principles:

- ▶ **Narrow focus**
Concentrate on a specific problem, for example account opening as opposed to using a generic method to solve business problems.
- ▶ **Use a model-driven, componentized, service-oriented solution to a business problem instead of building an application.**
- ▶ **Formalize the solution structure and behavior for the target problem.** For example, define the components for an account opening solution, their interfaces and behavior.

- Provide the solution's out-of-the-box reference implementation on the IBM middleware that provides the starting point for modernization projects.
- Deep configurability
Adapt the solution template for a specific bank by configuring the business process model, business rules, and data model.
- Solution components become part of an enriched library because they were developed with actual customers and harvested and hardened from one client engagement to another.

Each solution template contains the following sets of artifacts:

- Business requirements captured as a set of business processes, use cases and key performance indicator (KPI) definitions that are derived from industry standards and IFW models.
- A platform-independent solution design is a collection of models—base services, composite services, business service components message model, data model, business rules, and business events. These models can be adapted to a specific bank runtime platform.
- A complete reference implementation of the solution design on the IBM software platform with stubbed-out integration to the core applications.

You can customize the solution templates at any of the artifact levels with BTW. The solution templates use a component model that defines behavior. Figure 16 shows the component model.

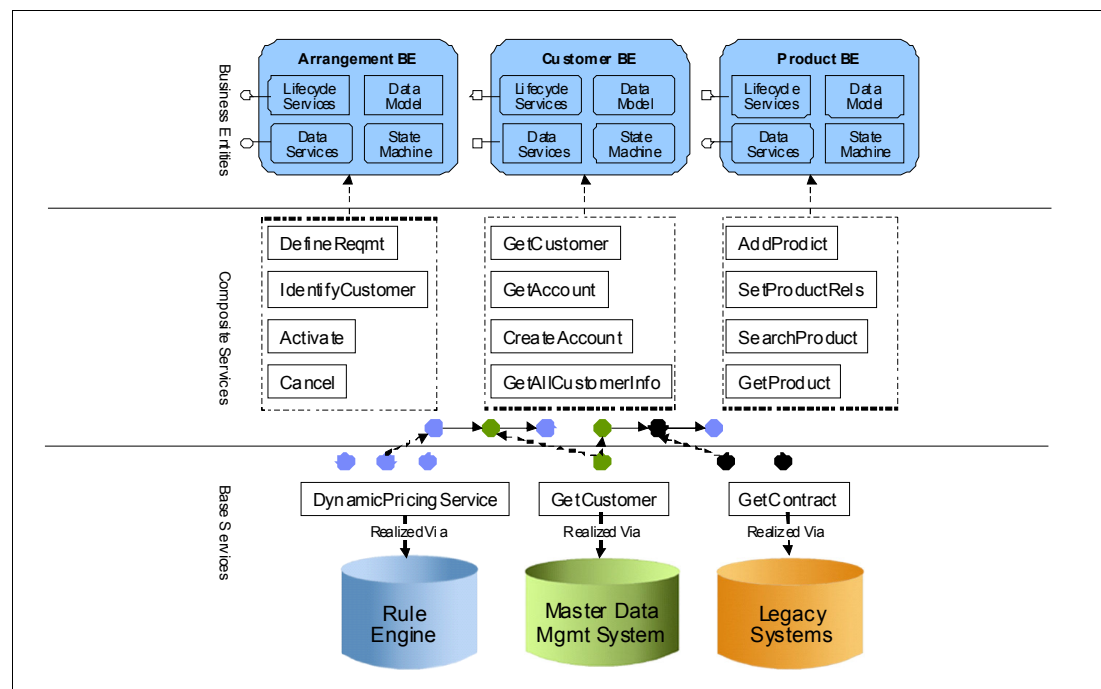


Figure 16 Template solutions - component model

Solution component types

We have identified three types of components based on behavior, as follows:

- **Base services**

These encapsulate business functions as atomic services, that is, they cannot be further decomposed from a service perspective. You can implement a base service using a rule engine, master data management server, human task management systems, and product processors. You can also program a new base service in Java, Cobol or other languages. Another option is to realize a base service using legacy systems, for example a legacy CICS® application may expose specific functions as Web services.

- **Composite services**

These are composed from base services using a flow composition model. The composite service component has a service interface and a flow model that defines the sequence in which the base services are invoked. You can implement these with a flow engine. Composite services can be interruptible, that is, executing them can be suspended to wait for a return from a base service invocation. Multiple composite services may invoke the same base service.

- **Business entities**

These are the dominant business objects of the core banking domain. A business entity has a lifecycle model, a data model, and an interface model. The lifecycle model describes the states of the entity, events that cause state transitions, and the actions performed as a consequence of state transitions. These actions can include invocations of atomic or composite services or can send an event to another business entity. The interface model further describes two kinds of services exposed by the business entity component: the lifecycle services that change the state of the entity, and the data access services that create, update, and retrieve data attributes.

These business entities can be composed into various solution models. The component model allows you to reuse base service components and to rapidly develop new solution templates. For example, you can use the arrangement, customer, product, campaign, account and condition service components to support the product bundling scenario. Figure 17 on page 41 displays solution component details.

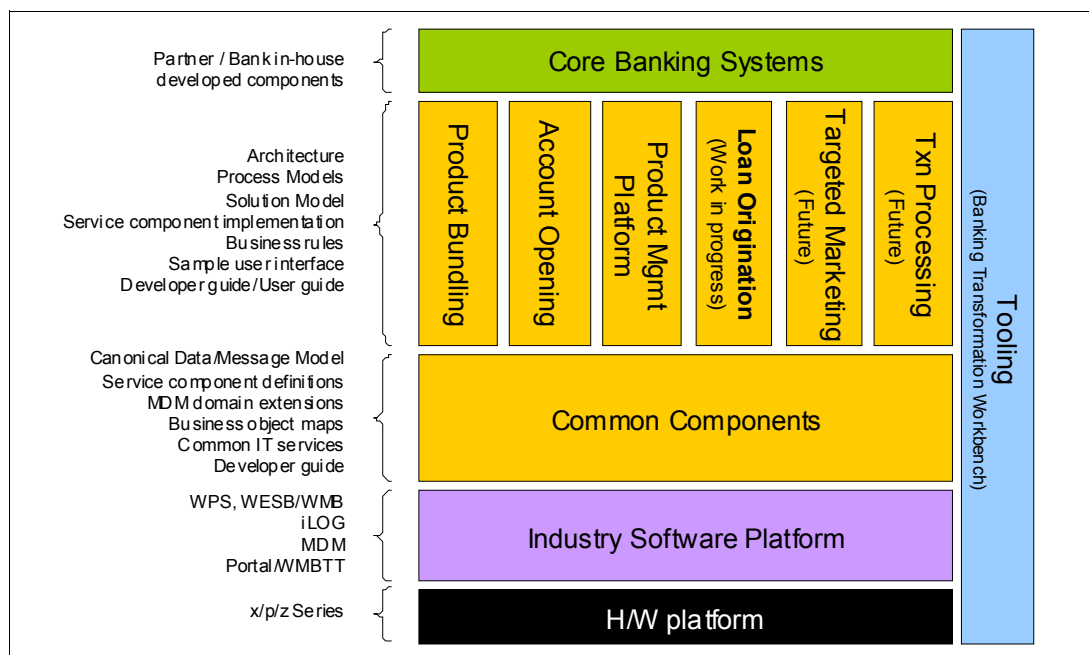


Figure 17 Solution components

Available solution templates

Solution templates are packaged as service assets and are delivered through service engagements. They use an IBM optimized software platform that contains key components such as WebSphere Process Server, Enterprise Service Bus, WebSphere Message Broker, iLOG Business Rules Management System and InfoSphere® Master Data Management Server. IBM also provides a basic user interface to drive solutions based on WebSphere Portal Server and WebSphere Multichannel Branch Transformation Toolkit.

Common components are employed to increase reusability across templates, including canonical data and message models, service component definitions, MDM domain extensions, interface maps, business object maps, and common IT services. In addition, IBM has developed specific use cases such as product bundling, account opening, and a product management platform. Each use case contains architectural work products that aid the sales and delivery process, process models, solution models, service component implementations and business rule implementations.

The solution templates also contain developer and user guides to support the bank teams as they customize specific solutions. To repeat, templates are not applications; they require integration with a bank's core applications, both those developed in-house or bought from third-party vendors.

Currently, the framework supports the following three solution templates:

- ▶ Product management platform
- ▶ Dynamic product bundling and relationship-based pricing
- ▶ Universal account opening

Each solution template is described in Table 2 on page 42.

Table 2 Description of available solution templates

Solution template	Bank business challenges	Solution outline	Bank benefits
Product management platform	<ul style="list-style-type: none"> ▶ Need to streamline product introduction and decrease time to market. ▶ Difficult to create and customize innovative products based on market requirements. ▶ Difficulty in customizing product offerings per individual customer results in reduced profitability. ▶ Inability to create bundles across LOB products, for example deposits, loan, and cards. ▶ Need to dynamically structure product bundles and offerings for a specific customer. 	<ul style="list-style-type: none"> ▶ A collaborative product development platform where various role players can develop and launch new products together. ▶ Uses rich industry content to develop highly flexible and extendible product schemas to support disparate product processing systems. ▶ Robust product catalog component that is integrated through an ESB with the consuming application. ▶ Developed on a highly scalable middleware platform including MDM Server, MDM, PIM, iLOG Business Rules, Enterprise Service Bus and Information Server. 	<ul style="list-style-type: none"> ▶ Build a central product repository with complete product definitions that support complex product hierarchy, structure, attributes and terms and conditions. ▶ Externalize business rules, for example pricing rules, interest calculations, eligibility criteria and product recommendations. ▶ Manage entire product lifecycle from concept to launch with the flexibility to quickly change, bundle, cross-sell and up-sell products to address regulations, market and compliance requirements. ▶ Integrate product data with consuming applications such as core banking system, contract origination, data warehousing, revenue forecasting and recognition systems. ▶ Measure product performance through product profitability analysis, customer segmentation analysis, and other business intelligence techniques: new products and product changes are driven by an understanding of what works and what does not.

Solution template	Bank business challenges	Solution outline	Bank benefits
Dynamic product bundling and relationship-based pricing	<ul style="list-style-type: none"> ▶ Need to support new customer-centric business models. ▶ Need to dynamically bundle products and services. ▶ Need to componentize and simplify core systems for better operational efficiency and flexibility. 	<ul style="list-style-type: none"> ▶ Constitutes a first step in transformation roadmap, that is, the design, development and deployment of a dynamic product bundling solution. ▶ Uses BTW (CBM, IFW, WebSphere Business Modeler, Rational Software Architect, and WebSphere Integration Developer) for solution design and implementation. ▶ The runtime stack includes WebSphere Process Server, InfoSphere MDM Server, iLOG Business Rule Management System. 	<ul style="list-style-type: none"> ▶ Supports customer-centric business processes by leveraging customer, product, and arrangement data available on the Enterprise Service Bus. ▶ Grows business by attracting new clients with customized product offers and service bundles. ▶ Architecture aids in quicker response to new market opportunities, for example regulatory changes. ▶ Simplifies IT with loosely coupled, reusable service components derived directly from business models. ▶ IT operational efficiency with end-user-leading SOA foundation products. ▶ IT development efficiency using generation of service components from IFW models.

Solution template	Bank business challenges	Solution outline	Bank benefits
Universal account opening	<ul style="list-style-type: none"> ▶ Need to streamline contract origination process across lines of business. ▶ Business team requires flexibility, agility and reduced time-to-market to roll out business process changes. ▶ Process flow control is divided between front end and application processing engine without clear guidelines. ▶ Regional and country level customizations to address variability-created heavily customized services that create maintainability and extensibility issue. 	<ul style="list-style-type: none"> ▶ Scalable, robust account opening solution that addresses all functional and business requirements. ▶ Process is shared and reused across various products and locations. ▶ Built on standardized technology stack providing extensive BPM capabilities. ▶ Developed on highly scalable middleware platform including WebSphere Process Server, MDM Server, iLOG Business Rules, Enterprise Service Bus, WebSphere Service Registry and Repository. 	<ul style="list-style-type: none"> ▶ Decreases time-to-market. ▶ Aligns business and IT. ▶ Promotes reuse via componentization and decoupling. ▶ Provides business agility and flexibility. ▶ Introduces new products and services faster and more dynamically. ▶ Faster response to business changes. ▶ Variability across countries and regions. ▶ Empowers by consistently introducing capabilities. ▶ Monitors and manages business and events, including key performance indicators (KPI), via middleware. ▶ Supports auditability and compliance via middleware. ▶ Dynamically changes business processes and business rules. ▶ Reduces application complexity and maintainanc. ▶ Focuses on componentization and decoupling. ▶ Minimizes custom framework complexity by leveraging middleware runtimes, products, and tooling.

Pillar 3: Framework architecture

The heart of the architecture is an enterprise integration platform built on open standards to support a multitude of integration patterns such as process choreography, service mediation and content-based routing, service registry, business rule execution and legacy application adapters. The framework architecture is displayed in Figure 18 on page 45.

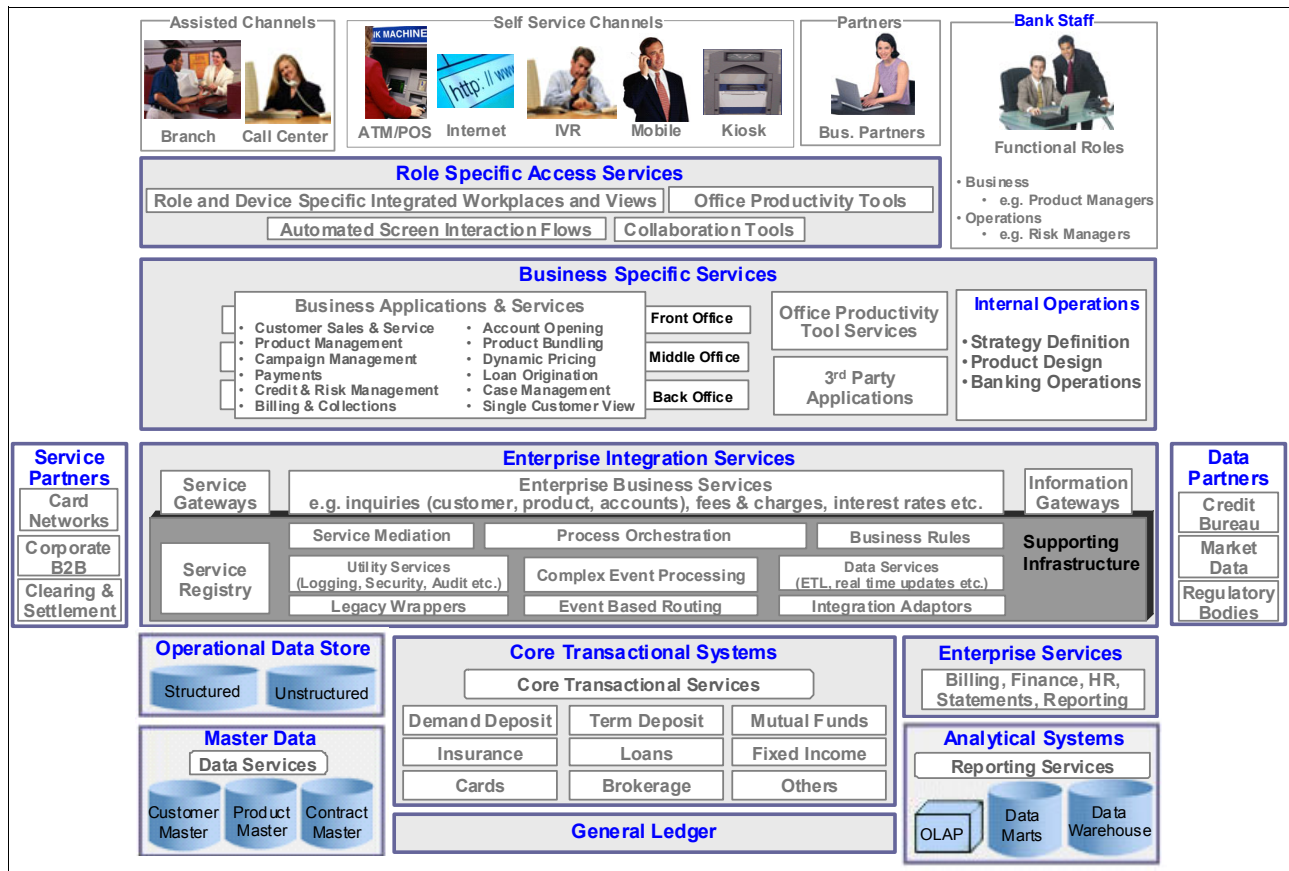


Figure 18 Four pillars of core banking transformation, architecture

This architecture provides many benefits including faster time to market, more innovative products, and better insight into how they are performing while allowing banks to adjust their pricing and product mix to meet the needs of specific customer segmentations. The architecture also provides process agility to meet the changing business needs and enables growth through scaling of hardware and middleware. Additionally, as the complexity decreases, so does the cost of system maintenance.

More features of this architecture include the following:

- ▶ Core transactional systems such as deposits, loans, cards, wealth, and insurance product processors are supported through transaction services. These services are choreographed through the integration platform.
- ▶ Master data services that provide a single truth of data are captured as a common repository, for example the common customer information file, a common product catalog, and reference account data. These are exposed as services for use through the integration platform. Also, operational data can be managed and exposed to the applications.
- ▶ A set of business-specific services is built on the integration platform to support front and back office operations. This set can be combined with office productivity tools and third-party applications to support internal bank operations such as strategy definition, product design, and relationship management. All these business-specific services are served through channels and internal banking portals using a role-specific multichannel integration layer.

Pillar 4: Integration platform

The Banking Industry Framework provides industry-optimized integration on a runtime platform derived from the rich portfolio of market-leading SOA software. A key component of the runtime platform is the InfoSphere MDM server. Many core banking transformation projects begin by consolidating customer data. The MDM server provides an integrated view of customer data and exposes it as services on the Enterprise Service Bus. The MDM server also provides consolidation of product and contract information and makes it available through a services interface to the enterprise-wide business processes.

The integration also contains WebSphere Process Server that provides dynamic, componentized, model-driven business process orchestration with WebSphere Enterprise Service Bus or WebSphere Message Broker. These two features provide service mediation and data mediation that integrate with legacy applications.

Additional integration components are the WebSphere ILOG® business rules engine that externalizes the business rules development, management and execution, and WebSphere Business events which support the detailed event processing required in a complex integration pattern.

Overall, the integration framework provides a robust IT service management with Tivoli software supported by IBM hardware and operating systems, all of which provide scalable and high performance computing environments that support high volume transaction processing and flexible business process execution.

More information about how such an integration platform can be designed, based on the IBM zEnterprise® platform, can be found in “Core banking systems infrastructure” on page 47.



Core banking systems infrastructure

The core systems environment typically serves as the back office for many of the bank's business functions. The expectations with regards to performance, availability and reliability are high, while there is focus on reducing or at least optimizing the TCO.

Many core systems environments in large banks are running on IBM mainframes. The mainframe provides a high quality environment that exactly fits the purpose of most core systems, which is to provide a high volume transactional environment with excellent availability and reliability. However, as discussed extensively in this paper, many of these environments have grown into an inflexible and hard to maintain web of programs where new integration requirements are hard to implement.

Infrastructure design methodology

When a core banking transformation project is started, the infrastructure should be redesigned based on the new application architecture and the nonfunctional requirements (NFRs). We suggest to follow a structured methodology to reach these infrastructure decisions. Also, this process should start early, because some infrastructure decisions may impact design decisions at the application architecture level. At a minimum this methodology should include the steps outlined in Figure 19 on page 48.

Important: Infrastructure solution design should be done early in the overall core banking transformation project in conjunction with the application architecture of the new environment. Performing this step once all code has been developed would be a big risk. Infrastructure may influence or even determine certain application architecture and middleware decisions. So, application, middleware and infrastructure architecture design are three activities that need to be balanced and not executed in a “waterfall” manner, that is, sequentially.

It should not be assumed automatically that the new modernized core systems environment should run in the same infrastructure configuration as the old one. The new architecture will most likely require more processor, more integration capabilities and, because of extensive reuse of shared services, high availability. The only way to find out what is needed is to go

through a structured methodology that starts with analysis of the nonfunctional requirements (NFRs), the logical architecture, and the service level objectives (SLOs).

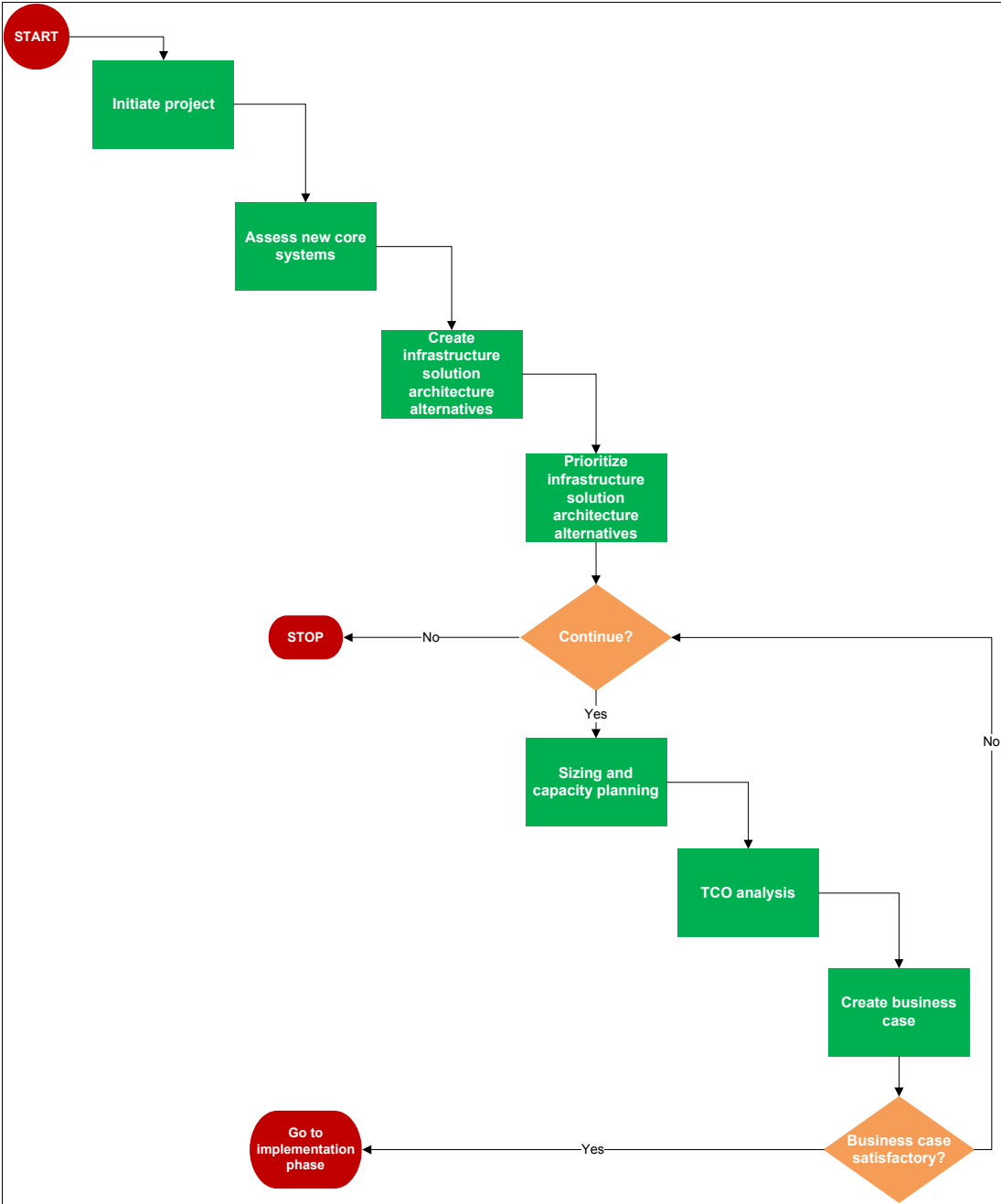


Figure 19 Process for determining the infrastructure for the new core systems

The following sections contain a brief explanation of each step and decision point.

Initiate project

In this activity it is important to define the following:

- Expectations regarding the outcome of the project

It is important that everybody understands the background of the project and expectations of the stakeholders. For example, expectations could exist regarding documents and reports to be produced or to convince somebody of a certain solution.

- Scope of the project

The scope of an infrastructure design project is typically set by specifying the core banking applications or workloads to be included. A selection could be made based on various criteria, such as applications with a similar architecture, applications belonging to a specific business unit or geography, or applications associated with a product domain (loans, mortgages, savings, and so on).

- Project organization

It is essential that all required stakeholders and subject matter experts are “on board” and committed to perform their part during the project. Not everybody is necessarily required in all activities, though. Both IBM and the client will be represented in the project.

- Terms and conditions (T&Cs)

It is important to agree on the T&Cs for the project. Is there any funding required and who provides the funding? Also, there needs to be agreement on interim milestones.

- Project plan

We highly suggest to create a project plan, covering the different stages, their work products and milestones, and assumptions and dependencies.

- Tools needed

In some of the activities certain tools should be used. Some of these tools may be for IBM internal use only. The tools should be defined up front, and also who will use them.

Assess new core systems

This activity includes a variety of techniques, questionnaires and workshops to obtain a good understanding of the new core banking applications in scope. This is necessary to be able to define candidate infrastructure solution architectures in the next stage. It is in this activity where application architects, business users and infrastructure architects have to work together to make sure they have the same understanding of the new core banking application architecture. At a minimum the following information needs to be collected, analyzed and discussed:

- Application portfolio

Which new core banking applications have been defined and how do they relate to and depend on each other. Can they be grouped in specific domains, based on function, geography, security risk, and so on.

- Application architecture

What design principles have been followed in designing the new core banking applications? Examples are:

- Data access is always performed using data access services running in an application server.
- Business rules are executed in a business rules management system,

- ▶ **Middleware architecture**

Which middleware technologies will be used to execute the new core banking applications? Examples are:

- WebSphere Business Process Manager Version 7.5
- WebSphere Message Broker Version 7

- ▶ **Nonfunctional requirements**

These are requirements derived from the service-level objectives and agreements and describe the quality of the new applications in terms of availability, security and performance.

Create infrastructure solution architecture alternatives

Once a good understanding has been built of the new core banking systems architecture and its nonfunctional requirements, one or several infrastructure solution architectures can be developed. Each of the alternatives must meet the nonfunctional requirements and be able to run the new core banking systems environment within the stated SLOs, but one alternative may just be able to do it a bit better than another. The final decision on which one will be best can only be made when the sizing and the TCO analysis have been done.

Prioritize infrastructure solution alternatives

During this activity the solution architecture alternatives defined in the previous activity are prioritized. This prioritization is done based on qualitative criteria and the alternatives are matched with nonfunctional requirements (NFRs) and service level objectives.

Important: At this point no TCO analysis has been done yet. It could very well be that the most favorable solution architecture from a qualitative point of view is not necessarily the most favorable from a TCO point of view. In the last activity of this methodology (“Create a business case” on page 51) this evaluation is made with pros and cons.

Sizing and capacity planning

During this activity detailed information is collected to be able to size the required infrastructure configuration. This sizing is necessary to assure that the SLAs and SLOs can be met and that the TCO of the required infrastructure can be calculated. If this activity is not performed, it will be completely unknown how the new core systems will work out from a quality and cost point of view. Also, during this activity, a capacity planning is made for the coming years based on growth predictions. By doing this, future bottlenecks can be avoided and insight can be gained into TCO for the future.

This activity can be executed per solution architecture alternative or for all solution architectures at once. The latter will make it possible to make direct comparisons between the solution architecture alternatives with respect to sizing and capacity planning data. If this stage is executed per solution architecture alternative, the one that was ranked the highest in “Prioritize infrastructure solution alternatives” on page 50 would be the first one to examine.

TCO analysis

A proper TCO analysis can only be performed if all metrics are available for the applications in scope. That is why this activity can only take place after the previous activity. Like the previous

activity, this activity can either be executed per solution architecture alternative or all solution architecture alternatives at once. The latter will make it possible to make direct comparisons.

Create a business case

This activity is the final step in the methodology. Based on TCO information and the earlier defined attributes of the solution architecture alternatives, a business case is built. There are two options:

- ▶ Only one solution architecture alternative is justified and presented.
- ▶ All solution architecture alternatives are evaluated, ranked and presented in the context of both TCO and qualitative aspects.

Balancing the transformed architecture and applications

Core banking applications often span heterogeneous platforms, appliances and devices over a wide range of resources. This can create issues for IT as it works to meet demanding business and performance objectives at a competitive Total Cost of Ownership (TCO). These complex challenges underscore the criticality of a holistic systems design optimized around workloads. Such designs must support the established quality of service from the core banking system while extending that level of service to diverse workloads hosted on an optimized, yet unified, environment.

Modernization efforts typically implement a shared services model that consolidates and centralizes application management and hosting services. However, one size that fits all shared services models does not work; to be effective, modernization must use an appropriate level of commonality across business units, product lines, and regions. Accessing the right information at the right time across the extended enterprise is key to staying ahead of the competition, innovating faster, and improving operational efficiency.

Consequently, core banking environments must optimally balance standardization with local customization and be flexible to accommodate change from both internal and external organizations. The core banking infrastructure must integrate heterogeneous platforms and optimally utilize IT resources while cost effectively scaling demand. Many banks deploy multitier workloads on heterogeneous infrastructures. For example, mission critical back-end workloads and customer data management need the availability, resiliency, security, and scalability strength of the mainframe.

However, front-end workloads, such as access channel integration services and enterprise integration services are better suited for distributed architectures. Creating and managing these multiple workloads, especially when they are implemented on various physically discrete servers, can lead to inefficient and ineffective solutions. One way to address this issue is to use a deployment architecture based on heterogeneous virtualized processors that work together as one infrastructure. Such deployment architecture and infrastructure provides the following capabilities:

- ▶ **Flexibility**
A bank needs numerous options to scale servers and storage, operating systems and subsystems in whatever way the architecture demands. The infrastructure must be flexible to add capacity on demand while responding to demand spikes.
- ▶ **Quality of service**
The system should provide a high degree of service with high performance, reliable and secure servers, and storage.

- **Manageability**

Banks need high performance systems that allow them to manage and control various resources attached to the infrastructure. Integrated management of various hardware components provides tremendous flexibility in deploying, scaling or updating without requiring intensive system maintenance.

- **Manageable TCO and Total Cost of Acquisition (TCA)**

Infrastructure components need proven TCO and TCA, supported from both development and runtime environments, to run core banking systems. TCO and TCA should reflect not only the cost of hardware, software licenses or annual maintenance, but also the total cost to install components plus the space, facility, power and cooling consumed by the infrastructure.

The role of the IBM mainframe in core banking

Many of today's core banking environments are running on the IBM mainframe (System z® platform) under the z/OS® operating system, and use a transaction monitor such as CICS or IMS™. This is especially true for the larger banks in the mature markets. Various other applications, such as applications for payments, front office and customer relationship management (CRM) invoke functions in these core banking applications. These other applications may exist on other platforms, such as UNIX or Microsoft Windows.

It is not realistic to expect that all applications in a bank will ever run on one single platform. either a mainframe, UNIX, or Windows. That is why the focus should be on making heterogeneous platforms work together in a reliable and efficient way. Certain applications or application functions are a better fit on one platform than another and a “fit for purpose” infrastructure architecture should position the applications and their functions on the best fit platform.

So, which is then the best fit platform for a core banking environment? We can answer this question once we have analyzed the characteristics of a typical core banking application, its nonfunctional requirements (NFRs), its required service level objectives (SLOs), and its interfaces with the rest of the IT landscape.

Core banking application characteristics

The following is an attempt to generalize the characteristics of a typical core banking application:

- **Access to large databases with the bank's operational data**

These databases include all the information about business entities such as mortgages, loans, and savings. This data includes functional data, but also audit records, transaction records, and so on.

- **A significant amount of interfaces between the core banking systems and other systems in the bank's IT ecosystem**

These interfaces process large volumes of data, typically millions of records per day in a medium to large size bank:

- Real-time and batch interfaces with other internal “core” applications such as the general ledger (GL)
- Sometimes real-time, but usually batch, interfaces with external parties, such as the central bank, government agencies, clearing houses, and other banks

- Real-time interfaces with front-office applications
- Real-time interfaces with customer relationship management (CRM) applications
- ▶ Significant batch processing schedules, in certain frequencies, such as hourly, daily, weekly, bi-weekly, monthly and yearly

A part of these batch jobs is functional, but a large part is related to creating interface files and housekeeping (backups, consolidation, and so on).
- ▶ High security

The information processed in a bank's core systems is sensitive. Much of the information falls under policies and laws with respect to privacy. Also, if the information processed and stored in the core systems environment falls into the wrong hands, the bank would be exposed to serious fraud threats. Access to the core banking applications and its data is tightly secured, at multiple levels (both logical and physical).
- ▶ High availability within the agreed service hours

As explained earlier, core banking systems are tied into many of the bank's IT systems and they really have to be fully available during the bank's service hours. In the era of Internet banking, these service hours have become 24 hours per day, 365 days per year, in most countries. Now that we see a strong increase in customers using mobile devices for banking, this requirement will even become stronger. The expectations are that most of the bank's core products should be available for purchase over the Internet through a self-service portal.
- ▶ Audit trails

There is a lot of focus on the financial health of banks and regulations require a bank to be able to produce an accurate insight into its liabilities and assets at any moment in time. Therefore, core banking systems have mechanisms to guarantee accuracy of the information stored. Transaction and audit records are created to be able to prove and eventually reconstruct transactions.
- ▶ Performance

Core banking systems have consistent performance, and response times are typically sub-second.
- ▶ Scalability

In the pre-Internet era scalability was not really that important, because traffic volumes could be accurately estimated based on the bank's office hours and the typical volumes during the day. But now that an ever increasing portion of the traffic comes in over the Internet channel, the load is not always predictable. Especially when a bank runs certain promotions, sudden peaks may occur. Therefore, core banking systems have to be very scalable while meeting the required performance.
- ▶ Smart business logic

Most of the actual code in a typical core banking system either deals with data access (reading, updating, creating and deleting database records) or making decisions, for example applying a certain condition to a loan or not. User interface logic remains to be part of most core banking systems too, but tends to be placed under the front-office domain. It is the business logic that we are interested in, because this business logic encapsulates the bank's business rules and its intelligence, which determines its competitiveness.
- ▶ Significant application portfolio

A bank does not have just one application for its core banking operations, and most banks do not even have just one application per product type (such as loans or savings). Instead, it is typical to find multiple applications per product type and multiple applications that are

shared among all applications, such as a pricing application or customer information file application. There may also be separate applications providing services or utilities to all other core banking applications. Depending on the size of the bank, its maturity and its geographical scope, the amount of different core banking applications can go into the hundreds.

- **Strict maintenance windows**

A bank's core banking systems cannot be updated at any moment in time. There are strict windows in which no routine maintenance and new functional requirements can be implemented in production. This calls for very strict planning of new releases of applications, and updates have to have a fallback procedure. Deployment procedures are very strict too.

Now you may say that the above characteristics are not that unique and may apply also to other IT systems. Maybe that is true, but the combination of size and the financial impact of the information processed in core banking systems make it a specific challenge.

Technology impact

There are many large banks that have proven for decades that the IBM mainframe, and the z/OS operating system, CICS and IMS in particular, have no issues accommodating all the characteristics discussed. Nevertheless, there are banks who do not use the mainframe (that is z/OS) and some banks consider to shift their technology base. To become agile and adopt modern architectural principles as discussed earlier in this document, the technology base will need to be examined and reconsidered. Some of the possible shifts are:

- Move from one operating system to another, such as moving from z/OS to UNIX.
- Move from a traditional transaction monitor, such as CICS or IMS, to a JEE application server.
- Move from one programming language to another. For example, moving from COBOL, or even Assembler, to Java.
- Move from traditional data stores, such as VSAM, to a relational database, such as DB2®.

In all cases, a shift in the technology base for core banking systems has a major impact and it is not always certain if and how the NFRs can be met. An apparent cost saving by switching the platform and/or the middleware may look good in the beginning, but be completely undone by rising cost in operability and all kinds of quality issues with the new environment. Obviously, a shift in technology for all of the above examples at once is a major risk.

Core banking systems blueprint

By using the core banking transformation roadmap and an architectural blueprint of the future core banking systems environment, a "fit for purpose" study can be performed to determine on which platform each architectural construct fits best. The following is a generic list of these constructs.

Attention: Note that in the to-be agile core banking system environment certain constructs are not application-specific anymore, while in existing legacy applications these constructs are contained within application silos.

- **Multichannel integration**

These are basically the user interface components of core banking systems, but implemented in a common middleware optimized for user interaction on multiple devices.

- ▶ **Business process management (BPM)**
The BPM layer is a common runtime running business processes involving services from multiple back-end applications.
- ▶ **Service registry**
The service registry is also a common runtime used by all service requests throughout the entire core banking application landscape.
- ▶ **Business rules**
One of the focus areas in the transformation is to implement business rules under a “business rules management system”, instead of keeping them imbedded in procedural code. This business rules layer is also a common construct shared between multiple core banking applications.
- ▶ **Analytics middleware**
The analytics middleware is a set of middleware used for all core banking operations and consists at a minimum of a business data warehouse, a business intelligence environment for reporting, and an analytics environment for statistics and predictions.
- ▶ **Enterprise service bus (ESB) and other connectivity infrastructures.**
This shared middleware construct entails the entire communication infrastructure used between services in the core banking systems and between the core banking systems and other systems within a bank. A combination of different levels of communication technology may be found in this construct:
 - Enterprise service bus, applying SOA-style functionality to integration
 - WebSphere MQ, for message-oriented integration
 - RPC-style connectors
- ▶ **Master data management server**
As explained earlier, one of the higher priorities in core banking transformation is to implement master data management. The master data management server is also a common middleware construct shared between all core banking systems.
- ▶ **Core banking applications**
These are the applications themselves, organized per product or product type, encapsulating business logic and business rules and having access to operational data. The technology used per core banking application may be different and there is no rule that they have to use the same programming language, database, or transaction manager. It is, however, advised to implement functions as componentized services.

Figure 20 on page 56 shows these constructs and how they relate to business functions.

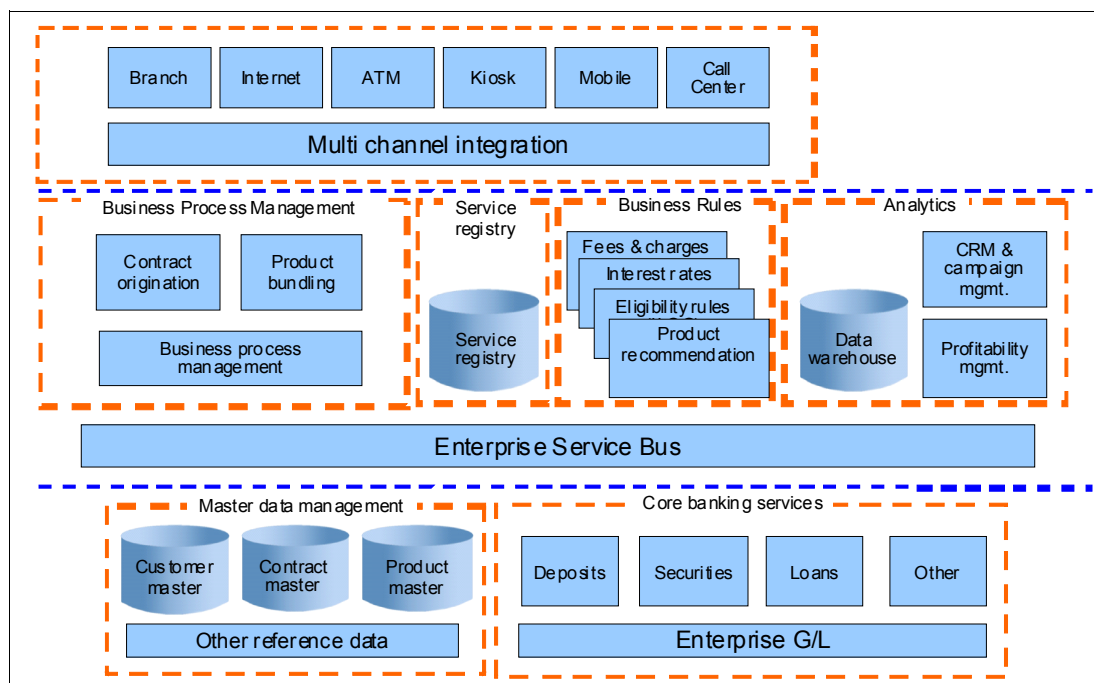


Figure 20 Core banking systems blueprint

Platform placement

Once you have a finalized blueprint, similar to the one shown in Figure 20, along with the nonfunctional requirements, you can perform the infrastructure design methodology as explained in “Infrastructure design methodology” on page 47.

The platform placement options for these constructs are shown in Figure 21 on page 57.

Workload category	Middleware	Platforms	Key
Multi-channel integration	<ul style="list-style-type: none"> WebSphere Portal WMBTT 	A Lz L W	1
Business process management	<ul style="list-style-type: none"> IBM BPM Server WebSphere Business Monitor WebSphere Business Events 	A Lz z	2
Service registry	<ul style="list-style-type: none"> WebSphere Service Registry & Repository 	A Lz z	3
Business rules	<ul style="list-style-type: none"> WebSphere IL OG BRMS 	A Lz z	4
Analytics	<ul style="list-style-type: none"> Cognos BI, TM1 and Metrics SPSS Modeler and Statistics 	A Lz I	5
Enterprise service bus and connectivity	<ul style="list-style-type: none"> WebSphere ESB WebSphere Message Broker WebSphere Transformation Extender WebSphere DataPower WebSphere MQ Adapters and connectors (e.g. CTG) 	A Lz z L D	6
Master data management	<ul style="list-style-type: none"> InfoSphere MDM Server InfoSphere Information Server 	Lz z	7
Core banking applications	<ul style="list-style-type: none"> DB2 on z/OS CICS/MS transaction monitor WebSphere Application Server 	z	8

<i>Platform legend</i>	A	z	Lz	L	W	D	I
	AIX	z/OS	Linux on System z	Linux	Windows	DataPower	IDAA

Figure 21 Platform placement options core banking systems blueprint

IBM zEnterprise system

Increasingly, business applications span heterogeneous platforms, appliances, and devices, and this wide range of resources creates real issues for IT shops trying to meet business objectives. Simply adding servers, routers, and other IT equipment ultimately will not solve your IT challenges, and may even make them worse. Even using virtualization techniques can only go so far in helping you to manage a massive number of servers, routers, and other devices. The ability to manage resources for these heterogeneous applications as one logical entity has been lacking—until now. The IBM zEnterprise System technology, referred to as zEnterprise, combines scalable computing power with a ground breaking new architecture that is able to manage heterogeneous workloads from a single point of control.

With its built-in management capabilities zEnterprise is perfectly positioned to meet customers' integration, automation, security, and cost requirements. To address these IT transformation issues, the zEnterprise system provides a new architecture, consisting of heterogeneous virtualized processors that work together as one infrastructure. The system introduces a revolution in the end-to-end management of heterogeneous systems, while offering expanded and evolved traditional System z capabilities.

Research outlines five key benefits of zEnterprise:

- Improved performance of the IBM compilers and the z/OS operating system
- Enhanced workload optimization scalability with the zEnterprise central processor complex (CPC) system architecture
- Better Linux workload consolidation, performance, and economics with z/VM®
- New support for extending multiplatform virtualization on the zEnterprise BladeCenter® Extension (zBX)

- Streamlined day-to-day operations and management with the Unified Resource Manager

The IBM zEnterprise system can host the entire blueprint, honoring classic IBM mainframe qualities of service and providing choice of operating system.

In the following diagrams (Figure 22 and Figure 23 on page 59) the core banking system constructs are mapped to a platform option on zEnterprise. The numbers indicate the link.

Note: The mapping shown in the diagrams is a *possible* or *likely* mapping. However, the mapping of each construct depends on certain requirements and a thorough assessment study is needed to determine the mapping in each case.

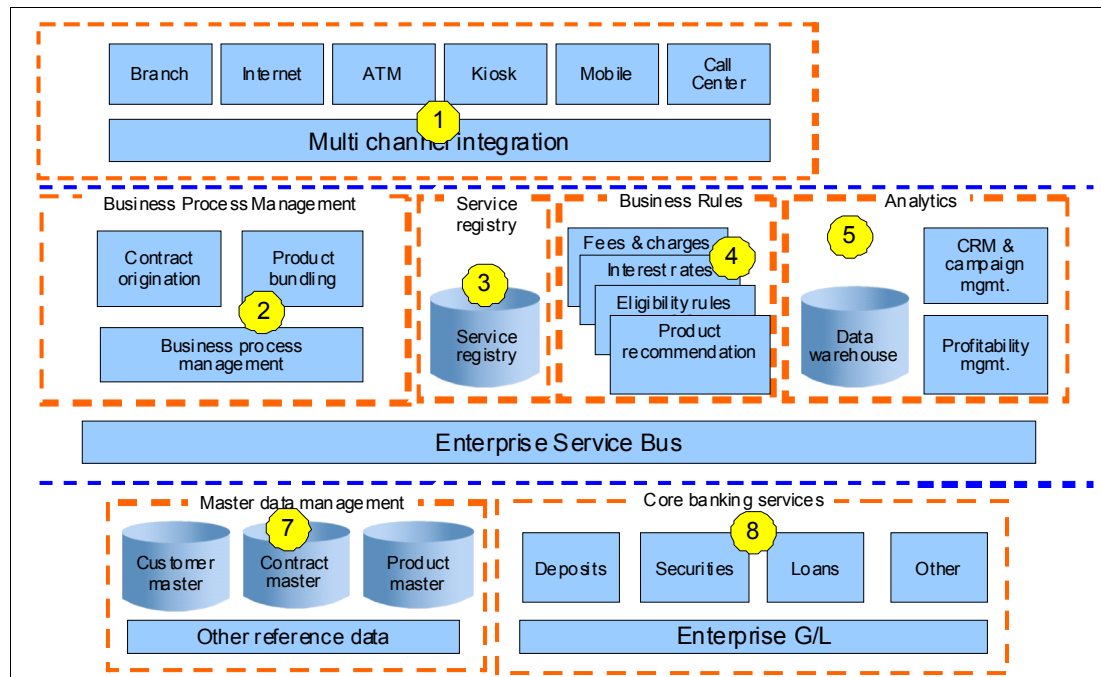


Figure 22 Core banking system blueprint, with numbers that map to zEnterprise topology

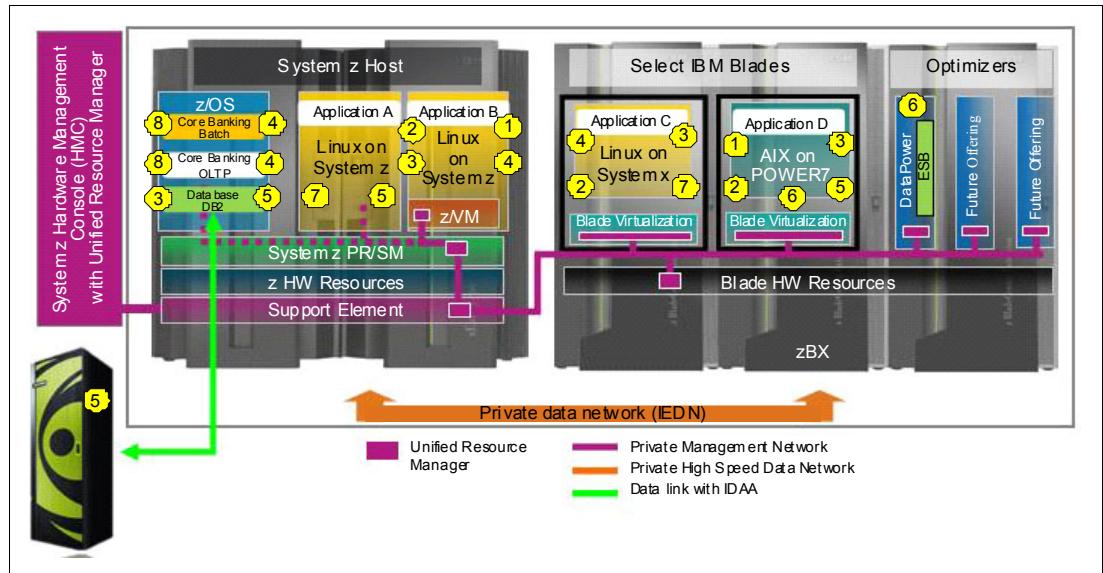


Figure 23 Possible zEnterprise topology



Next steps

If you feel the pain points of an inflexible core banking systems environment, it is a good idea to start looking at the options to transition to a modernized environment encapsulating all the benefits explained earlier in this paper. By using a combination of business architecture, IT architecture and frameworks, productive software and hardware, and a structured methodology you have all the ingredients to be successful.

Reasons for choosing IBM

IBM believes that legacy modernization is a foundation for further benefits, rather than an end in itself. This is why we advocate a progressive approach that delivers value while you are building the technology foundation. IBM has the domain knowledge, business transformation skills, technical expertise and wealth of real-world experience combined with unparalleled software and hardware platforms to help you successfully modernize your core systems. IBM is uniquely positioned to help you tackle the challenge, with capabilities that span the full spectrum of your needs, from hardware and software through business and technology consulting to outsourcing and managed services.

IBM makes extensive use of sophisticated software tools and draws on proven processes and experience with other financial institutions to analyze your existing application portfolio, business processes, application architecture, interfaces, and business rules. This enables us to identify components that are candidates for relocation, renovation, restructuring, reprioritization, or rationalization. Our industry-specific process and data models, used by hundreds of banks, drive this analysis.

Modernization is a very broad topic, encompassing elements as diverse as infrastructure, regionalization, globalization, right-sourcing skills, business operating model changes, and innovative ways to fund transformation through operating expenses instead of capital expenses. One way that banks are addressing modernization issues is by using the IBM Banking Industry Framework. This unified banking framework spans the enterprise and includes software foundations to provide end-to-end banking solutions. Framework domains, in turn, identify the technology usage patterns, software, industry extensions, and accelerators needed to address key banking pain points.

The major domains in the IBM Banking Industry Framework are:

- ▶ Core Banking Transformation Domain
Highlights how to modernize legacy applications that support core banking functions and align them with changing business needs.
- ▶ Payments and Securities Domain
Provides the middleware tooling to progressively transform payments operations to become more flexible and efficient.
- ▶ Integrated Risk Management Domain
Addresses a holistic approach for managing financial risk, operational and IT risk, financial crimes and compliance.
- ▶ Customer Care & Insight Domain
Builds a foundation for creating a single view of the customer, ultimately enabling more effective and efficient sales and service.

One of the most successful and critical modernization components is the Industrialized Software Development factory supported by IBM tools and processes. This end-to-end capability is unique to IBM and is the linchpin that ensures that IT projects deliver predictable value on time and with quality results.

IBM has successfully completed many large programs for banks of all sizes around the world and is ready to be your partner. We have all of the methods, tools and experience to renovate any bank's core banking applications with a high-quality, committed team, using proven processes to yield predictable results. Typically, we have already worked with your bank for decades and have developed a deep understanding of your applications, architecture, infrastructure, people, processes and culture.

We can also leverage the IBM global enterprise, including linking to the IBM global network of Business Partners, IBM worldwide development, and IBM Research. Additionally, we offer diversified global sourcing options delivered through strategic global delivery centers in India, China, Romania, Brazil, Argentina, Vietnam, the Philippines, and Egypt.



Summary

The extensive experience brought by IBM to core banking transformation indicates that the most successful transformation approach is a progressive one in which modernization is a process that incorporates flexibility and business acumen, and can adapt as requirements change.

Creating a componentized architecture that separates key constructs and their assets from the core transaction engine is a critical factor in achieving a successful transformation. Such a scenario makes the architecture the central concern and allows a bank to benefit from the necessary flexibility and efficiency. After the core architecture is established, the bank can address each requirement and modification on a case-by-case basis by choosing from custom and packaged options.

Further, the core transformation method outlined in this IBM® Redguide™ publication allows a bank to make deployment decisions based purely on business benefits by using a progressive program that delivers value at each step of the process, and every step can be tailored to keep pace as needs and requirements change. Finally, this method ensures that the overall transformation process remains in line with evolving business objectives.

Other resources for more information

For more information about this topic, consult the following resources:

- ▶ Landing page for IBM core banking transformation solutions:
<http://www.ibm.com/software/industry/banking/transformation.html>
- ▶ Banking Industry Framework (of which core banking transformation is a part):
<http://www.ibm.com/software/industry/banking/framework/index.html>
- ▶ Brochure on the IBM Banking Industry Framework:
ftp://public.dhe.ibm.com/software/industries/frameworks/pdf/BZD03001_BANKING.pdf
- ▶ IBM Banking Industry Framework announcement:
<http://www.ibm.com/press/us/en/pressrelease/28401.wss>

- ▶ IBM Redpaper™ *Case Study: SOA Banking Business Pattern*, REDP-4467
- ▶ Banking Transformation Workbench video:
http://www.ibm.com/common/ssi/cgi-bin/ssialias?infotype=SA&subtype=PS&appname=SUGE_BZ_CZ_USEN&htmlfid=BZV03001USEN&attachment=BZV03001USEN.WMV
- ▶ Landing page for solutions for banking on System z:
<http://www.ibm.com/systems/z/solutions/banking.html>
- ▶ White paper *Managing 21st Century Business and Technology Innovation: Core Banking Transformation with System z*, ZSW03011USEN:
<http://public.dhe.ibm.com/common/ssi/ecm/en/zsw03011usen/ZSW03011USEN.PDF>

The team who wrote this guide

This guide was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO).

Alex Louwe Kooijmans is a Senior Architect at the Financial Services Center of Excellence at IBM Systems & Technology Group. Prior to this position he spent almost ten years in the International Technical Support Organization leading IBM Redbooks® projects, teaching workshops, and running technical events with a focus on using the IBM mainframe in new ways. Alex has also been a Client Technical Advisor to various banks in The Netherlands, and has worked in various positions in application development. His current focus is on modernizing core banking systems and the role of IBM mainframe technology.

Rishi Balaji is an IBM accredited Application Architect with eleven and a half years of experience in the design and development of IT solutions. His expertise ranges from product development to consulting and asset development in areas such as service-oriented architecture (SOA), case management, and business intelligence using JEE and related technologies. He is a contributing author at IBM developerWorks® and has also co-authored a book on asset-based development. Rishi currently works at the Global Business Solution Center as an application architect for the banking industry.

Yasodhar Patnaik is the Chief Architect for husbanding Industry Framework in the IBM Software Group. In his current role, he is responsible for thought leadership, roadmap, and vision for the framework. Yasodhar is also responsible globally for engaging with banking clients as a subject matter expert in banking solutions and technologies.

Saket Sinha is the worldwide core banking transformation leader for IBM. He focuses globally on core system renovation and transformation opportunities for IBM banking clients. Prior to this, Saket was a subject matter expert for the Strategy and Change practice for the Global Banking and Financial markets. He was responsible for providing thought leadership, critical insights, client engagement support, and advisory services to both banking and financial market clients.

Thanks to the following people for their contributions to this project:

Tom Seevers, Ph.D
 IBM Fellow, Vice President of Technology and CTO
 IBM Financial Services Sector

Ken Muckenhaupt, Anna Wang
 IBM Systems and Technology Group, Financial Services Center of Excellence

Irena Slywkanycz and Alfred Schwab
International Technical Support Organization, Poughkeepsie center

Michel Van der Poorten
Client Executive, Financial Services Sector, IBM Belgium

Marcel Vonk
IBM Solution sales executive, ABN-Amro account

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.




Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>



The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

BladeCenter®	IMS™	System z®
CICS®	InfoSphere®	Tivoli®
DataPower®	Rational®	WebSphere®
DB2®	Redbooks®	z/OS®
developerWorks®	Redguide™	z/VM®
IBM®	Redpaper™	zEnterprise®
ILOG®	Redbooks (logo)  ®	

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.