Chapter 13

Legacy Information System

Overview

- Legacy Information System
- Solutions For LIS
- Wrapper
  - Def
  - Types of Wrapping (Database,System, Application, Function)
  - Encapsulation Level (Process, Transaction, Program, Module, Procedure)
  - Constructing Wrapper (two interface and three event driven modules)
  - Adapting Program for wrapper (Transaction, Program, Module, Procedure)
  - Screen Scraping
- Migration
  - Migration Planning (13 stage)
  - Migration Methods (7 methods)

Legacy Software

Legacy system:
“Any information system that significantly resists modification and evolution to meet new and constantly changing business requirements.”

Solutions for Legacy Software

There are several categories of solutions for legacy information system (LIS) or simply legacy system. These solutions generally fall into six categories as follows

- **Freeze:** The organization decides to not carry out further work on an LIS.
- **Outsource:** The organization may outsource it to a specialist organization offering this service.
- **Carry on maintenance:** Despite all the problems associated with supporting a software system, the organization decides to carry on maintenance for another period.
- **Discard and redevelop:** The organization throws all the software away and redevelops the application once again from scratch, using a new hardware platform and modern architecture tools and database.

Solutions for Legacy Software

- **Wrap:**
  - Black-box-based modernization technique.
  - Surround the LIS with a new layer of software to hide the complexity of the existing interfaces, applications, and data, with the new interfaces.
  - Gives existing components a modern and improved appearance.

- **Migrate:**
  - A LIS is ported to a modern platform,
  - Retaining most of the system's functionality and introducing minimal disturbance in the business environment.
  - Avoids the expensive and long process of new development.
  - Reuse of portions of the LIS, namely, implementation, design, specification, and requirements, is maximized.
  - Target system runs in a different computing environment.

Legacy Systems: Challenges and available solutions

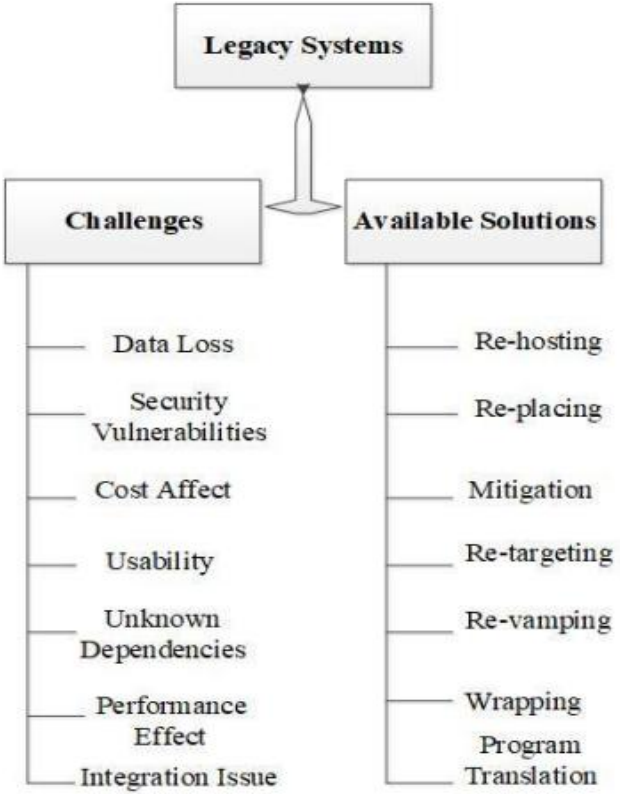

***Figure. 1**. Legacy System Challenges with Available Solutions*

Ali, M., Hussain, S., Ashraf, M. and Paracha, M.K., 2020. Addressing software related issues on legacy systems–a review. *International journal of scientific & technology research*, *9*(03), pp.3738-3742.

Modernizing Legacy Systems

- **Re-hosting**
  Use legacy software in different platform without any changes. This will decrease the maintenance cost of legacy hardware.
- **Re-placing**
  Replacement techniques is used whenever existing system cannot fulfill all the requirement of any organization/business. But in software legacy systems, replacing is done rewriting or adding new functionalities in old software's.
- **Mitigation**
  Adding new functionalities in new versions of software to address the error occurred in current versions. This is a technique to move the legacy applications to more flexible environments.

Ali, M., Hussain, S., Ashraf, M. and Paracha, M.K., 2020. Addressing software related issues on legacy systems–a review. *International journal of scientific & technology research*, *9*(03), pp.3738-3742.

Modernizing Legacy Systems

- **Re-targeting**

  Transformation of legacy system into new system with some additional features and functionalities. Converting a system platform into a new hardware platform.

- **Re-vamping**

  Used to replace the user interfere design in new design. It was a most visible part of any software/application. It is improving the visibility and usability of system.

- **Wrapping**

  Modifies the interaction logics in new version of any old system. This process is cost effective.

- **Program translation**

  Used to made the cross compatibility of software on different hardware and platforms.

Ali, M., Hussain, S., Ashraf, M. and Paracha, M.K., 2020. Addressing software related issues on legacy systems–a review. *International journal of scientific & technology research*, *9*(03), pp.3738-3742.

Wrapper

Wrapping a legacy system means **encapsulating** it in a way that **hides its internal complexity** and **exposes a modern interface** (e.g., web service, API) so that new systems can interact with it **without needing to understand or modify the legacy code**.

Wrapping is a “black-box” reengineering activity because the interface of the legacy system is analyzed but not its internal details.

There are four categories of wrappers:

- Database Wrappers.
- System service wrappers.
- Application wrappers.
- Function wrappers.

Database Wrappers

Database wrappers are software components designed to enable **modern applications** to interact with **legacy databases** or **non-standard data formats** without modifying the underlying data storage systems.

Database wrappers can be further classified into two types:

- Forward Wrappers
- Backward Wrappers

Forward Wrappers

- Adds a new component to a legacy system.
- New component is developed with modern practices, whereas the legacy database and the legacy code are not modified.
- The wrapper integrates the new component with the legacy system.

Backward Wrappers

- Data are migrated first following the Database first approach.
- New components are developed that use the new database.
- Legacy components access the new data via wrappers.

Forward and Backward Wrappers

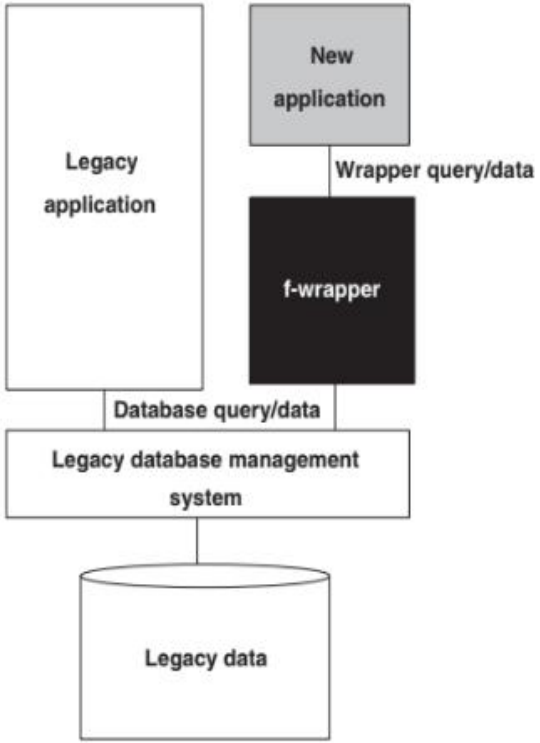

FIGURE 5.1 Forward wrapper. From Reference 10. © 2006 ACM

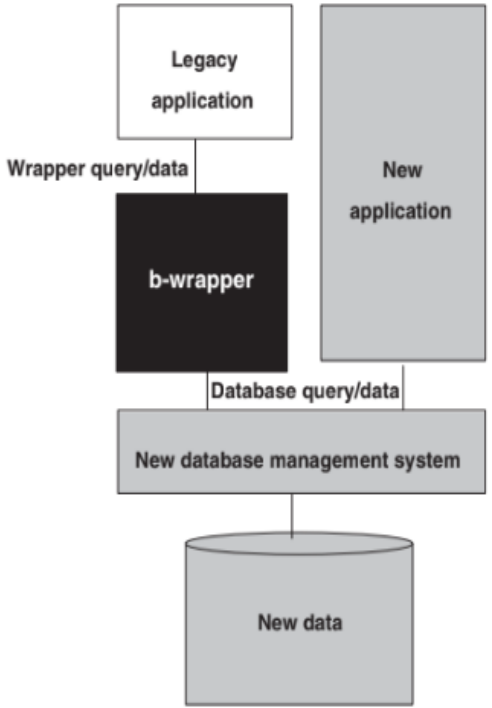

FIGURE 5.2 Backward wrapper. From Reference 10. © 2006 ACM

Types of Wrapping

System service wrappers: Support customized access to commonly used system services, namely, routing, sorting, and printing. A client program may access those services without knowing their interfaces.

A mobile app using a wrapper to access the device's GPS routing service without dealing directly with the GPS system interface.

Application wrappers: Encapsulate online transactions or batch processes. These wrappers enable new clients to include legacy components as objects and invoke those objects to produce reports or update files.

A modern web portal that integrates a legacy payroll batch process wrapped as a service to generate employee salary reports on demand.

Function wrappers: Provide an interface to call functions in a wrapped entity. In this mechanism, only certain parts of a program—and not the full program—are invoked from the client applications. Therefore, limited access is provided by function wrappers.

A financial analytics tool calling specific legacy calculation functions (e.g., tax computation) via wrappers, accessing only those functions without running the full legacy program.

Wrapper

R → PTPMP → R

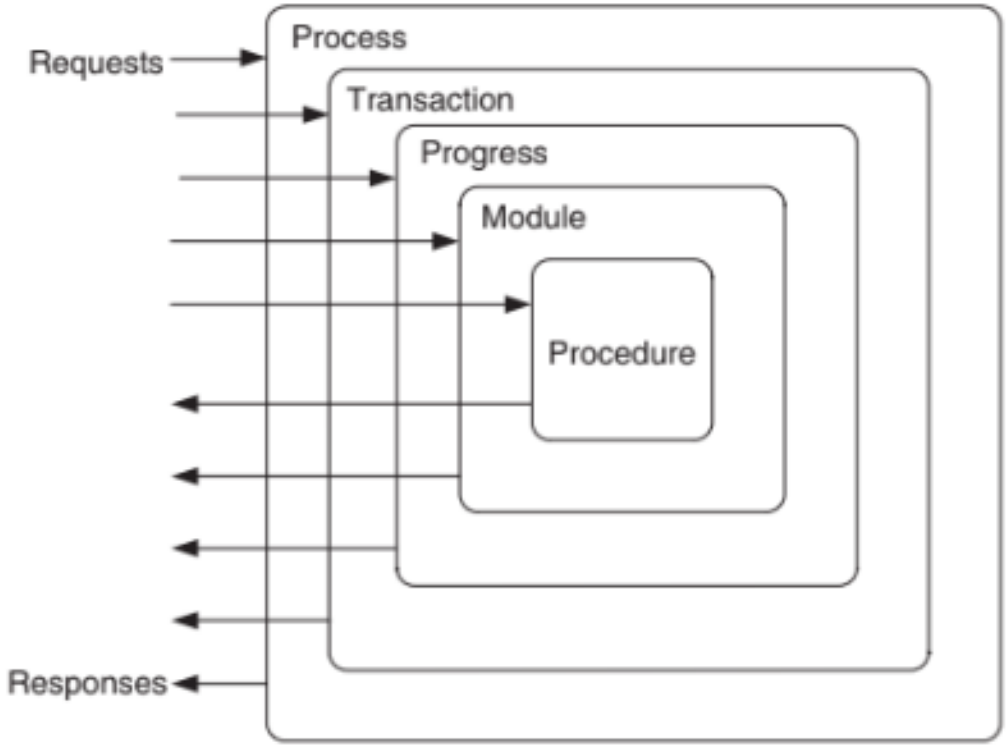

FIGURE 5.3 Levels of encapsulation. From Reference 11. © 1996 IEEE

Wrapper

Adapting a Legacy Program for Wrapping:

A wrapped program is modified to some extent, and it is expected that the modified legacy programs continue to operate in the normal mode.

Therefore, legacy programs are adapted with tools, rather than manually. Manual adaptations are prone to errors. Sneed [12] recommended four types of tools as discussed below.

- Transaction wrapper.
- Program wrapper.
- Module wrapper.
- Procedure wrapper.

Wrapper

Screen Scraping:

Screen scraping is a common form of wrapping in which modern graphical interfaces are used to replace text-based interfaces.

Screen scraping is a short-term solution to a larger problem. Many serious issues are not addressed by simply mounting a GUI on a legacy system. For example, screen scraping…

- Does not evolve to support new functions
- Incurs high maintenance cost
- Ignores the problem of overloading.

- Screen scraping simply provides a straightforward way to continue to use a legacy system via a graphical user interface.

Benefits and Problems of wrapping

Benefits:

- Quick and cost-effective solution.
- Changes and improves a system's outlook.
- As we use the same components of legacy system which are already tested and trusted by their company.

Problems:

- Wrapping is a short-term solution.
- Complicates system's maintenance and management.
- Additional software layer is used in wrapping which increases architecture overhead and increases page overload.
- Wrapping do not solve problems already present in maintenance and upgrading.
- Overhead of script interpretation and database access at the page load time.
- Performance and structure is not as better as can be achieved in redevelopment.

Aatif, K. and Farhan, S., Migration of Legacy Systems to Web Applications–A Novel Approach. *Journal of American Science2013*, *9*, p.11s.

Benefits and Problems of wrapping

Pros	Cons
Cost-effective	Short-term
Quick implementation	Complex maintenance
Trusted components	Architecture overhead
	Performance degradation
	Legacy problems persist

Aatif, K. and Farhan, S., Migration of Legacy Systems to Web Applications–A Novel Approach. *Journal of American Science2013*, *9*, p.11s.

Migration

- When wrapping is unsuitable and redevelopment is not acceptable due to substantial risk, Migration is an alternative

- Migration is to move an existing, operational system to a new platform, retaining the functionality of the legacy system, while causing as little disruption to the existing operational and business environment as possible.

- Migration involves changes, often including restructuring the system, enhancing the functionality, or modifying the attributes.

Strategies for Modernizing Legacy Systems

Step 1: Assess Your Legacy Systems

- Functionality
- Cost Analysis
- Security and Compliance
- Integration Capabilities

Step 2: Define Your Modernization Goals

- Improved Efficiency
- Cost Reduction
- Enhanced Agility
- Enhanced Security
- Compliance

Step 3: Create a Modernization Roadmap

- Prioritization
- Technology Selection
- Timeline
- Resource Allocation

https://www.linkedin.com/pulse/strategies-modernizing-legacy-systems-step-by-step-guide-ouderkirk/

Strategies for Modernizing Legacy Systems

Step 4: Choose the Right Modernization Approach
- Rehosting (Lift and Shift)
- Re-platforming (Lift and Reshape)
- Refactoring (Re-architecting)
- Retirement and Replacement

Step 5: Develop a Comprehensive Data Migration Plan
- Data Cleansing
- Data Mapping
- Testing

Step 6: Engage Stakeholders and Communicate Effectively

Step 7: Execute and Monitor

Step 8: Post-Modernization Evaluation

https://www.linkedin.com/pulse/strategies-modernizing-legacy-systems-step-by-step-guide-ouderkirk/

Migration steps

LIS migration involves creation of a modern database from the legacy database and adaptation of the application program components accordingly.

A database has two main components: **schema** of the database and **data** stored in the database.

Migration comprises three main steps:

- Conversion of…
  - Existing schema to a target schema
  - Data
  - Program

Migration Stpes

Hainaut, J.L., Cleve, A., Henrard, J. and Hick, J.M., 2008. Migration of legacy information systems. In *Software evolution* (pp. 105-138). Berlin, Heidelberg: Springer Berlin Heidelberg.

Migration

Testing and functionality:

- Migration engineers spent close to 80% of their time on testing the target system.
- Outputs of the target system are consistent
- To justify the project expense and the risk, in practice, migration projects often add new functionalities.

Migration

Cut over, also referred to as roll over:

The last step of a migration project is the cut over from the LIS to the target system.

There are three kinds of transition strategies, as proposed by Simon:

1. **Cut-and-run.**

- The simplest transition strategy is to switch off the legacy system and turn on the new system.
- Too risky because it would entail the entire information flow to be managed by a completely new system.

Migration

Cut over, also referred to as roll over:

2. Phased interoperability:

- To reduce risks, cut over is gradually performed in incremental steps.
- In each step, replace a small number of legacy components— data or application—by their counterparts in the target system.

3. Parallel operation:

- The target system and the LIS operate at the same time.
- Tests are continuously performed on the target system;
- Once the new system is considered to be reliable, the LIS is taken off service.

Migration Methods

No single approach can be applied to all kinds of legacy systems, because they vary in their scale, complexity, and risks of failure while migrating. The seven approaches are as follows:

1. Cold turkey
2. Database first
3. Database last
4. Composite database
5. Chicken little
6. Butterfly
7. Iterative

Migration Methods

Cold Turkey (Big Bang approach):

- Redesigning and recoding the LIS from the very beginning using a new execution platform, modern software architecture, and new tools and databases.
- The risk of failure is high for this approach to be seriously considered.

- When to use: If a legacy system has stable, well-defined functionality and is small in size, this approach could be adopted.

Database First (Forward Migration Method):

- This method first migrates the database, including the data, to a modern DBMS, and then gradually migrates the legacy application programs and interfaces.

Migration Methods

Database First (Forward Migration Method):

The LIS simultaneously operates with the new system via a forward gateway while interfaces and legacy applications are being reengineered.

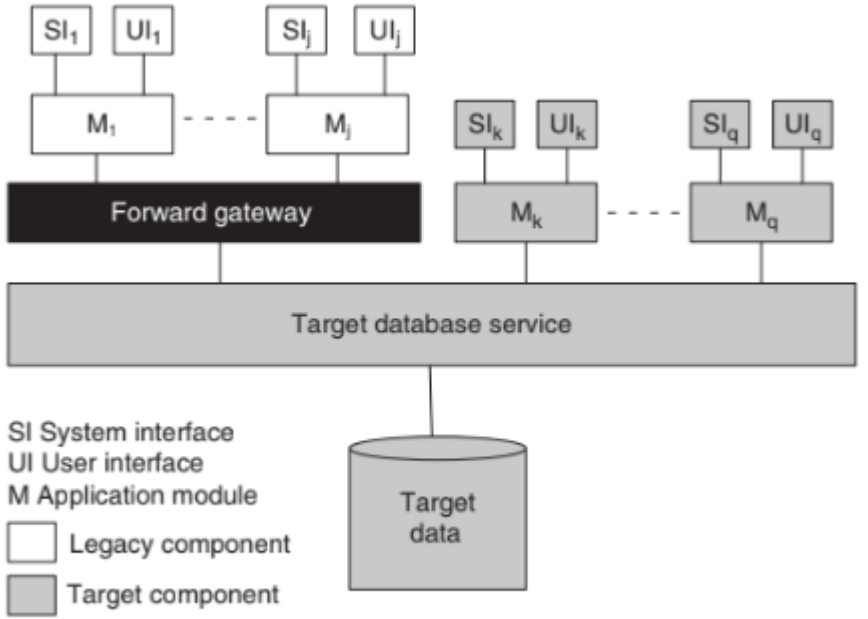

FIGURE 5.6 Database first approach. From Reference 19. © 1999 IEEE

Migration Methods

Database Last (Backward Migration Method):

When to use: When LIS is fully decomposable.

legacy applications are incrementally migrated to the target platform, but the legacy database stays on the original platform.

Migration of the database is done last. Similar to the database first approach, interoperability of both the information systems is supported by means of a gateway.

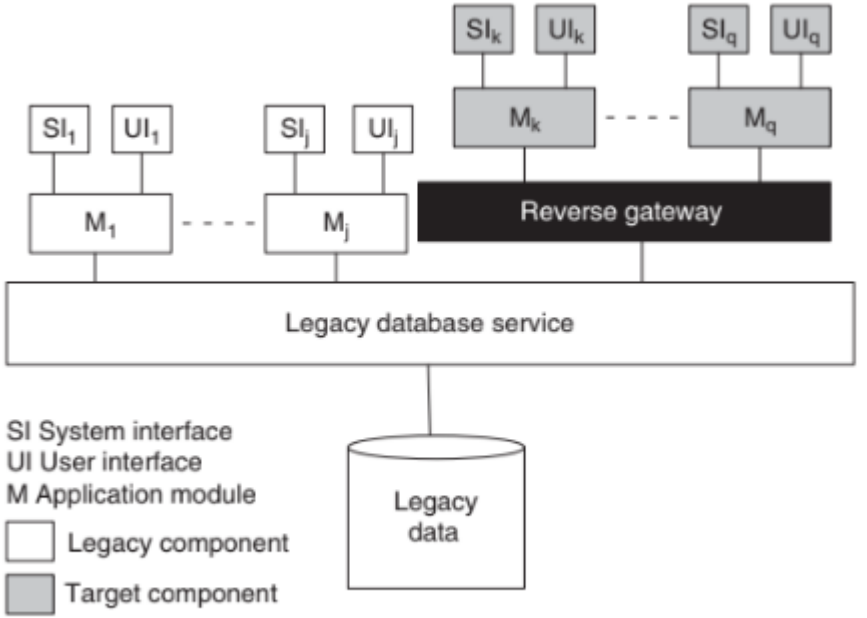

FIGURE 5.7 Database last approach. From Reference 19. © 1999 IEEE

Migration Methods

Composite Database:

When to use: When LISs are fully decomposable, semi decomposable, and non decomposable.

In the composite database approach, the target information system is run in parallel with the legacy system during the migration process.

While migration is continuing, as shown in Figure 5.8, the two systems form a composite information system, employing a combination of forward and reverse gateways. In this approach, data may be duplicated across both the databases: legacy and target.

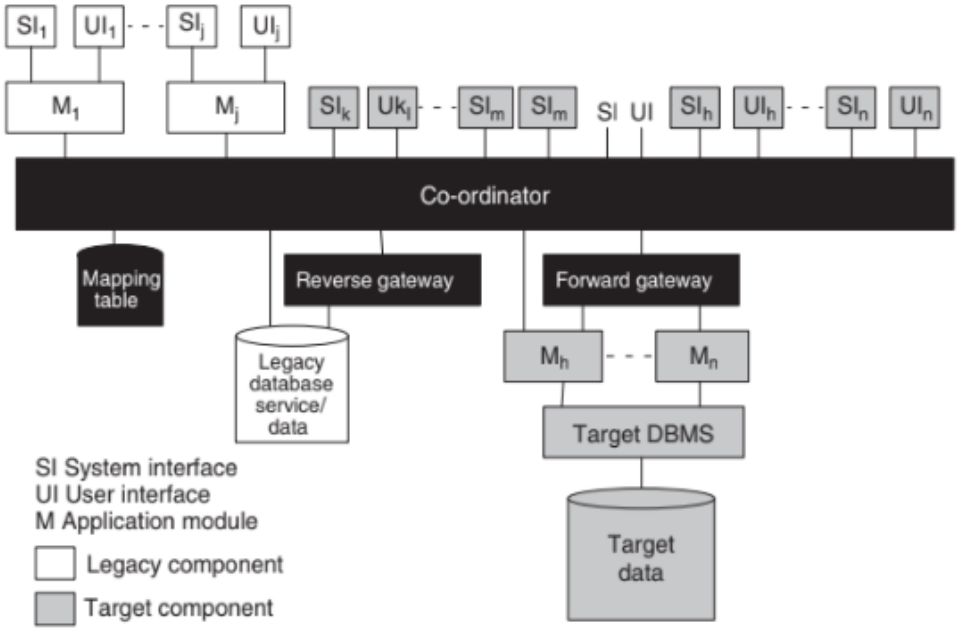

FIGURE 5.8 Composite database approach. From Reference 19. © 1999 IEEE

Migration Methods

Composite Database:

A transaction co-ordinators is employed to maintain data integrity.

The co-ordinators intercepts update requests from both target and legacy applications and processes the requests to determine whether or not the requests refer to data replicated in both the databases.

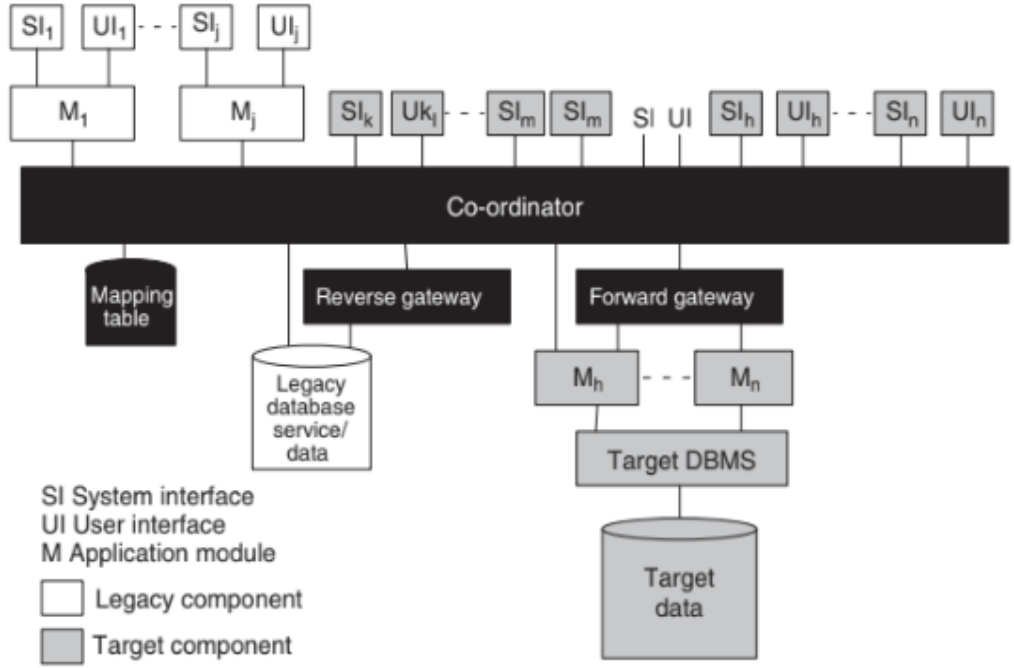

FIGURE 5.8 Composite database approach. From Reference 19. © 1999 IEEE

Migration Methods

Chicken Little:

The Chicken little strategy refines the composite database strategy, by proposing migration solutions for fully decomposable, semidecomposable, and nondecomposable legacy systems with different kinds of gateways.

The differences between those gateways are based upon:

- the locations of the gateways in the system
- the degree of functionality of the gateways.

Challenges And Strategies For Legacy System Migration

Pros and Cons of Legacy Systems

Pros	Cons
Familiar workflows	Security vulnerabilities
Proven stability	Compatibility issues
Established processes	High maintenance
-	Limited scalability
-	Outdated technology
-	Integration challenges
-	Performance issues
-	Skills shortage
-	Compliance gaps

https://medium.com/@amandubey_6607/challenges-and-strategies-for-legacy-system-migration-bca1181d14d2

Challenges And Strategies For Legacy System Migration

Migration Strategies

- Thorough assessment
- Clear objectives
- Detailed planning
- Pilot project
- Data integrity
- Phased approach
- User engagement
- Rigorous testing
- Process documentation
- Continuous monitoring

https://medium.com/@amandubey_6607/challenges-and-strategies-for-legacy-system-migration-bca1181d14d2

Cloud Migration

Cloud migration is the process of moving applications, data, and infrastructure from on-premises environments to cloud-based platforms.

When to use it: when you need improved scalability, cost optimization, operational efficiency, or enhanced collaboration between development and operations teams.

It's particularly valuable for organizations seeking to modernize legacy systems and leverage cloud-native benefits.

https://cloudfuel.eu/blog/how-to-prepare-your-application-portfolio-for-cloud-migrations/

Cloud Migration

How to Use It:

- **Define Clear Objectives** - Determine your motivation by exploring the four pillars of cloud-native applications (methodology, automation, modularity, infrastructure) and align with business goals.
- **Create Application Portfolio** - Document all applications including their purpose, dependencies, data flows, and interconnections to serve as your migration roadmap.
- **Choose Migration Strategy** - Select appropriate approach for each application (rehost, re-platform, refactor, retire, etc.) based on complexity, dependencies, and business requirements.
- **Conduct Detailed Architecture Analysis** - Analyze infrastructure components, identify improvement areas, and determine if containerization or microservices would enhance performance.
- **Execute and Monitor** - Implement the migration plan while remaining flexible, continuously monitor progress, gather lessons learned, and optimize applications post-migration.

https://cloudfuel.eu/blog/how-to-prepare-your-application-portfolio-for-cloud-migrations/