7 Kasım 2012 - Çırağan Palace Kempinski

IBM Connected 2012 Istanbul

Learn. Collaborate. Innovate.

Progressive Transformation Approach

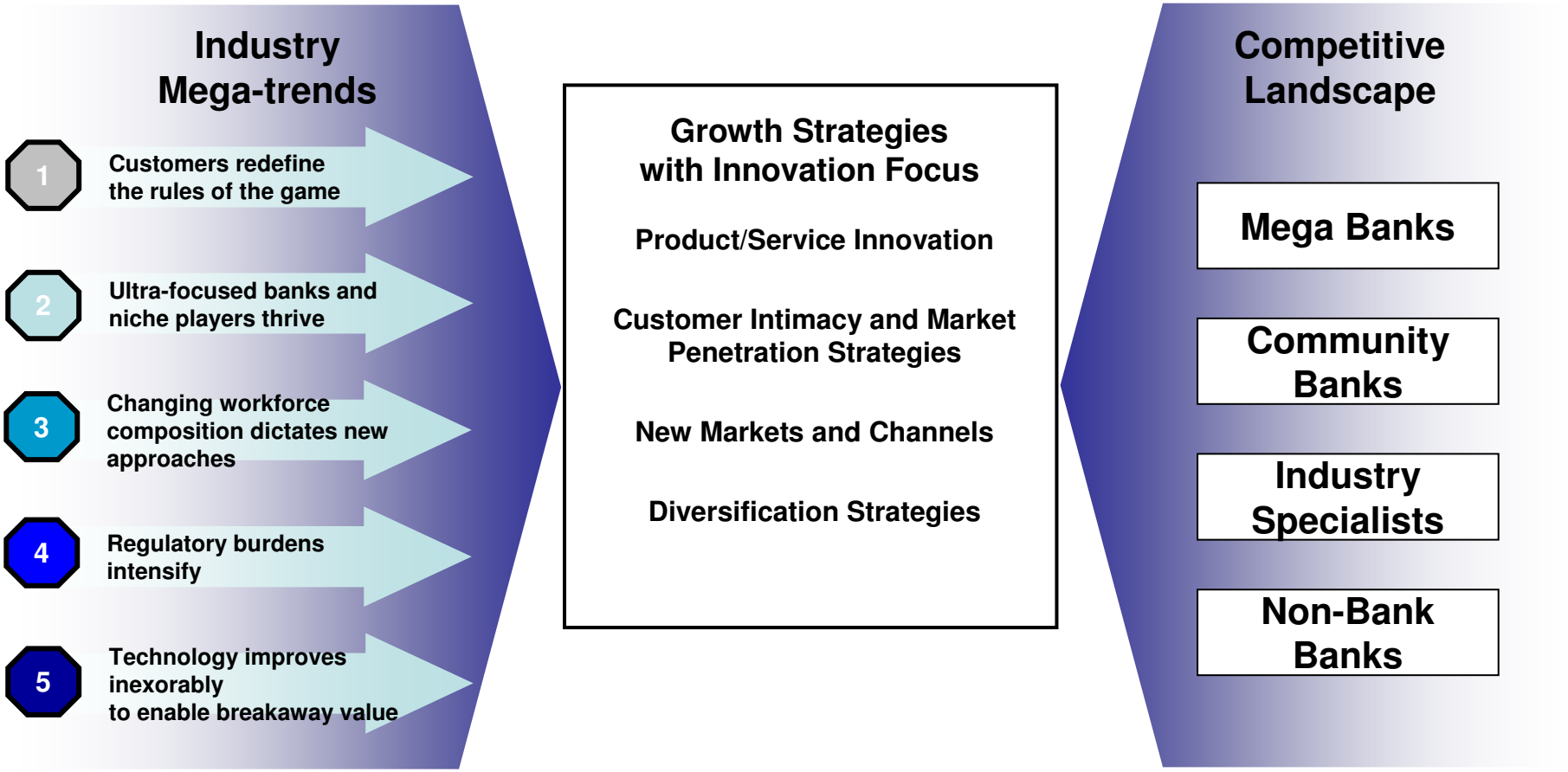
Adnan Şenyurt
Application Innovation Services
IBM Global Services

Topics for today's discussion

- Market Snapshot
- Core Banking Modernization Overview
- The Need For Architectural Discipline in Banking Modernization
- IBM's Core Banking Transformation Framework (CBTF)
- Approach and Methodology
- Case Studies

Banks will be driven to in four strategic areas as industry trends become more pronounced and as competitive forces intensify

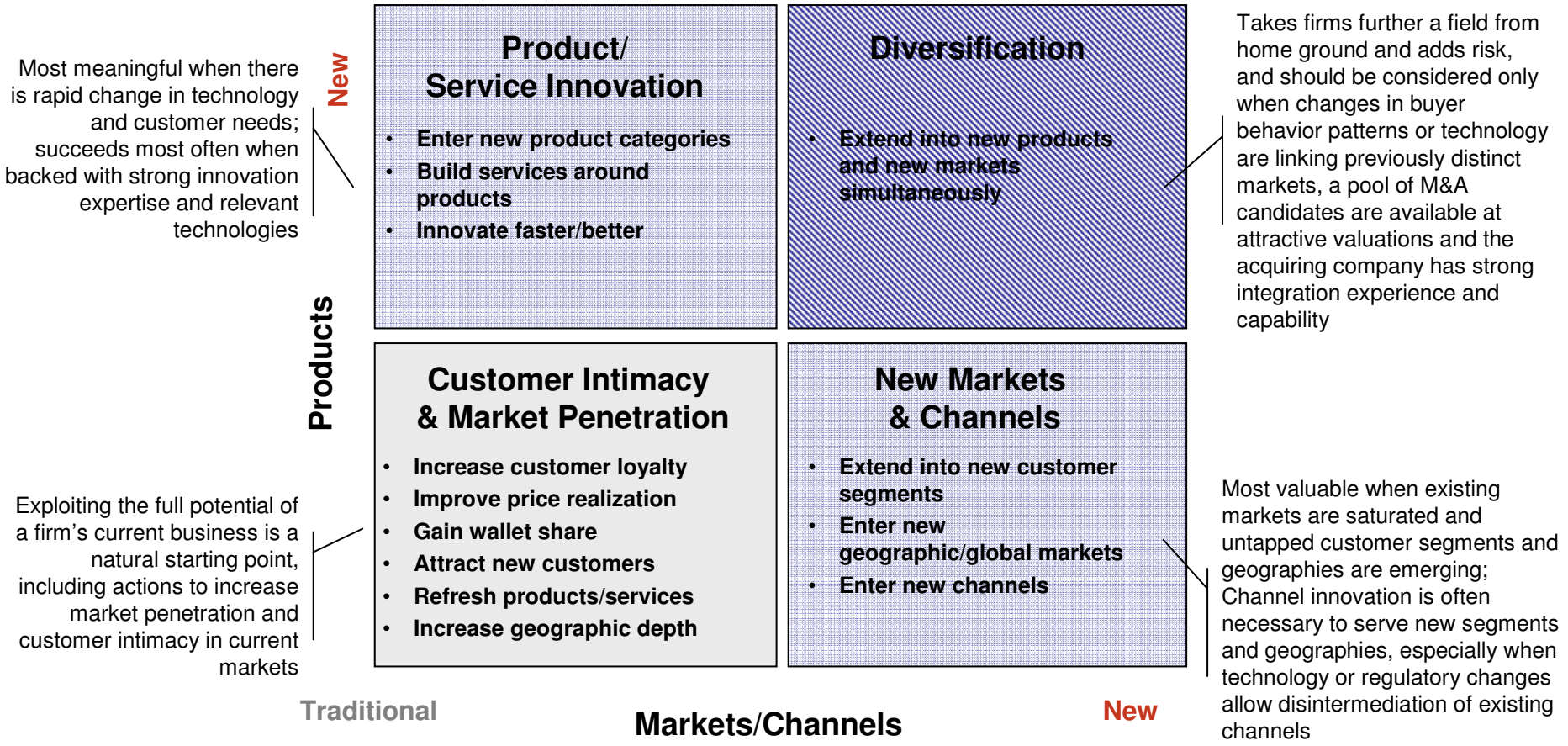
Marketplace Conditions Making Innovation Imperative



Source: IBM Institute for Business Value

Banks have a number of different options to consider as they evaluate which opportunities for innovative growth are right for them

Opportunities for Innovative Growth



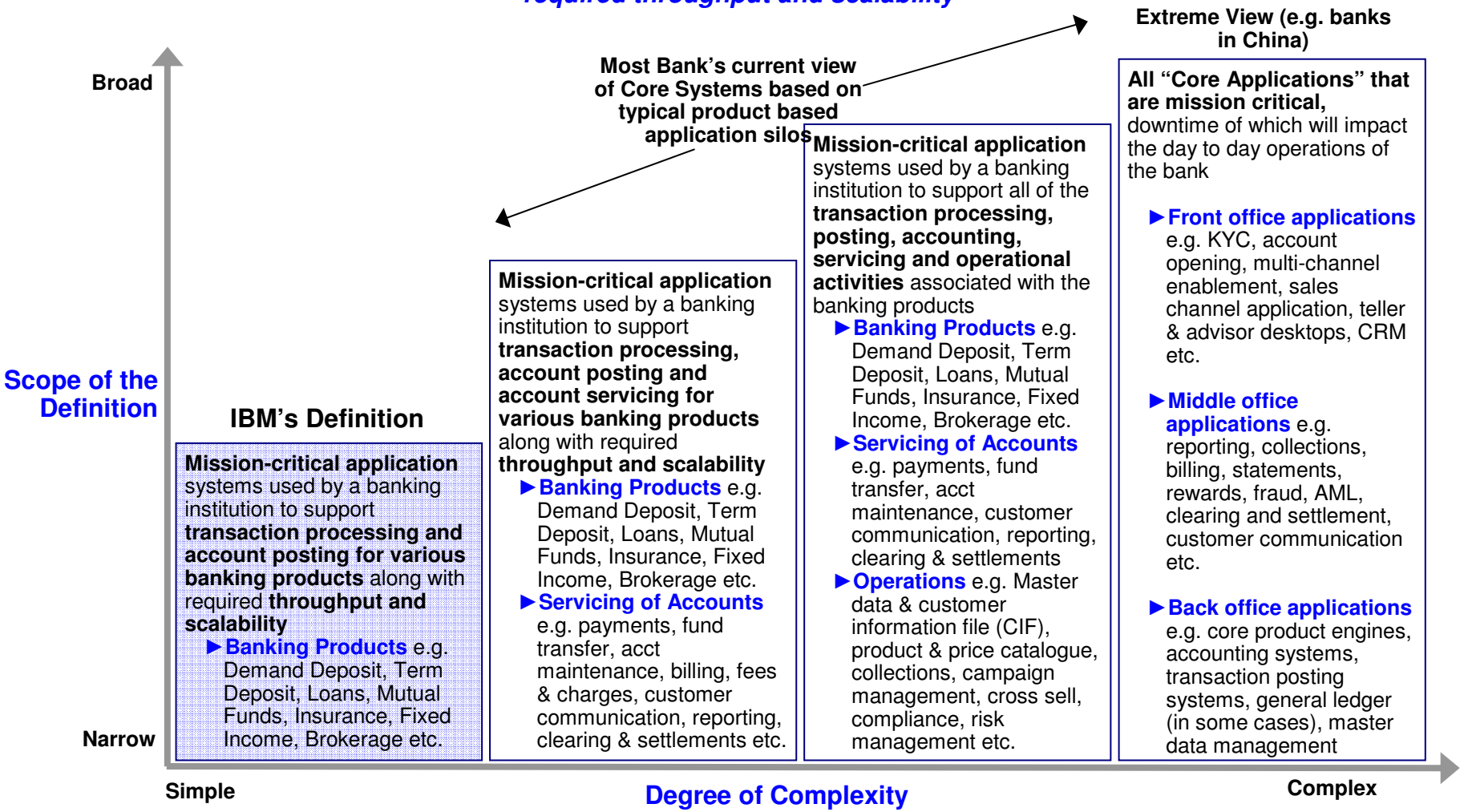
Source: Adapted from The Growth Triathalon, IBM Institute for Business Value

Topics for today's discussion

- Market Snapshot
- Core Banking Modernization Overview
- The Need For Architectural Discipline in Banking Modernization
- IBM's Core Banking Transformation Framework (CBTF)
- Approach and Methodology
- Case Studies

What is the definition of Core Banking Systems?

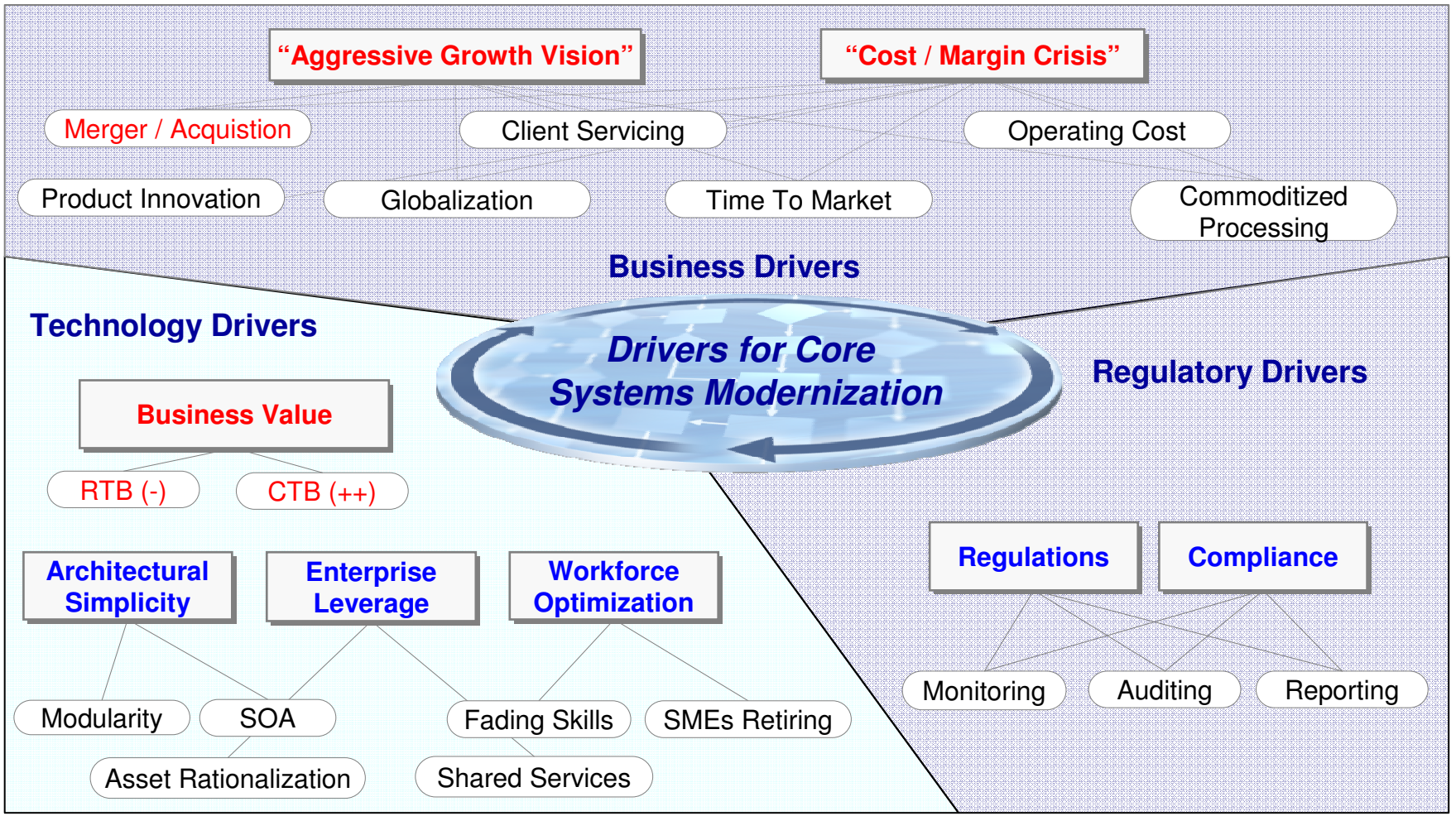
IBM's view of "Core Banking Systems" definition includes all mission-critical application systems used by banking institution to support transaction processing and account posting for various banking products along with required throughput and scalability



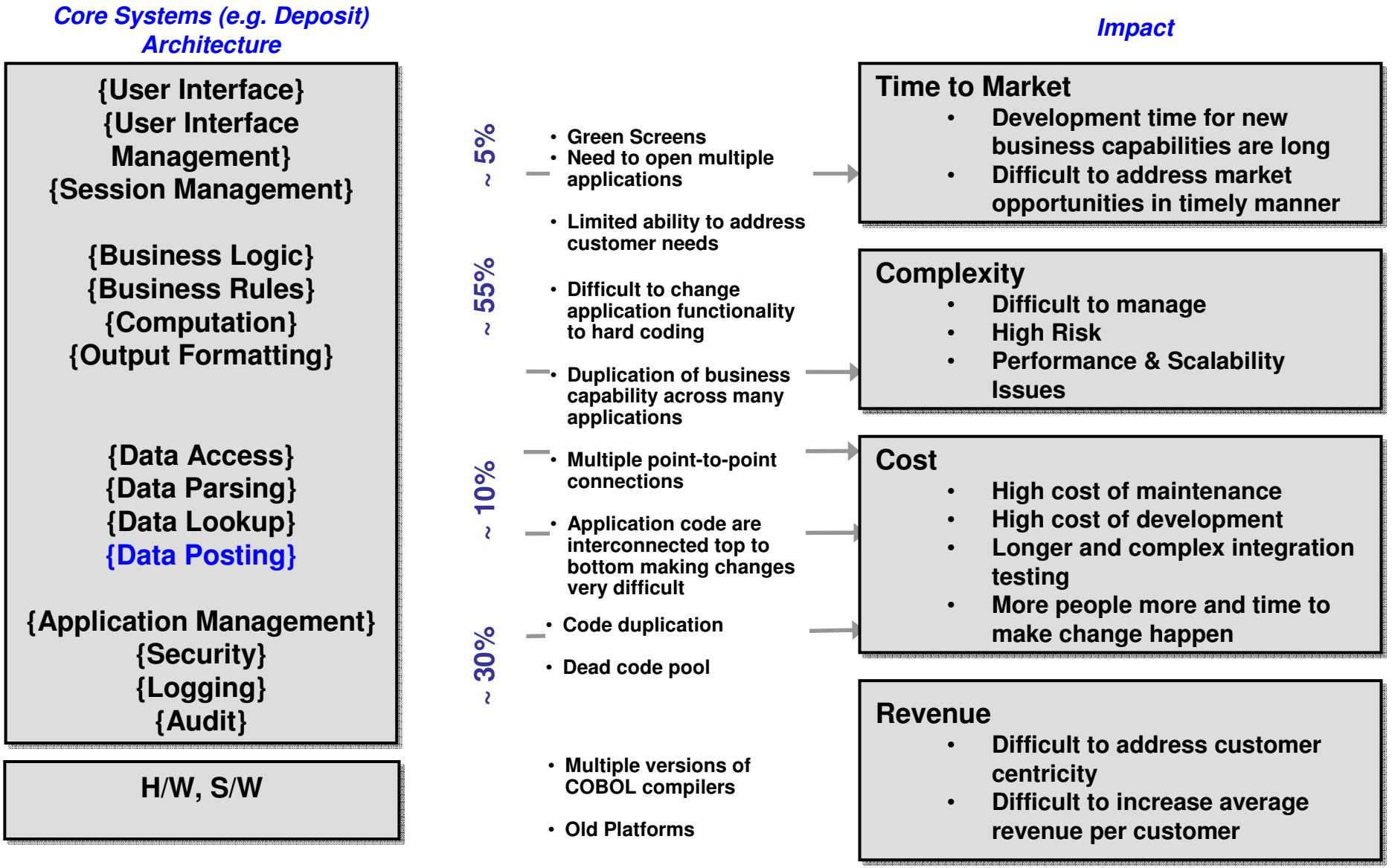
(1) In this document "Core Banking" and "Core Systems" will be used interchangeably with the same contextual meaning

Theme: Determine the main transformation themes that applies to your bank and express your case within the themes boundary

Banks are looking at core systems modernization in response to specific business, technology and regulatory drivers

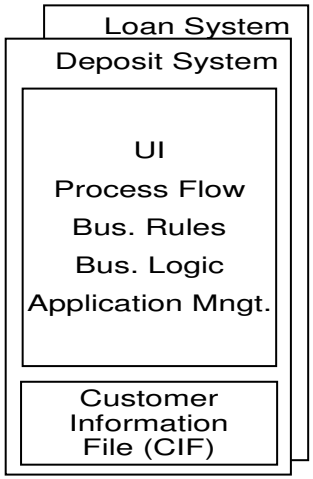


So what aspects of core systems really manifests the problem?



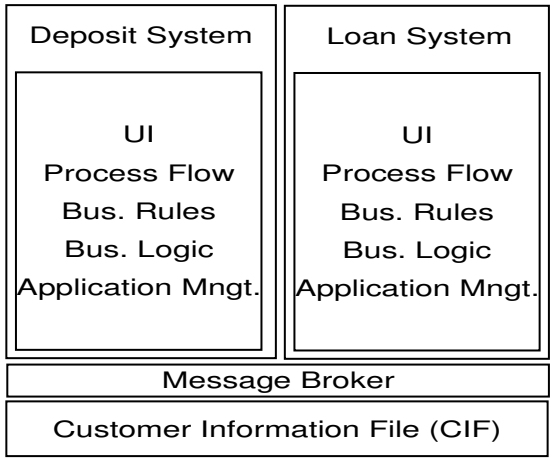
Package based core banking systems have evolved over time, but yet to provide solutions in line with contemporary architecture principles..

1st Generation



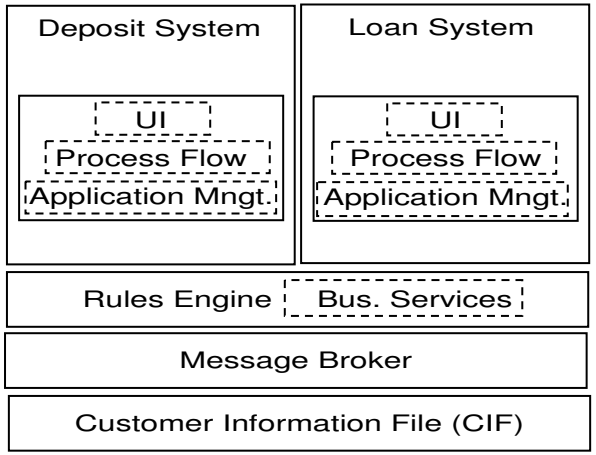
- **Embodied the same siloed product based system with little or no shared system capabilities**
- **Used separate customer information file but tightly coupled with application**
- **Requires huge customer integration and customization effort**
- **Still inflexible in meeting newer business requirements**

2nd Generation



- **Still embodied the siloed product based systems structure with little or no shared system capabilities**
- **Started using single CIF across multiple product systems (deposits, loans etc.)**
- **Started using proprietary message broker to ease integration challenges**
- **Customization and system integration with rest of the banking system increased with addition of new architectural constructs**

3rd Generation and Beyond



- **Faced with persistent demand, package systems are trying to become as modular as possible**
- **To maximize immediate and long term revenue, package vendors are interested in pushing their whole package and not small modules**
- **They want to maximize downstream application enhancement, integration and maintenance revenue stream and often will down price their initial license and fees**

What does Core Banking Modernization mean to Banks?

Core Banking Modernization is more than just modernizing or replacing the core systems

To Address New / Changed Banking Processes & Regulations	To Transform Old/Legacy Monolithic Applications	To Exploit New Architectural / Technological Constructs	To Exploit New Hardware & Software Capabilities
Business Process Modernization	**Application Modernization**	**Architecture Modernization**	**Platform Modernization**
• Cater to changing business needs • Consolidation across business lines	• Accommodate new business requirements • Address complexity and efficiency challenges	• Adopt a more modular architecture • Separate architectural concerns	• Leverage new H/W & S/W capabilities • Migrate platform for cost / performance reasons

Core Banking Modernization

When banks embark on core banking system modernization they may include one or more areas in their modernization focus

(1) In this document "Core Banking" and "Core Systems" will be used interchangeably with the same contextual meaning

What approaches should be pursued by banks for modernization?

Core Banking System Modernization Approach		Key Characteristics
Package Implementation	Rip & Replace	• Lack of knowledge, skill sets and documentation • More than 80% business requirements met by a package solution • Bank is willing to compromise on day 1 functionalities
	Progressively Replace	• Really old systems. Lack of knowledge and skill set • Rip and replace is not an option as bank cannot compromise on day 1 functionality offered via package • Package solution can meet upto 80% of business requirements
Re-Write		• Systems are too complex to be replaced by package solutions or undertaking legacy modernization. Best option is to re-write again • Lost documentation • Legacy analysis proves too cumbersome and expensive
Progressively Modernize	Legacy Applications	• Legacy systems old, has many cascading dependencies and contains years of organic development and differentiated capabilities that are not available in packages • Package solutions can only meet upto 50% business requirements • Banks want to undertake a step orderly modernization driven by solving priority pain points • Banks wants to harvest its legacy and the differentiated capabilities
Hybrid		• A combination of other approaches • Usually targets a contained module for package replacement e.g. trade finance if business requirements are met in package and to address time to market issues • Choosing package for new markets and modernizing legacy for core • Tries to mix best of legacy leverage with capabilities from package

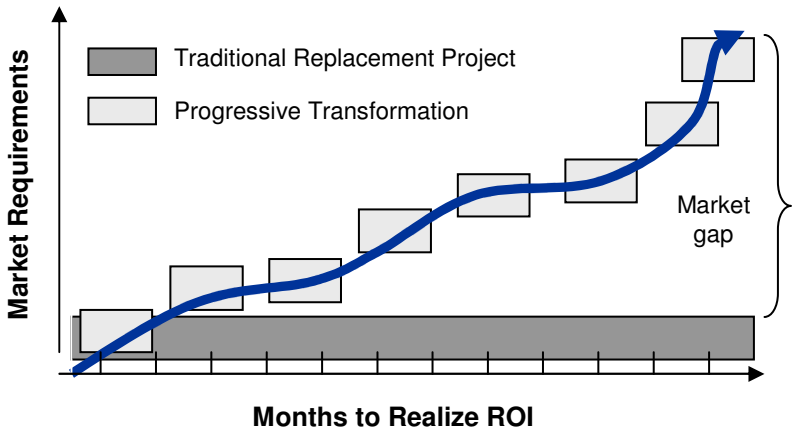
Preferred Choice By Many Banks

IBM supports the progressive modernization approach - modernizing existing assets with a well defined transformational roadmap

Progressive Renovation Focus Areas

1. **Transformational Planning**
   - ► Business and application scope
   - ► Application analysis and architecture design
   - ► Execution roadmap
   - ► Business case
2. **Foundational Architecture**
   - ► Master data
   - ► Integration layer
   - ► Service oriented design
   - ► Separation of application concerns (UI, logic, integration, data access etc.)
3. **Legacy Modernization**
   - ► Legacy harvesting of business rules, workflow abstraction and modeling
   - ► Data migration, code modernization
4. **New Capability Development**
   - ► Model driven development
   - ► Business service modeling to support business processes
5. **Execution through a modernization factory**
   - ► Tools, assets, accelerators and methods
   - ► Software modeling, development, and testing
6. **Governance and change control**
   - ► Identify, design, develop, publish and maintain services
   - ► Program Management
   - ► Change and configuration management

Traditional vs. progressive approach



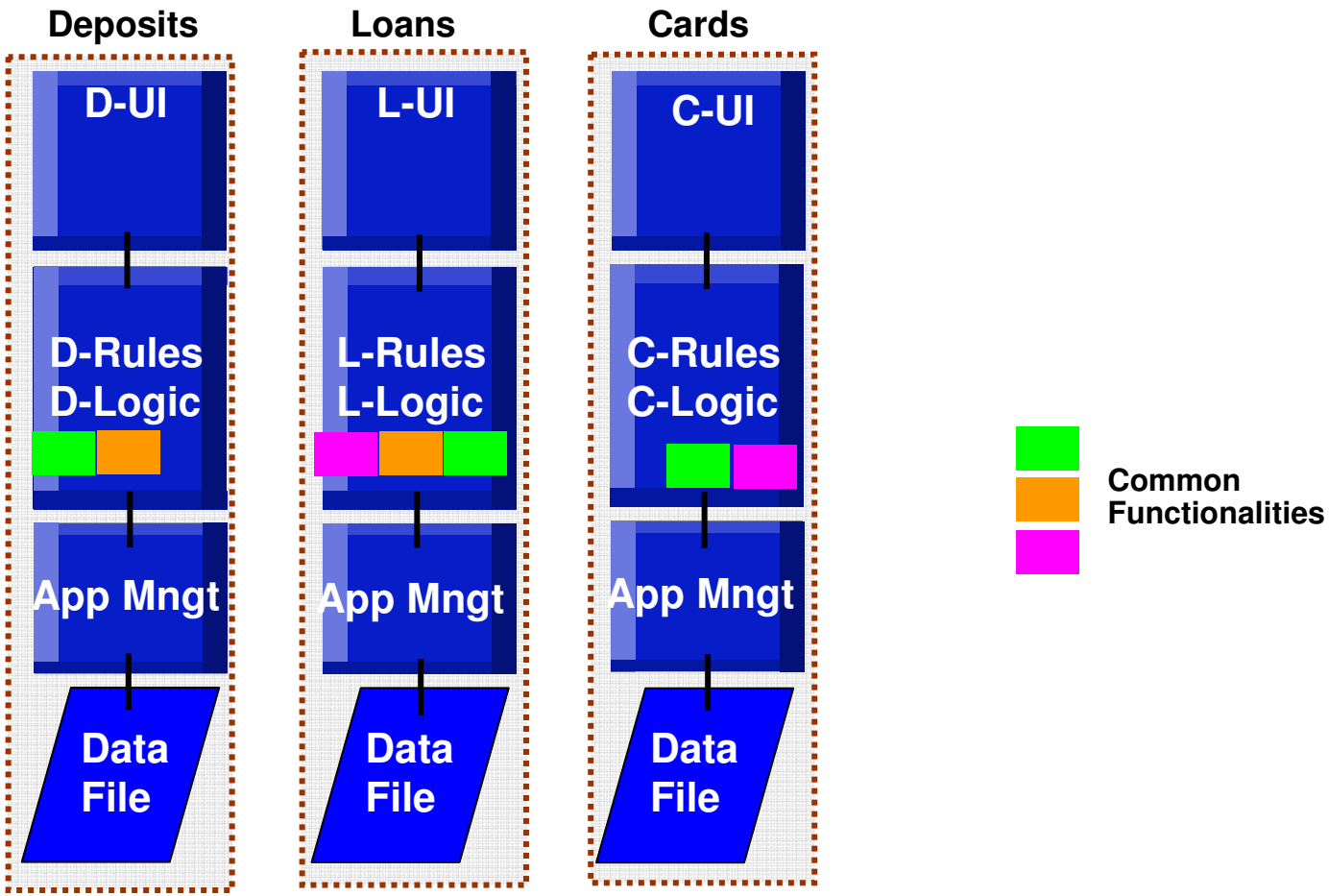
- ➲ **Progressive Renovation is an architecture led strategy**
- ➲ **Integration middleware, master data and service oriented design are key elements of progressive renovation approach**
- ➲ **The modernized environment could be a mix of old and new co-existing together**
  - ➲ **Old legacy system codes – wrapped as services**
  - ➲ **Old legacy code transformed into new code base**
  - ➲ **Newly developed capabilities**
  - ➲ **3rd party integrated capabilities**

Topics for today's discussion

- Market Snapshot
- Core Banking Modernization Overview
- The Need For Architectural Discipline in Banking Modernization
- IBM's Core Banking Transformation Framework (CBTF)
- Approach and Methodology
- Case Studies

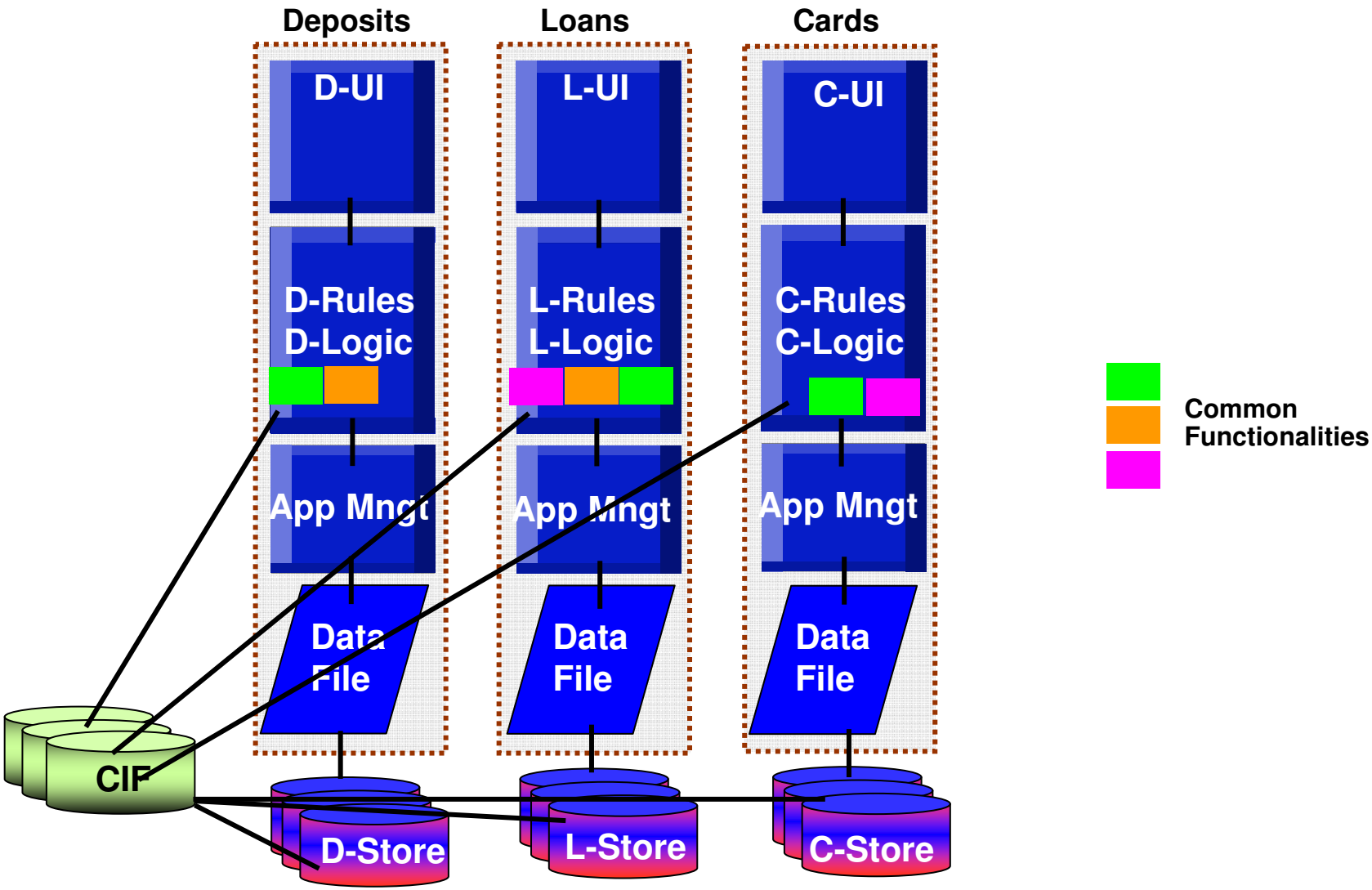
Typical view of legacy based core banking applications: 30 years back the view was simple and evolving

Siloed Applications

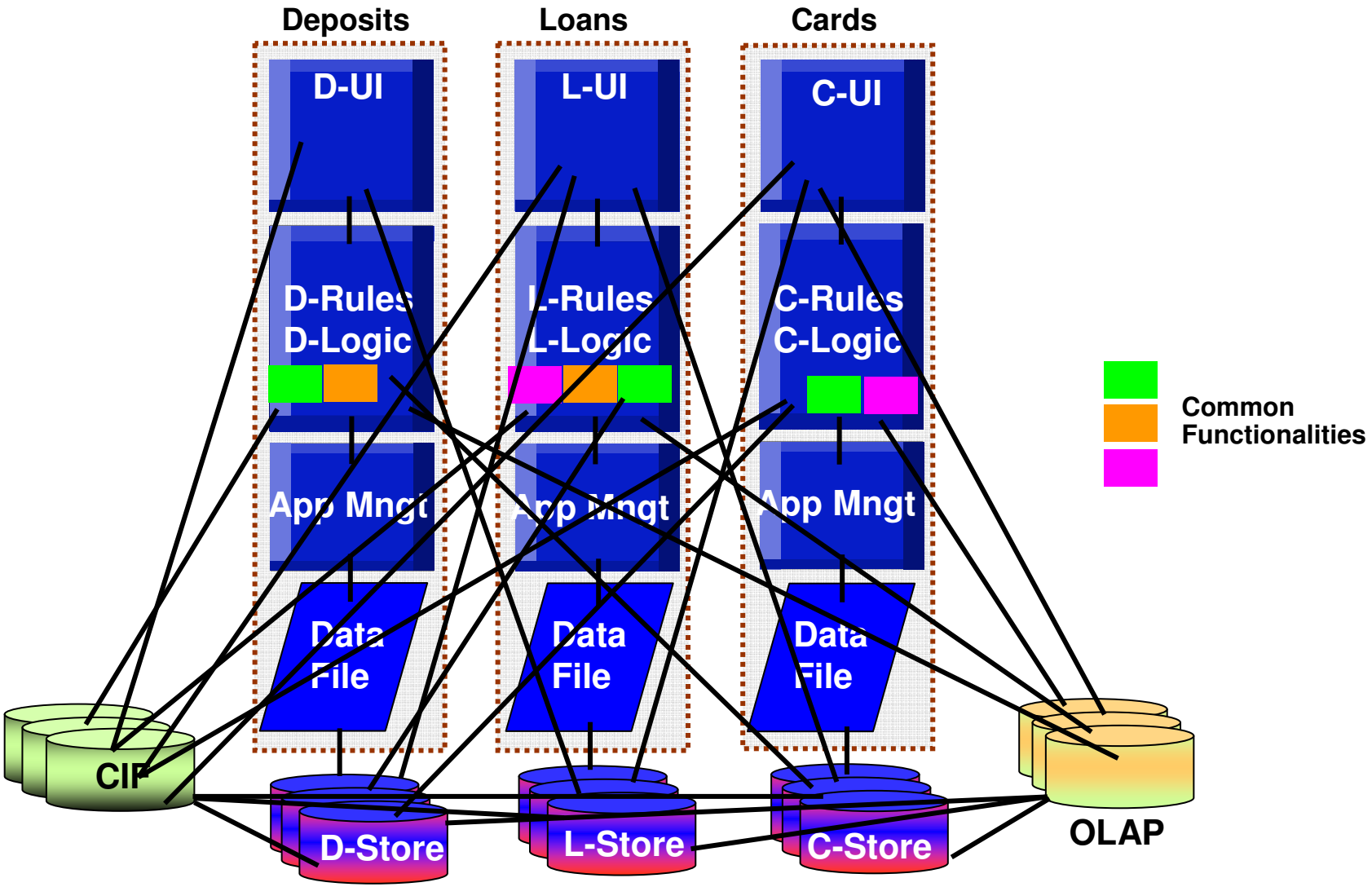


20 years back with the explosion of data, new constructs (relational databases) were introduced to the core banking systems

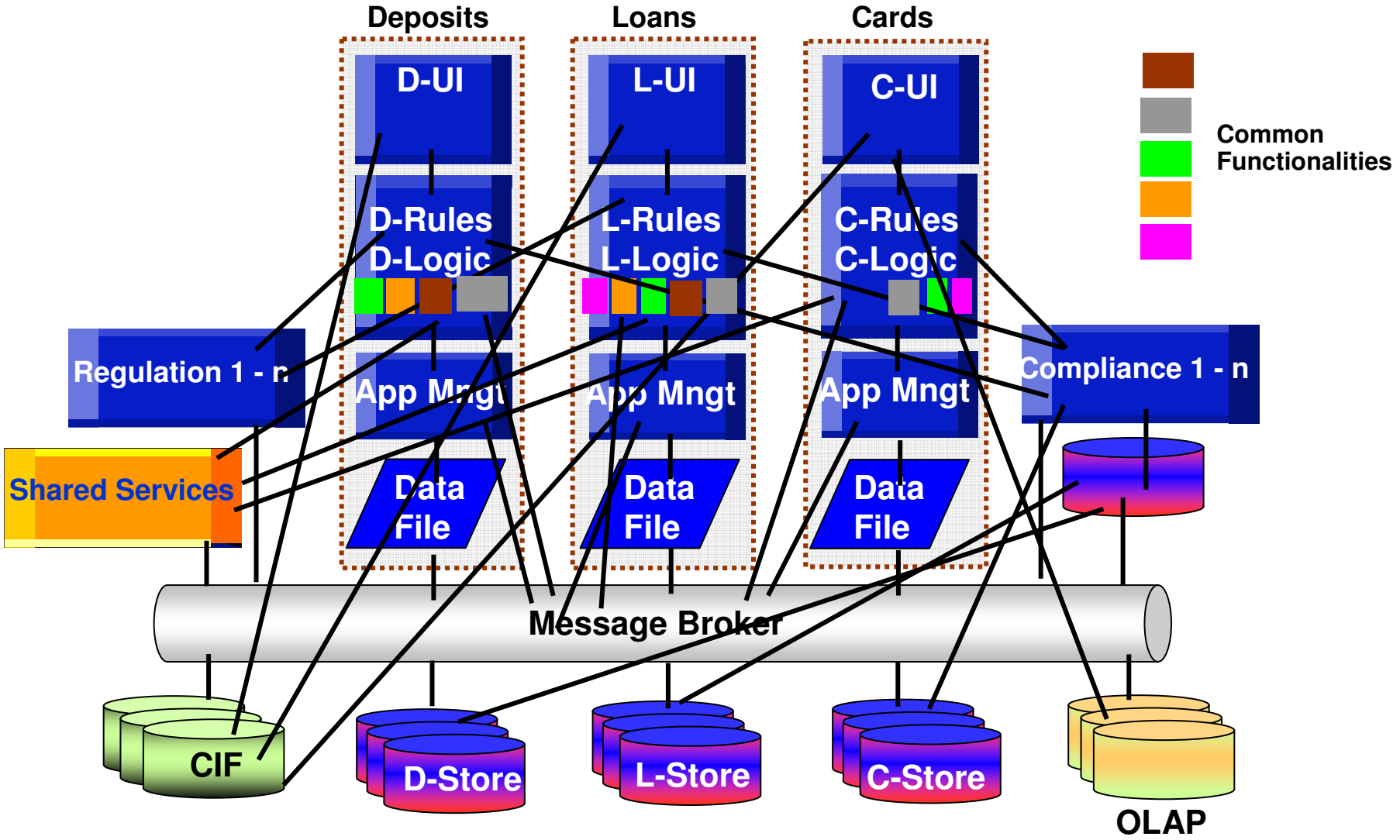
Siloed Applications



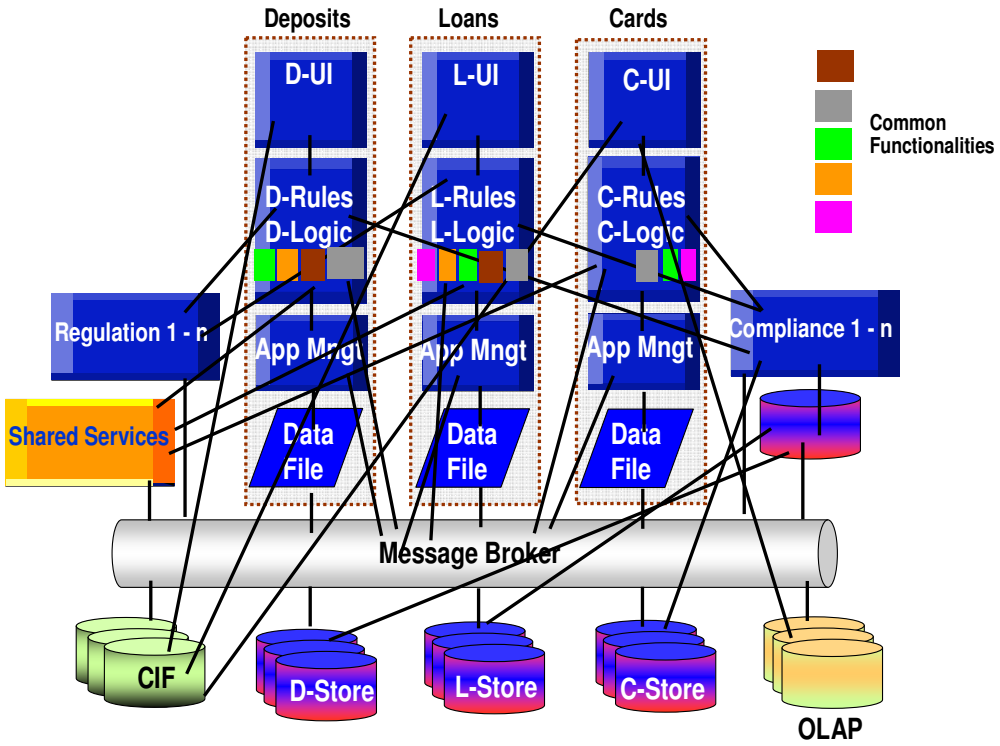
With rapid changes to the banking ecosystem, M&A, new regulations and globalization, "Customer Centricity" and hence analytics became the new mantra which forced banks to integrate tightly across product silos



As cost of new system development and maintenance increased, the evolving middleware technologies provided some initial relief in reducing point to point interfaces. However in last 10-15 years banks have seen dramatic changes resulting in increased time to market pressure forcing them to take more short cuts to meet the rapidly changing business needs.



As a result banks, today are saddled with poor architectural design and need surgical interventions to keep pace with the evolving business needs and operating models



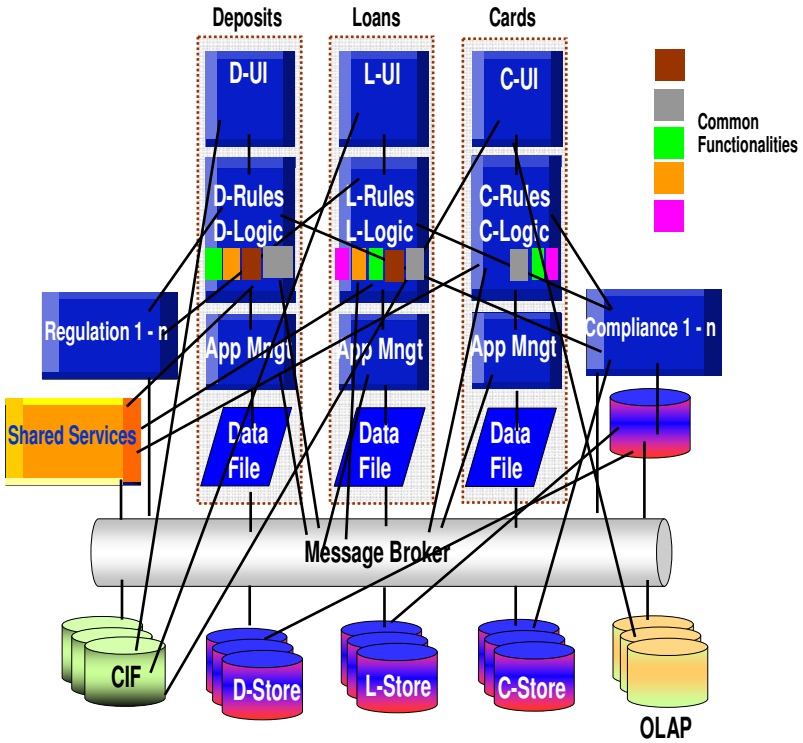
Key Goals

- **Building new business capabilities or enhancements in timely manner to drive growth and revenue**
- **Reducing point-to-point Interfaces**
- **Reducing development and testing time**
- **Reducing TCO of systems**
- **Enabling sharing of capabilities across business lines at enterprise levels**
- **Integrating data across disparate views**
- **Making systems more modular**
- **Enforcing modular design and standards for development**
- **Solving ageing workforce and skill issues**

Time to market and high cost of development and enhancements are the two most pertinent problems that is plaguing banks IT. Over the last 5 years testing cost and time has almost increased by 4 times

Transforming the bank from current to "to be" architectural design is usually done in multiple ways

"Current"

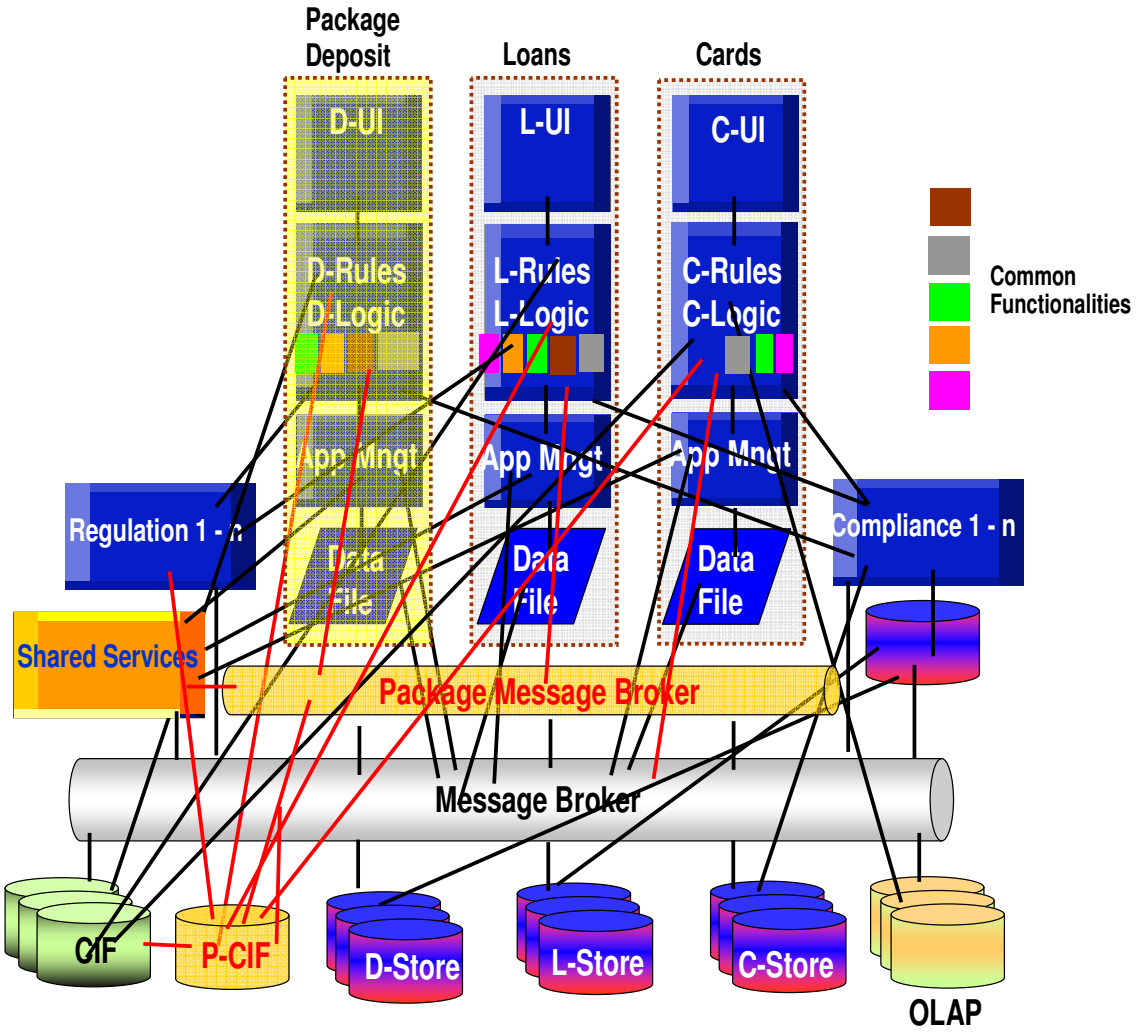


"To Be"

- **Package Led Approaches**
  - **Big Bang Approach**
  - **Progressive Package Implementation Approach**
- **Architecture Led Approaches**
  - **Progressive transformation of legacy applications**
  - **Replacement of legacy with packages, where suitable, that meet architectural discipline**

Many banks are not aware of possibilities from an architecture led approaches due to lack of knowledge of technological capabilities and internal will to undertake what seems to be very complex and time consuming task

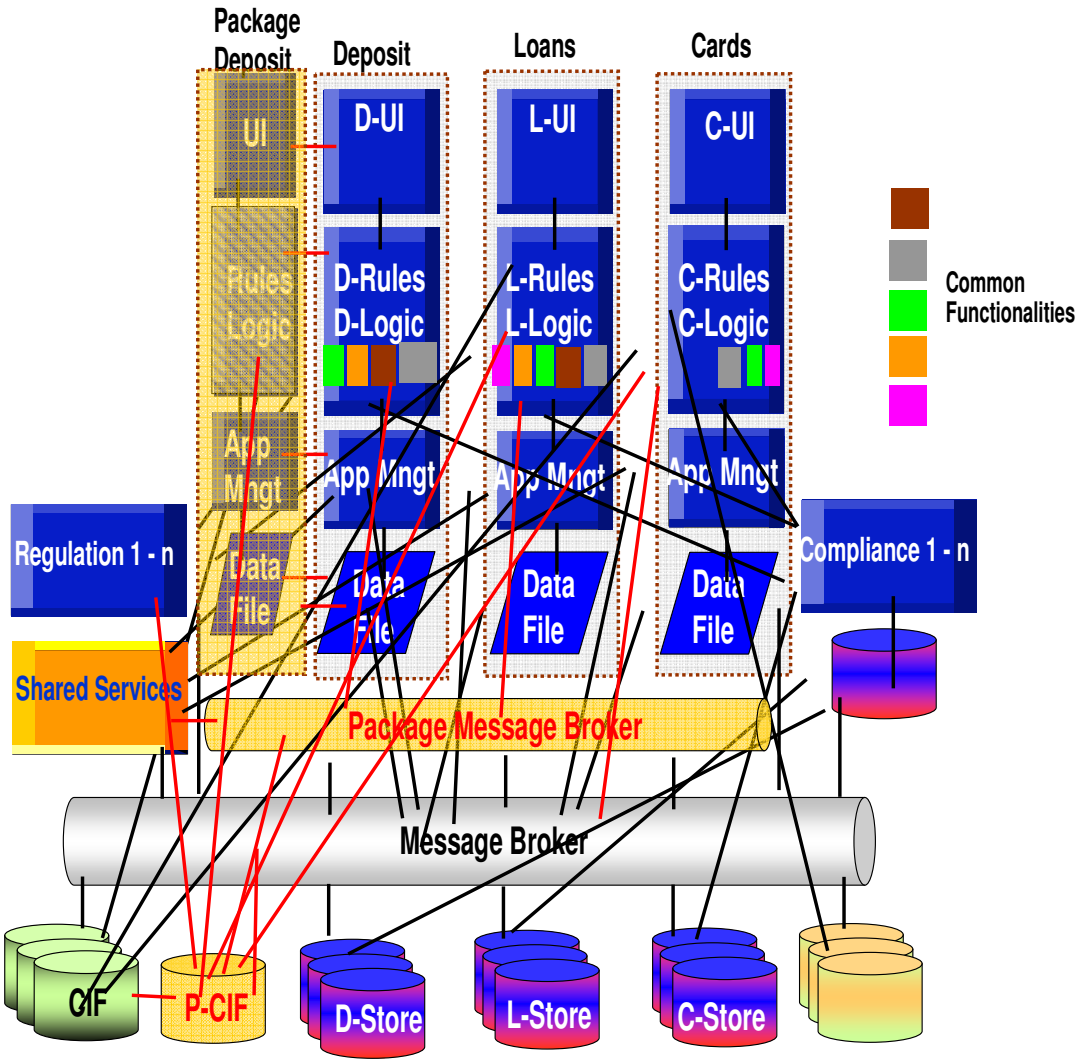
Big Bang Method



"Lesson's Learned

- Package replacement will invariably lead to increase in point to point interfaces
- Future development, will invariably have to deal with both legacy and package CIF's and message broker architecture resulting in more development time and testing
- Involves some very proprietary interfaces definitions that uses specific message and data formats that often prove difficult to integrate with other co-existing systems
- Banks specific needs are customized during package implementation usually as one-off implementation, thereby making the package "out of context" from future releases
- Dependencies increase on outside vendors to maintain and do new capability development
- Does not solve duplication of enterprise capabilities across business units thereby doing little to C/I ratio at enterprise level

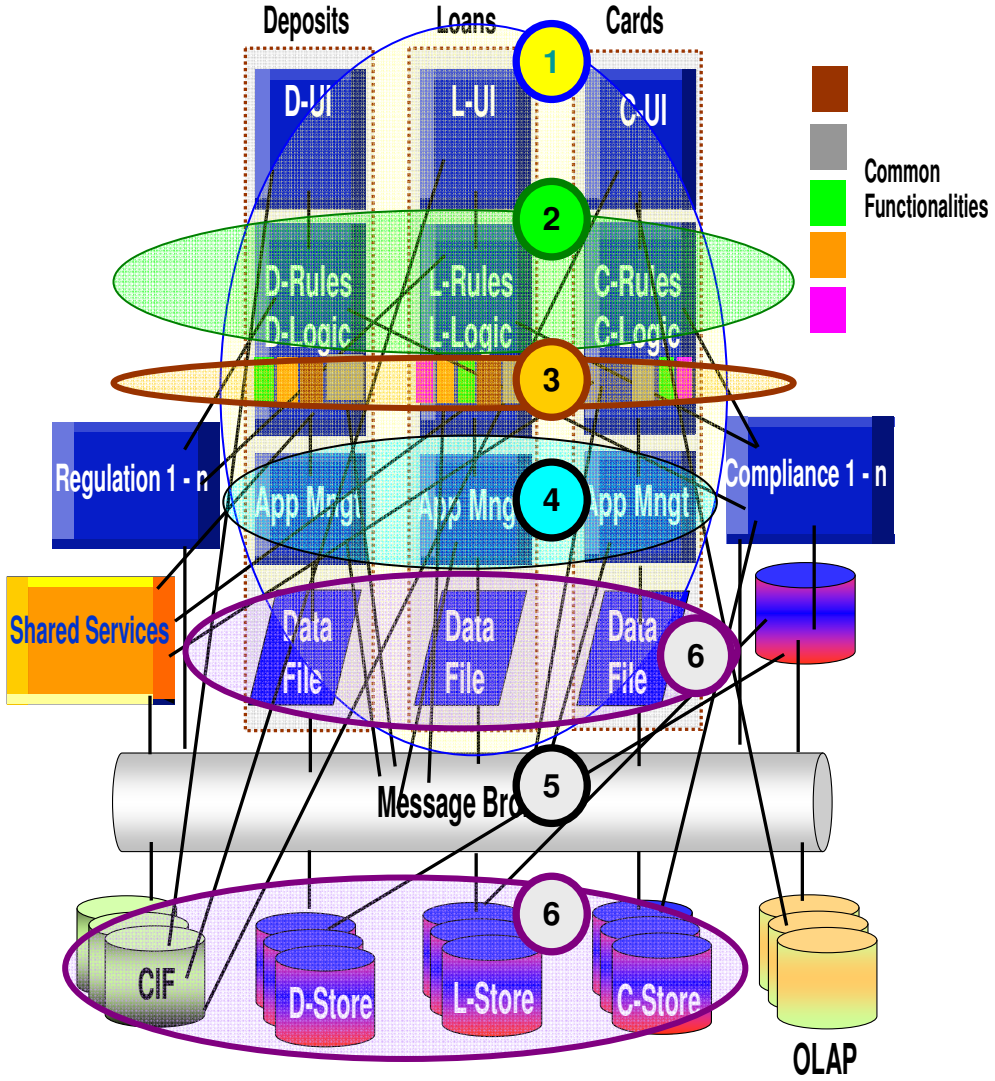
Progressive Method



"Lesson's Learned"

- **Progressive introduction of package modules is no different either. The issue with increase number of interfaces remains**
- **Progressive introduction forces to develop many throw away interim interfaces required during integration and progressive replacement**
- **Difficult to estimate the integration cost as business requirements and system dependencies are hard to establish**
- **By focusing only on the package and not focusing on maturing the core architectural discipline, which are outside the package domain, such as middleware, ESB, integrated view of data, business rules etc. progressive implementation of package often leads to serious budget overrun, delay in timeline and successful implementation**
- **While readily available business capabilities might provide a welcome change, subsequent customization and enterprise integration is always a pain**

Core Modernization – Architecture Led



1
- Modularize application components and separate architectural concerns

2
- Extract business rules and business logic and expose them as re-usable services

3
- Extract embedded services from applications out into the ESB and make it available as a shared services

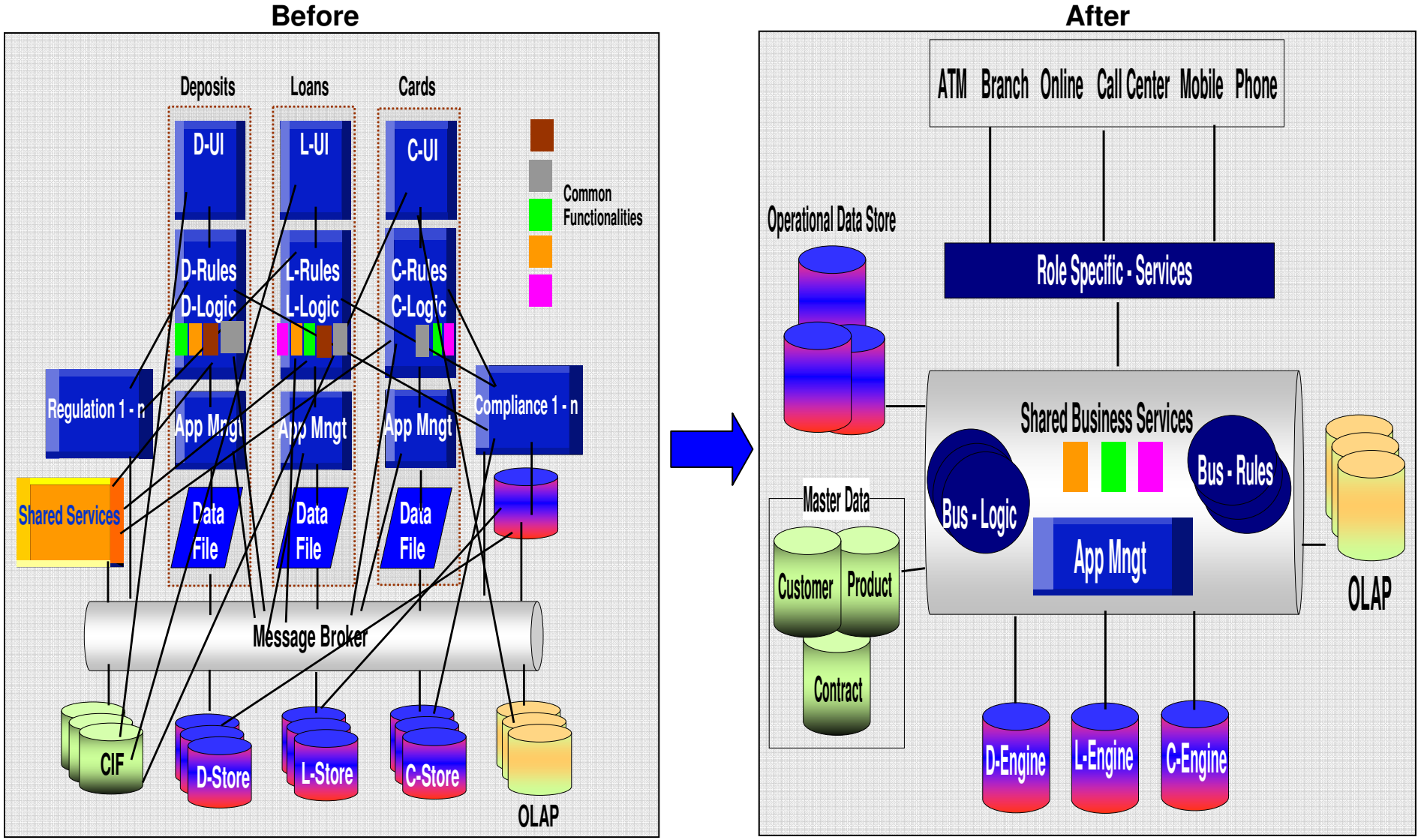
4
- Use most of application management features such as security, audit etc. from Integration bus

5
- Mature the integration layer and add enterprise services bus
- Reduce point to point interfaces

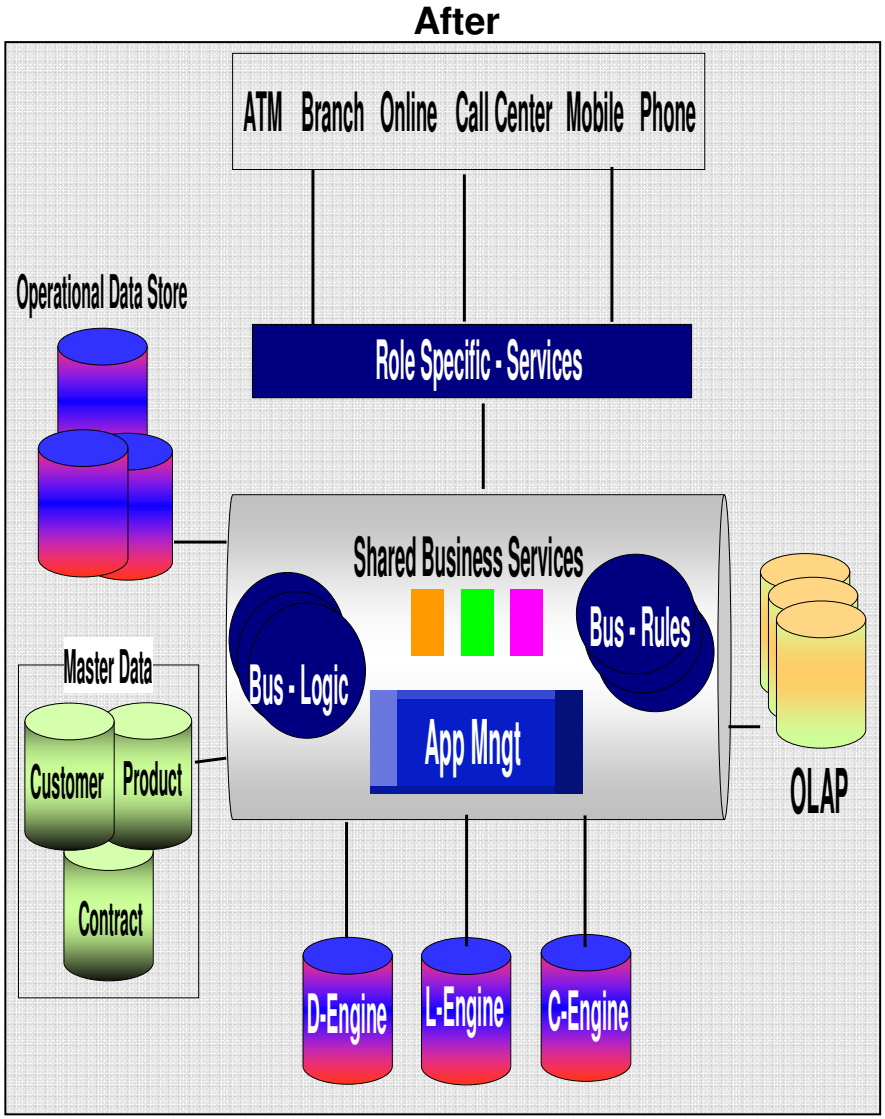
6
- Create Integrated View of Data
- Separate legacy application & data
- Use master data constructs like customer, product, contract masters

IBM' Core Banking Transformation Framework (CBTF) provides methods, tools, accelerators and templates for banks to do all the above to drive their banking modernization

Core Modernization – Architecture Led (before and after)



Many leading banks across multiple geographies are using architecture led approaches to drive banking modernization



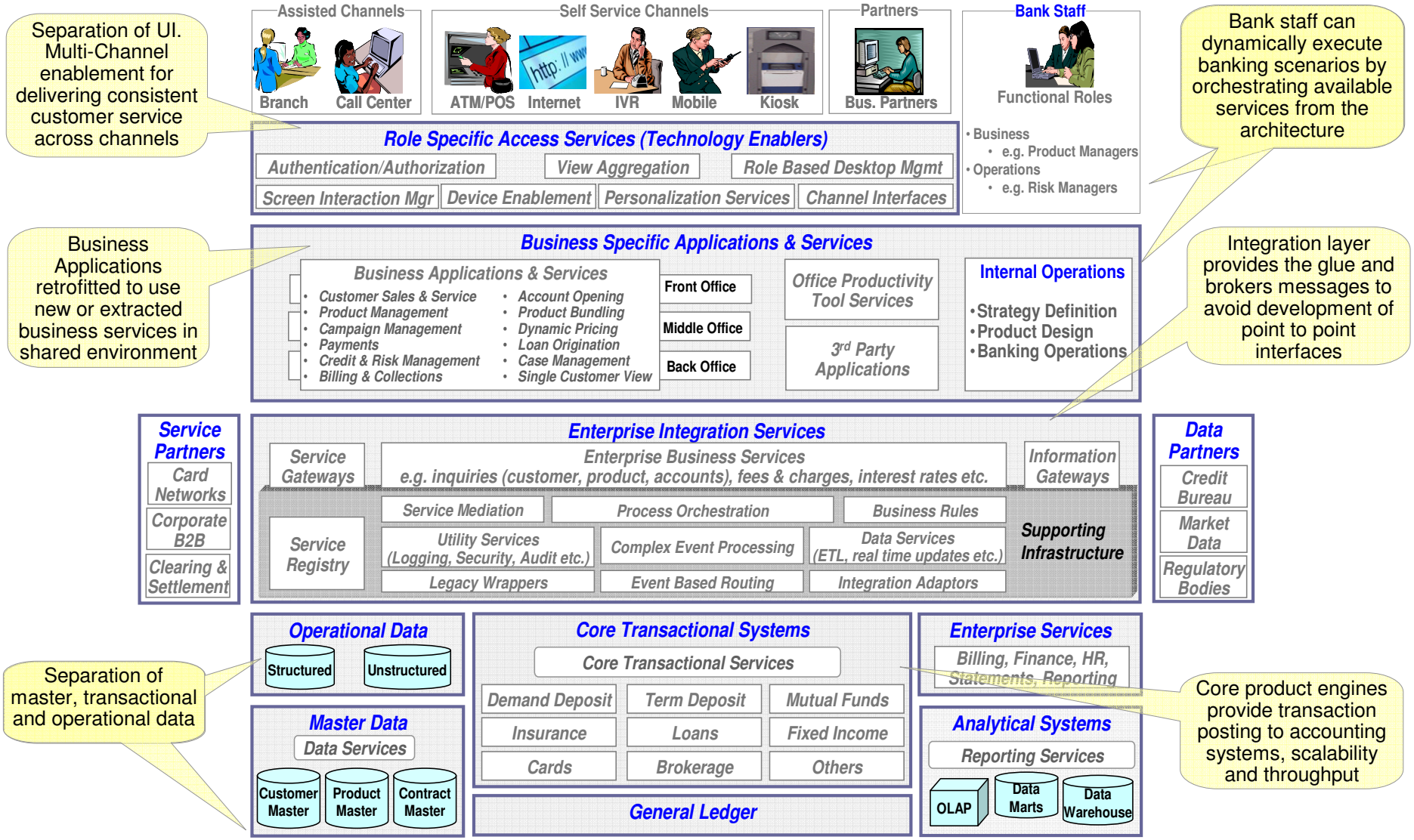
"Factoids"

- Focusing on separation of "architectural concerns" to drive modularity, simplicity and flexibility
- Increased adoption of industry reference models (process and data) to drive building core architectural constructs as the first step to transformation
- Use of middleware and ESB to achieve following primary objectives
  - Publishing services to quickly meet the needs of the business
  - Overriding underlying complexity
  - Using services to mediate legacy environment thereby allowing more time to modernize legacy systems
- Use of master data constructs to drive integrated view of data and aligning applications for the use of integrated view of data
- Big emphasis on enterprise leverage of common shared services, business logics and business rules

Topics for today's discussion

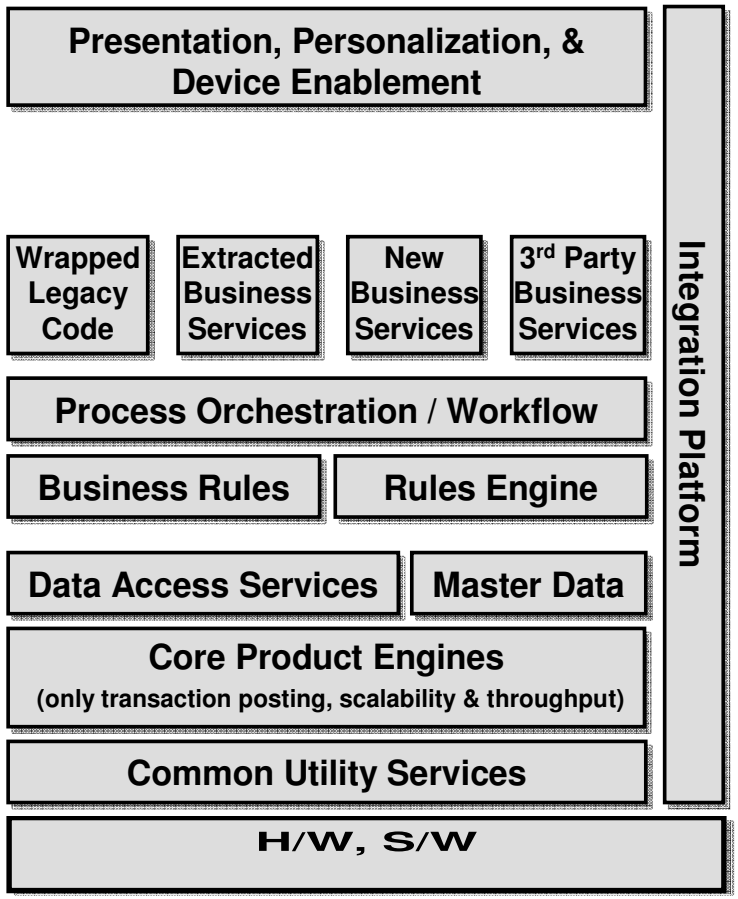
- Market Snapshot
- Core Banking Modernization Overview
- The Need For Architectural Discipline in Banking Modernization
- IBM's Core Banking Transformation Framework (CBTF)
- Approach and Methodology
- Case Studies

Our design goal is the achievement of the separation of architectural concerns for increased modularity and agility (functional & infrastructure view)



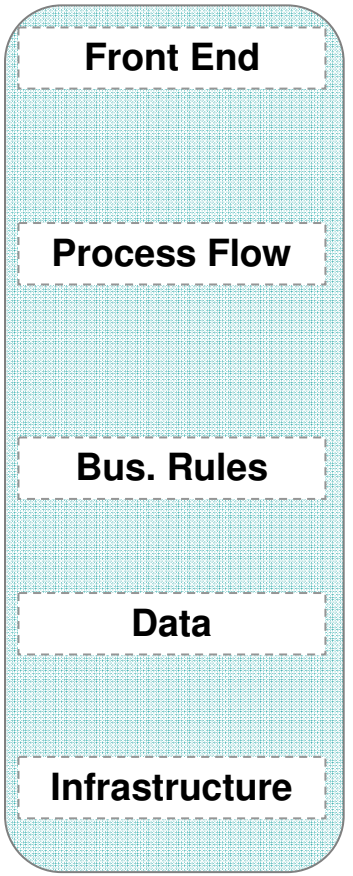
Our Core Banking Transformation Framework (CBTF) provides a more puritan PoV to drive banking transformation which strongly resonates with achieving better modularity, flexibility, agility and time to market benefits

IBM's PoV on Transformed Core Banking Systems

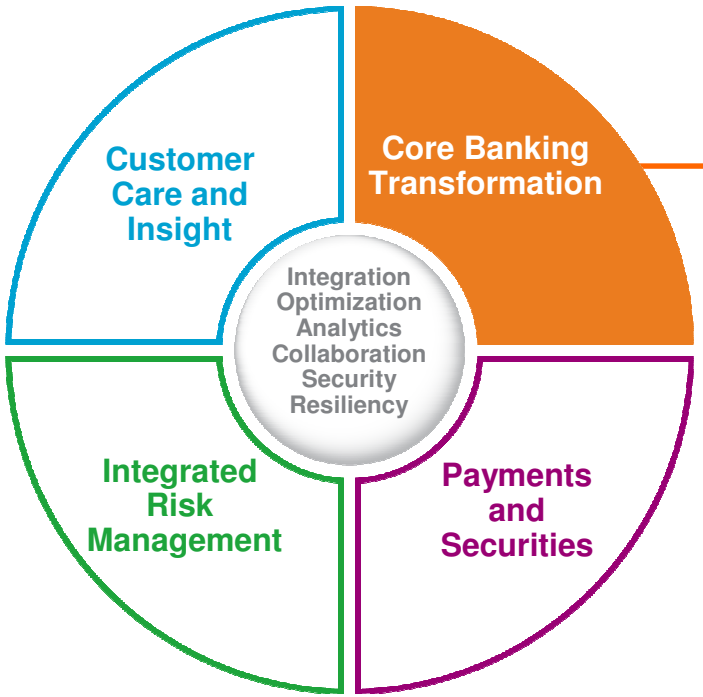


- Better customer intimacy and centricity with our **CC&I solution framework**
- Pre-defined set of business services capabilities through our **CBM/IFW**
- Pre-defined **solution templates** to drive key business outcomes such as account opening, product bundling, offers management, etc.
- Capability to externalize and centralize all business rules in easy to use **rules engine**
- Capability to externalize and centralize all data instances through an **integrated view of master data**
- Host of infrastructure capability at both H/W and S/W level to provide improved **integration**, scalability, throughput and **application management capabilities**

Next Generation Application Architecture with Separation of Concerns



IBM's Core Banking Transformation Framework (CBTF) is a combination of assets, tools, accelerators, methods and S/W products that help banks modernize and run their core banking systems



Analysis & Design	1. Process, data, & service Models 2. Master data templates for customer, product and contract master 3. Solution templates for overarching banking processes e.g. account opening, product bundling etc. 4. Reusable service components to aid application build up e.g. fees & charges, rate calculation etc. 5. Reporting templates and services for analytics 6. Common application level IT foundational services 7. Integration message models 8. Legacy analysis & discovery tools 9. Architectural work products
Develop	1. Integrated banking transformation workbench 2. SOA service components development infrastructure 3. Integration tools with support for mediation & transformation 4. Legacy extraction & transformation tools 5. Business rules development 6. Integration tools for Master Data Management
Runtime	1. Runtime infrastructure and software products 2. Banking extensions and bundles 3. System management tools

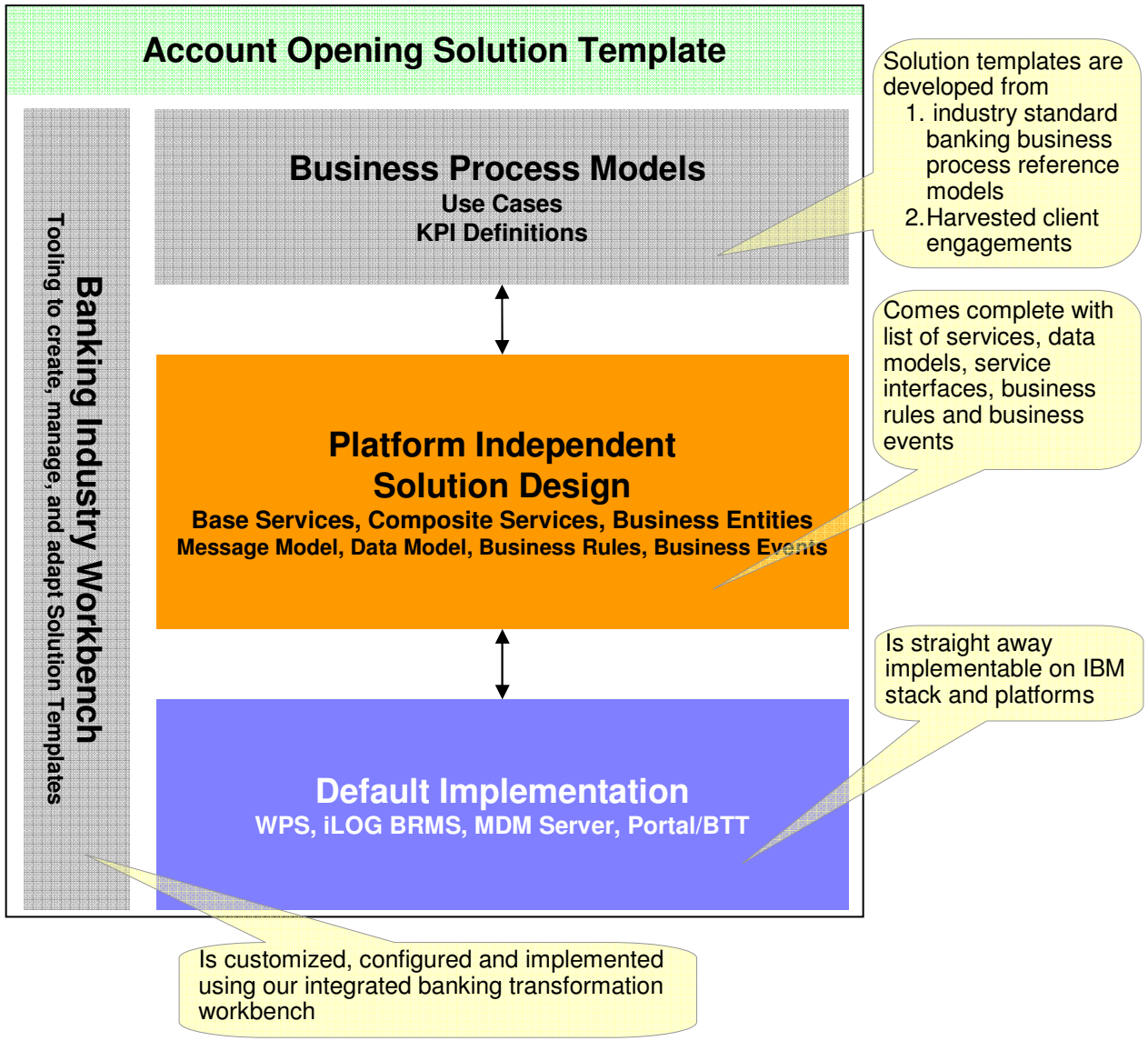
The framework gives you speed, flexibility and choice in deploying solutions while reducing cost and risk!

CBTF's assets, tools, accelerators and methods provide unique capabilities to support all aspects of core banking modernization

	What Do Banks Need To Modernize Their Core Systems	What CBTF Provides	CBTF Product Set
Strategy, Analysis & Planning	Strategy, Imperatives, Strong Business to IT Alignment	• Method to develop strategy, identify imperatives and define business to IT alignment	• CBM (Preferable but not mandatory) • CBM-IFW
	Business process analysis and identification of common business services	• Industry proven reference framework for business process, data models and service definitions • Banking specific solution templates e.g. acct opening, product bundling • 3rd party service components e.g. fees & charges	• IFW (preferable but not mandatory) • Bank Specific Solution templates & 3rd party service components • Process Server
	Service Oriented Design Constructs	• Service registry	• Websphere
	Message broker based integration middleware	• Complex event based routing • Bank specific messaging interfaces • Wrappers for legacy codes • Adaptors for linking into legacy systems • Common application management services such as audit, security etc.	• Websphere
	Business rules management	• Business rules engine	• iLog
	Master Data Constructs (customer, product, contract master)	• Master data management server	• MDM & banking master data extensions
	Data architecture and reporting for operational, transactional and analytical data	• Data management server • Reporting tools and templates	• BDW, Infosphere • Cognons, SPSS
	Legacy Asset Analysis and Modernization Methodology	• Analysis tools • Modernization methodology • Data migration methodology	• Rational Asset Analyzer, SUPA
Development	Integrated development environment	• Banking Transformation Workbench	• Component Business Modeler (CBM tools) • Websphere Business Modeler & Integration Developer • Information Framework (IFW) • Rational Products ► Requirement Composer, Rational Asset Manager, Rational Software Architect, Rational Data Architect, Rational Asset Analyzer, Rational team concert server and client • Infosphere Business Vocabulary
Deployment	Run-Time Environment		• Various S/W and H/W components
	IT system management and governance		• Tivoli

A key component of our CBTF are the solution templates – that uniquely differentiates us from our competitors

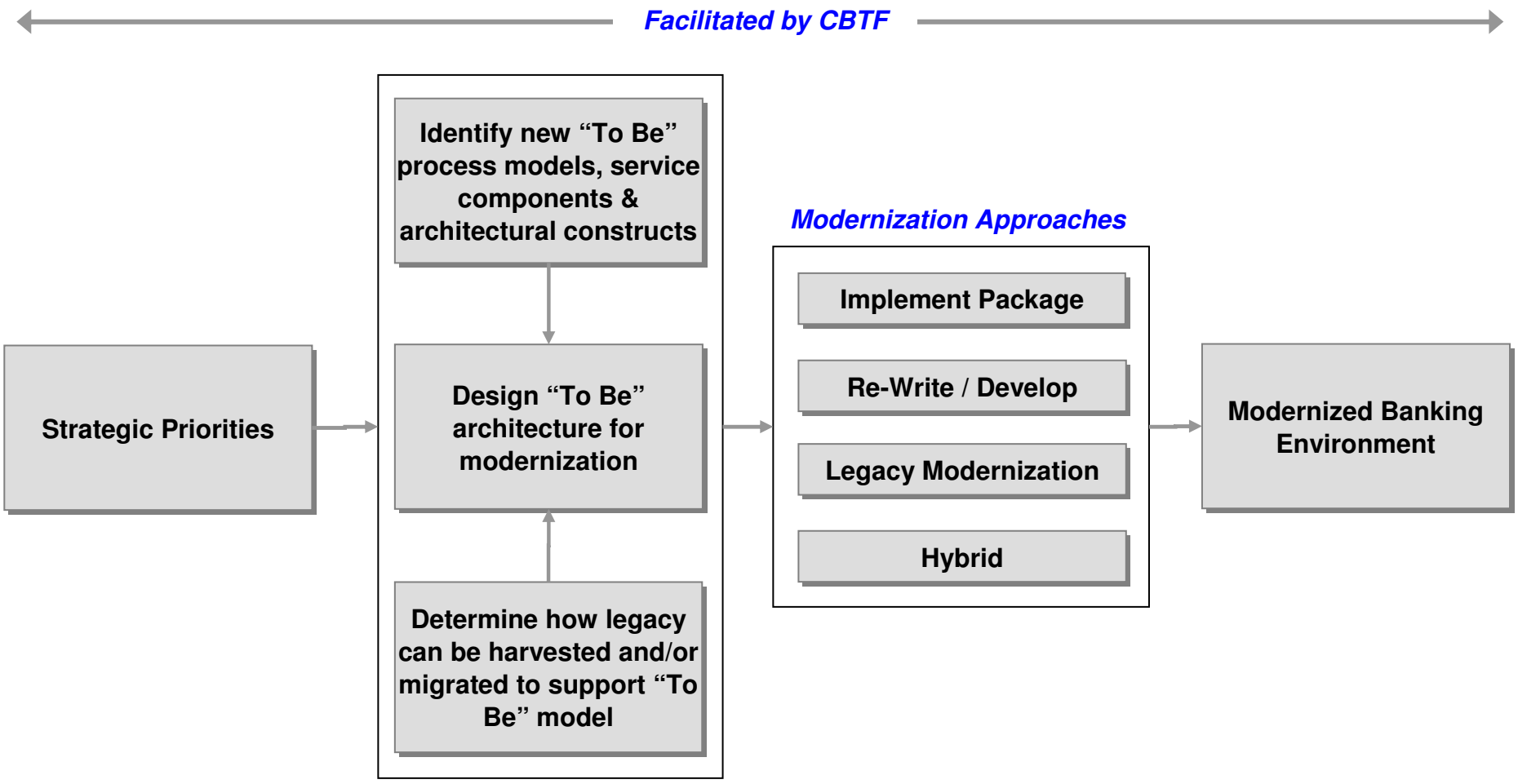
- Addresses a specific business problem e.g. account opening or product bundling
- Comes with pre-identified process, data and services model derived from industry standard reference frameworks
- Can be a starting point for most of modernization engagements
- Directly solves customer business related pain points
- Can be harvested from one client engagement to another thereby enriching the library of configurable solution components



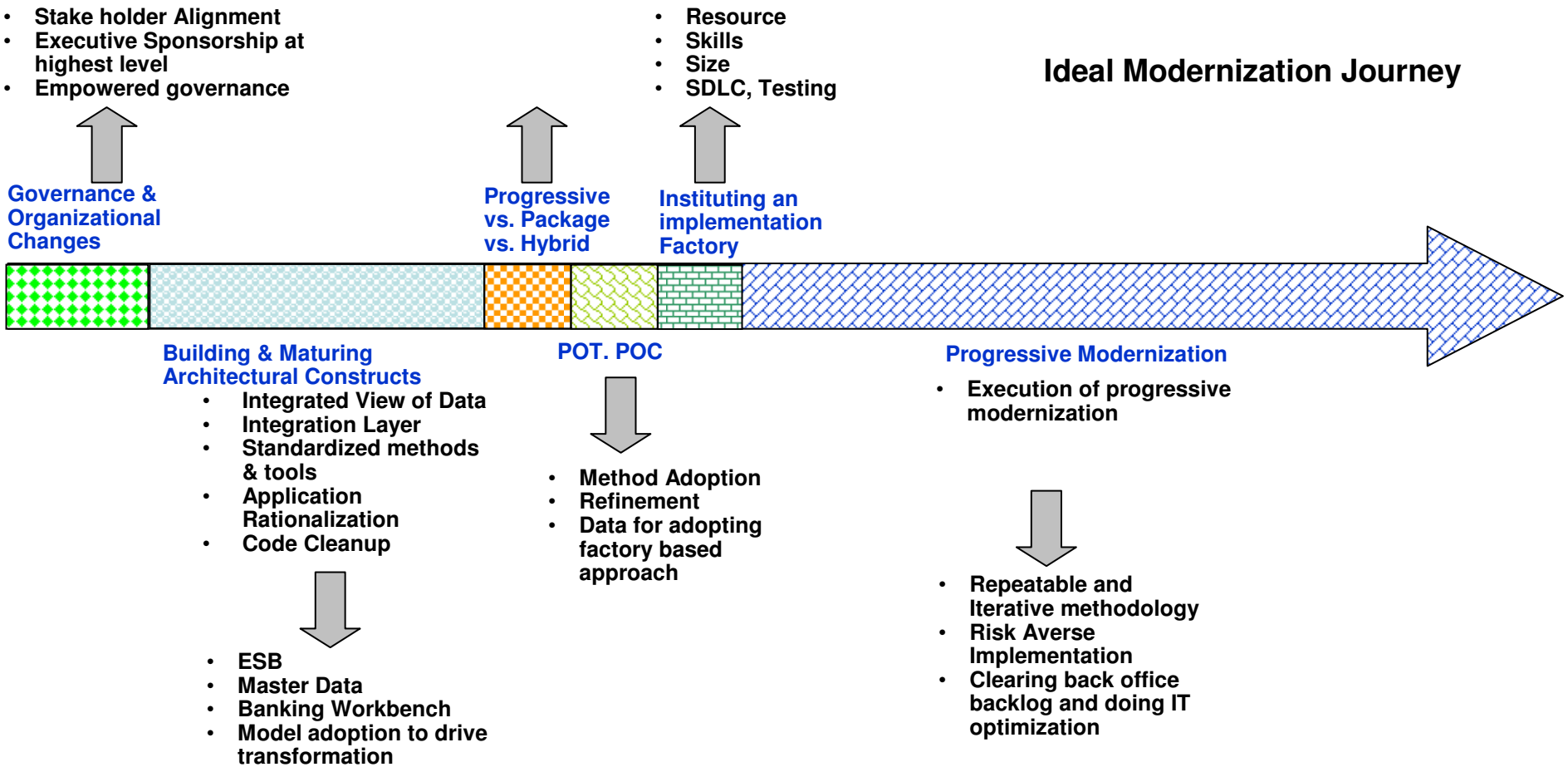
Topics for today's discussion

- Market Snapshot
- Core Banking Modernization Overview
- The Need For Architectural Discipline in Banking Modernization
- IBM's Core Banking Transformation Framework (CBTF)
- Approach and Methodology
- Case Studies

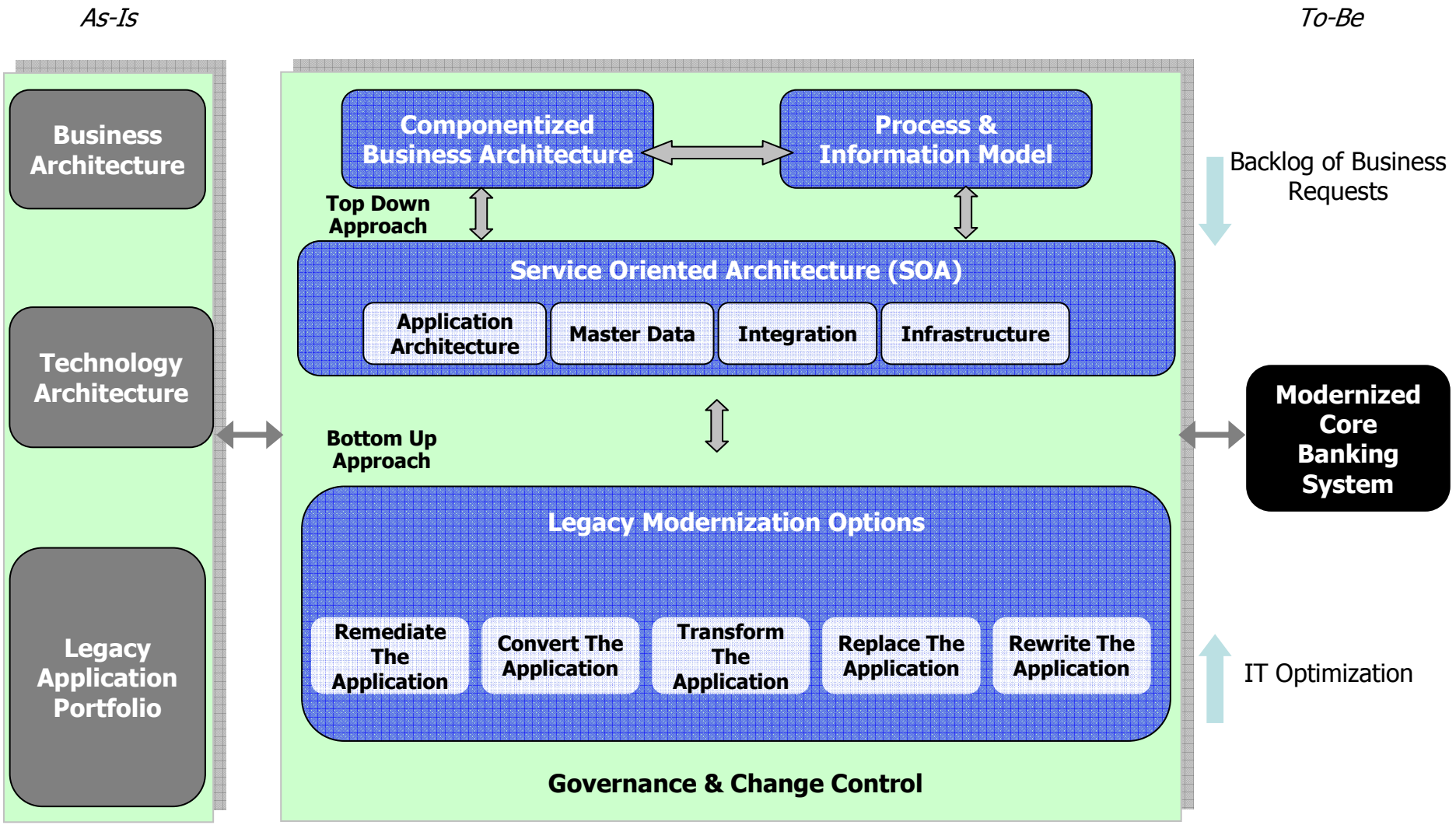
CBTF framework can enable banks to undertake one or more approaches to modernization based on strongest alignment of business needs with IT transformation efforts



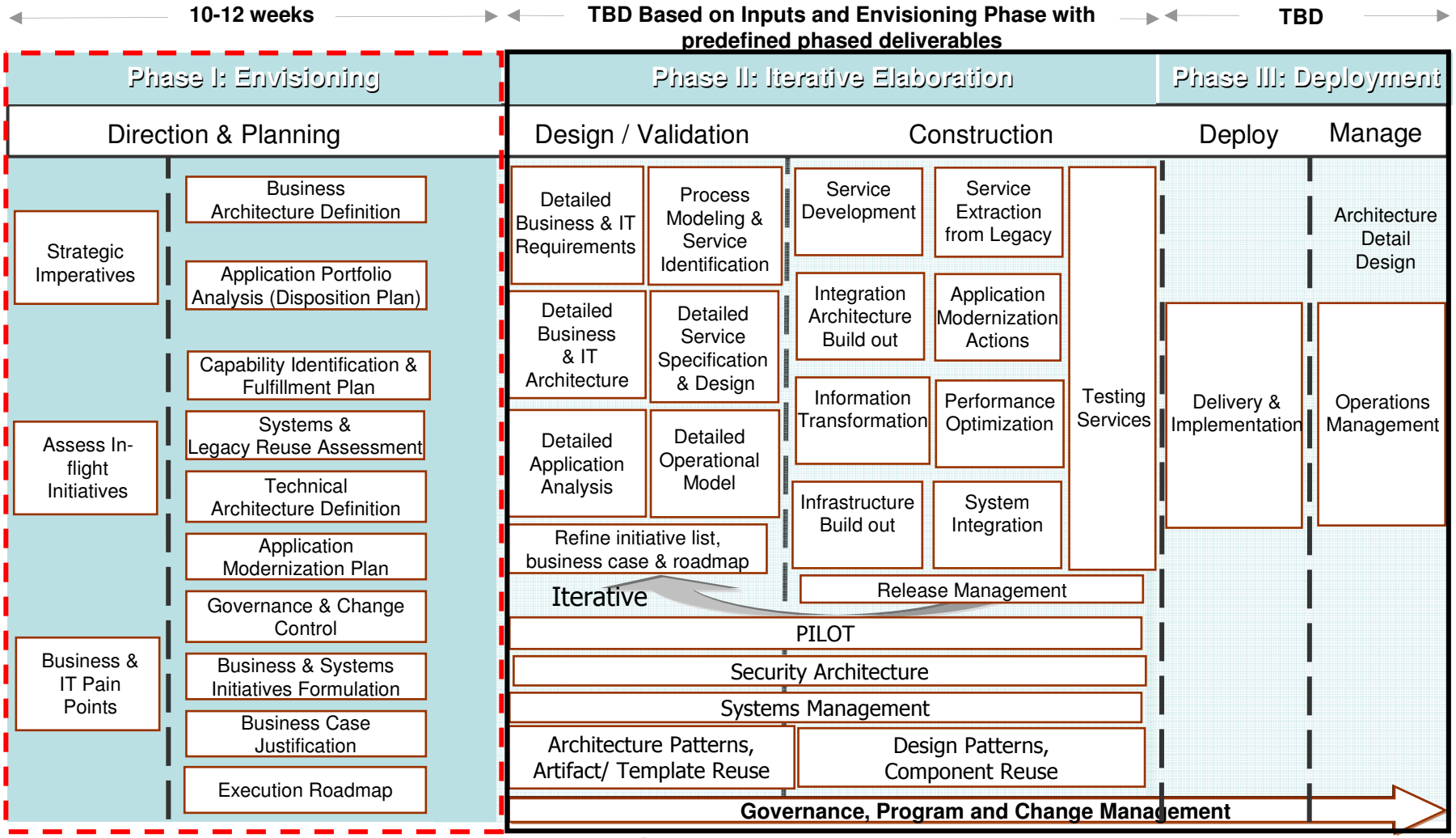
Core Modernization – Ideal Steps in Transformation



Modernization paths should include both top down approach for new business capabilities and a bottom up approach for harvesting unique differentiated capabilities from legacy assets

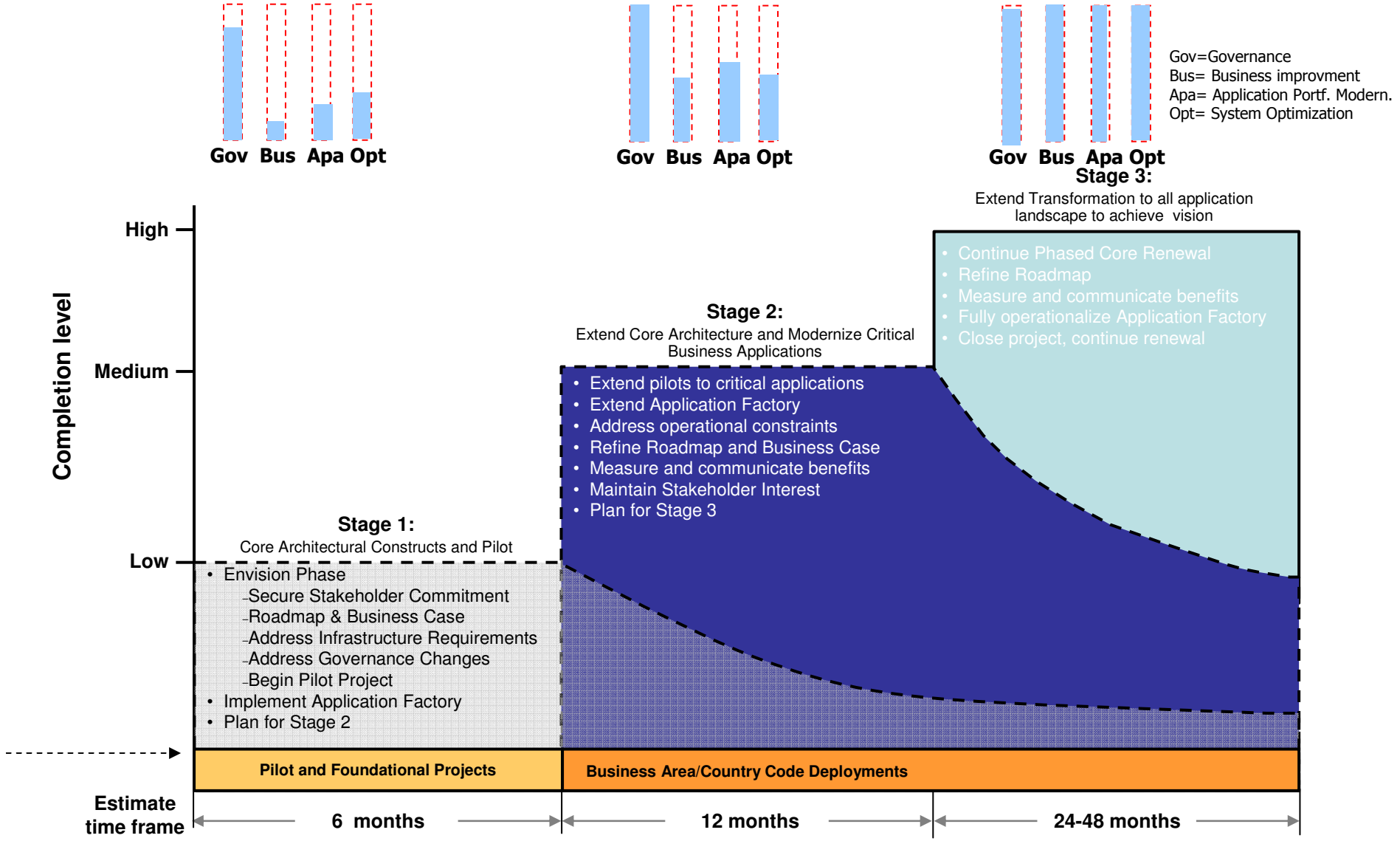


Following diagram breaks down different phases into more granular level and provides a snapshot of various steps during a core banking transformation initiative



IBM brings many industry tested points of views (POV's) as the starting point for the "To Be Architecture" as an accelerator to the approach

The Execution Roadmap is the key outcome of the envisioning phase, which will define how The Bank should execute to achieve based on CBTF approach



Topics for today's discussion

- Market Snapshot
- Core Banking Modernization Overview
- The Need For Architectural Discipline in Banking Modernization
- IBM's Core Banking Transformation Framework (CBTF)
- Approach and Methodology
- Case Studies

Case Study: Successful Package Implementation (Rip and Replace)

Bank : A bank in Asia

Customer Base : **>8** million

Branches : **>500**

Objectives : 30% YoY growth

Key Focus Area : Customer Acquisition

Time to Implement : 18 months

Motive: To quickly enter into the market with a set of banking products and focus on customer acquisition

Approach:

- Driven by CEO
- Choose package that best meets current business requirements > 80%
- Agreement to re-engineer business processes to suite package products and avoid costly customization
- Sales and Service processes aligned with features and functionalities offered by the package with minimal customization
- Early training on package features for bank staffs
- Investment in architecture and infrastructure in line to support package implementation

Results:

- On time and on budget implementation
- Branch transformation on new systems done overnight

Reasons for Success:

- Alignment with package features agreed very early on
- Strong governance driven right from the top

Current Situation and Future Impact:

- Bank achieved its targets in customer acquisition, actively supported by marketing blitz
- Bank is facing customer service, single customer view, new product introduction etc. challenges
- Thin architecture, shortage of skills etc. is limiting bank's ability to customize and expand the package capabilities

Case Study: Package Implementation (Progressive)

Bank : A large bank in North America

Customer Base : **>36**million

Branches : **>3000**

Objectives : YoY growth, cost

Key Focus Area : **Product Innovation,** Customer Acquisition and Customer Centricity

Time to Implement : Ongoing

Motive: To migrate from legacy systems to a package solution and drive product innovation, customer centricity

Approach:

- Driven by CEO
- Choose package that best meet the current and future business requirements > 50%
- Agree to customize package to suite business requirements
- Package modules heavily customize to meet unique product, know your customer and customer servicing across channels
- Significant investment made in putting a robust middleware (ESB) to connect all architectural components

Results:

- On time and slightly over budget implementation
- Branch transformation on new systems done in two phases

Reasons for Success:

- Big focus on capturing business requirements and freezing the requirements till implementation was done
- Strong governance driven right from the top
- Great emphasis on architectural constructs
- Good integration between CIF's during implementation

Current Situation and Future Impact:

- Bank is one the top banks in the country known for brining new products to the markets and delivering superior customer service
- Bank however finds itself limited by the bounds of the package solutions and is going out of the package to implement more complex products and achieve superior customer centricity across channels

Case Study: Unsuccessful Package Implementation (Progressive)

Bank : A large bank in Asia

Customer Base : **>10**million

Branches : **>1200**

Objectives : **12**% YoY, Cost

Key Focus Area : **Product Bundling, Single view of customer & Dynamic pricing**

Time to Implement : xxxx

Motive: To migrate from legacy systems to a package solution and deliver new business capabilities

Approach:

- Driven by Business & IT Steering Committee
- Choose package that best meet the current and future business requirements > 50%
- Agree to customize package by replicating capabilities from the existing legacy systems

Results:

- Unsuccessful attempt. Hugh cost and time overrun.
- The whole program had to be put on hold after spending 3 years and burning $300 million

Reasons for Failure:

- Attempt to capture current requirement got overblown due to complexity of the current legacy systems
- No governance or agreement to freeze BAU backlog hence change kept happening
- Inferior integration architecture. Bank had to develop many throw way point to point integration code and test them
- Legacy systems has cascading dependency and module replacement always grew in scope and effort

Current Situation and Future Impact:

- Bank is one the top banks in the country known for brining new products to the markets and delivering superior customer service
- Bank however finds itself limited by the bounds of the package solutions and is going out of the package to implement more complex products and achieve superior customer centricity across channels

Case Study: Legacy Modernization (Progressive)

Bank : A large North American Financial Institution

Customer Base : **>20 million**

Branches : **>6000**

Objectives : **CIR reduction, Share of Wallet**

Key Focus Area : **Time to Market, Efficiency, TCO reduction, Customer centricity**

Time to Implement : Ongoing

Motive: To modernize existing legacy systems for architectural simplicity, modularity and agility

Approach:

- Driven by Business & IT Steering Committee
- Adoption of SOA
- Legacy asset analysis and extraction / exposition of capabilities as re-usable services
- Top down design for identifying new services
- Adoption of architectural constructs that promote modularity and separation of architectural concerns

Results:

- Work still going. Initial architecture build out complete
- Bank saving almost 30% time on new components
- Few domain areas modernized into new environment

Reasons for Success:

- Architecture led strategy and focusing on establishing the core architectural constructs like master data
- Focus on both top down and bottom up approaches to modernization to accommodate BAU
- Strong governance structure to support a SOA development environment

Current Situation and Future Impact:

- Bank is well positioned on its strategy to harvest and enhance legacy assets and modernize them into a new SOA environment. They are doing it right from a componentized business architecture to drive all downstream application development.
- Bank has committed organizationally and financially to achieve a SOA for its banking applications

Case Study: Troublesome Legacy Modernization (Progressive)

Bank : A large North American Cards Company

Customer Base : **>40 million**

Branches : **XXXX**

Objectives : **Growth and Cost**

Key Focus Area : **Time to Market, Architectural Simplicity, SOA, Cost**

Time to Implement : Ongoing

Motive: To modernize existing legacy systems for architectural simplicity, reduced application maintenance cost and better architecture design

Approach:

- Driven by Business & IT Steering Committee
- Adoption of SOA
- Legacy asset analysis and extraction / exposition of capabilities as re-usable services
- Adoption of strategic architectural principles

Results:

- Adoption of SOA
- 1000's of services exposed in the environment
- Burdensome to maintain published services

Reasons for Troubles:

- Weak governance structure around service SDLC
- Mainly IT led initiative where bank focused on extraction and exposition of services from legacy assets
- Limited oversight on granularity of services resulting in 1000's of fine grained services
- Unsolved issues on ownership of services and how various applications will retrofit to use new SOA services
- Limited focus on new capability development using modernized environment

Current Situation and Future Impact:

- Organization is re-thinking its modernization strategy and trying to engage with business to bring the business relevance in identification of business services

Case Study: Troublesome Re-write Modernization

Bank : A large bank in South America

Customer Base : **>20 million**

Branches : **>3000**

Objectives : **Growth and Cost**

Key Focus Area : **Architecture, Time to Market, Product innovation, Customer centricity**

Time to Implement : Ongoing

Motive: To re-write most of the systems in new Java based environment using layered and modular application architecture

Approach:

- Driven by CIO
- Adoption of a componentized application architecture
- Legacy asset analysis and functional decomposition
- Development of sharable components (code base)
- Harvesting legacy assets for re-usable code (product engines, business rule implementation etc.)

Results:

- Bank struggled and hugely overran time and budget estimates
- Program was put on hold ($280 million, 3 years effort)
- Overall frustration with the approach among the application development community

Reasons for Troubles:

- The bank used a shared component (shared code base) model as a means to increase re-use. However as the code base was used across applications, any changes would require all sharing applications to go through testing process causing big drain on effort
- Weak governance structure on component SDLC
- Functional decomposition of legacy applications was not entirely successful because of complexity, poor documentation and lack of qualified business analysts

Current Situation and Future Impact:

- Organization is re-thinking its modernization strategy and trying to engage with business to bring the business relevance in identification of business services

Case Study: Re-Write Modernization

Bank : A large global bank

Customer Base : **>12 million**

Branches : **>1000**

Objectives : **Cost Reduction, Skills**

Key Focus Area : **Architecture, Time to Market, Modularity, Open Platforms**

Time to Implement : Ongoing

Motive: To re-write legacy application components for better code maintenance and change management

Approach:

- Driven by LOB's
- Legacy asset analysis and functional decomposition
- Harvesting legacy assets for re-usable code (product engines, business rule implementation and data access etc.)

Results:

- Bank re-writing key loan management system in new environment

Reasons for Troubles:

- The bank used SOA principles and conducted a detailed process analysis to identify common business services that can be developed as a re-usable and sharable services
- Bank developed 14 key coarse grain services and 42 fine grained services
- Service governance was not an issue as the IT was owned by the LOB
- Applications were retrofitted to use the new services. Portions of code were left alone and wrapped to expose them as services

Current Situation and Future Impact:

- Organization is thinking of creating a center of competency around the experience and scale the approach to other LOB's