Legacy Systems - Basic Migration Methods

Idaho State University | Computer Science

Isaac Griffith

CS 4423 and CS 5523
Department of Computer Science
Idaho State University

ROAR

Idaho State University
Computer Science

Outcomes

After today's lecture you will:

- Understand and be able to describe the following basic methods for Legacy System migration
  - Cold Turkey
  - Database First
  - Database Last
  - Composite Database
  - Chicken Little

ROAR

Basic Migration Methods

CS 4423/5523

ROAR

Migration Methods

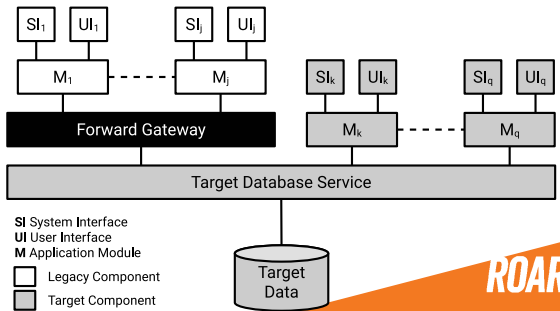
- No single migration approach applies to all legacy systems, as they vary in
  - scale
  - complexity
  - risk of failure
- The seven approaches are as follows:
  - **Cold turkey**
  - **Database first**
  - **Database last**
  - **Composite database**
  - **Chicken little**
  - Butterfly
  - Iterative

Cold Turkey

- aka the **Big Bang approach**
- Involves redesigning and recoding the LIS from the very beginning using:
  - a new execution platform
  - modern software architecture
  - new tools and databases.
- Due to the complexity
  - the renovated system must include many new features + the original functionality
  - the risk of failure is high
- Only use this approach for legacy systems that are
  - stable
  - well-defined functionality
  - small in size

Database First

- aka **forward migration method**
  1. migrate the database and data to a modern DBMS
  2. gradually migrates the program
- During migration:
  - LIS operates with the new system via a **forward gateway** while legacy components are migrated
    - A software module which mediates among operational software components
- This approach allows for:
  - LIS applications to access the database on the target side
  - The forward gateway to translate legacy calls to target calls
  - Translate target database outputs for use by the legacy system

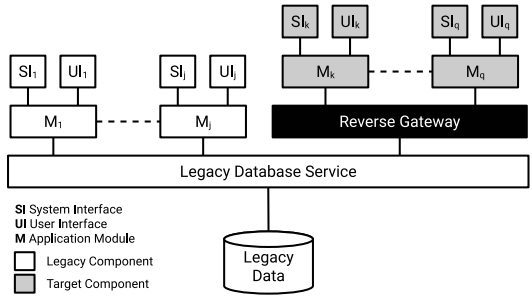


Database First

- **Benefits:**
  - Data is migrated first leading to improved productivity
  - Legacy system operates concurrently with migrated components
- **Drawbacks:**
  1. only applicable to a completely decomposable legacy system, with a clean interface to the legacy database
  2. the new database structure must be defined before migration can begin
  3. it is difficult to construct the forward gateway
- **System types:**
  - **decomposable** - the user and system interfaces, applications, and databases are considered to be distinct components connected by clearly defined interfaces.
  - **semidecomposable** - only the user and the system interfaces are separate components; the database service and applications are inseparable.
  - **nondecomposable** - one where no functional components can be separated

Database Last

- Suitable only for a fully decomposable LIS.
- In this approach
  - legacy applications are incrementally migrated to the target platform
  - the legacy database is done last
- Target applications access the LIS database via a **reverse gateway**
  - translates new applications calls into legacy database calls
  - supports interoperability between LIS and Target systems



Database Last

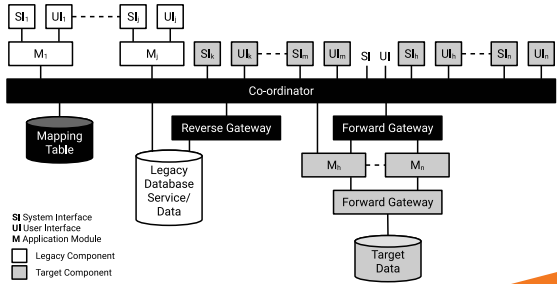
- Main issues with this approach:
  1. Performance is reduced: schema mapping of the target database to the legacy database can be slow
  2. Legacy database may not support useful features of relational databases
     - triggers
     - integrity
     - constraints

Idaho State University
Computer Science

Composite Database

- Applicable to all types of LIS.
- In this approach:
  - the target system and LIS operate in parallel during migration
    - forming a composite system through a combination of forward and reverse gateways
  - Initially, the target system is small, but will grow as migration continues
  - At completion, target system provides all functionality

- Approach employs a **Transaction Co-ordinator**
  - Allows data duplication across both databases
  - Maintains data integrity
  - Intercepts update requests from both applications to determine if it refer replicated data
    - If so, propagates update to both the databases using a two-phase commit protocol.



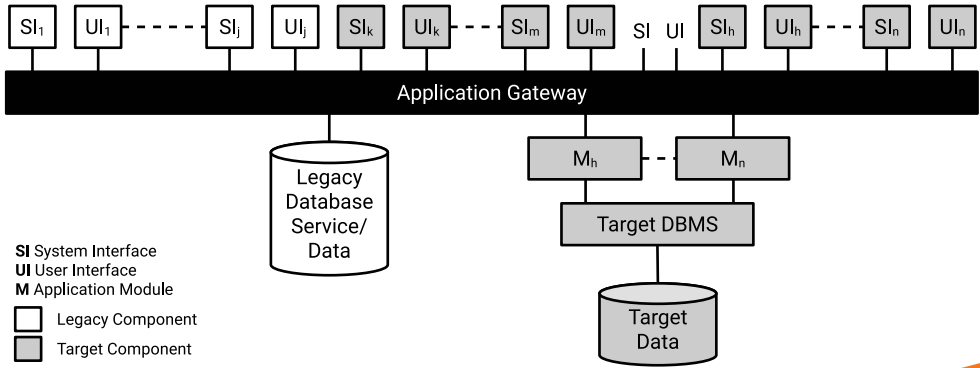
ROAR

Chicken Little

- Refines the composite database strategy
  - proposing migration solutions for fully decomposable, semidecomposable, and nondecomposable legacy systems with different kinds of gateways.
  - The differences between those gateways are based upon:
    1. the locations of the gateways in the system; and
    2. the degree of functionality of the gateways.
  - Still maintains concurrent operation of the target and legacy systems during the migration
  - Data is stored in both the migrating legacy system and the evolving target system
- Decomposable LIS:
  - **database gateway:** located between the database service and the application modules
    - can be either a forward gateway or a reverse gateway.

Chicken Little

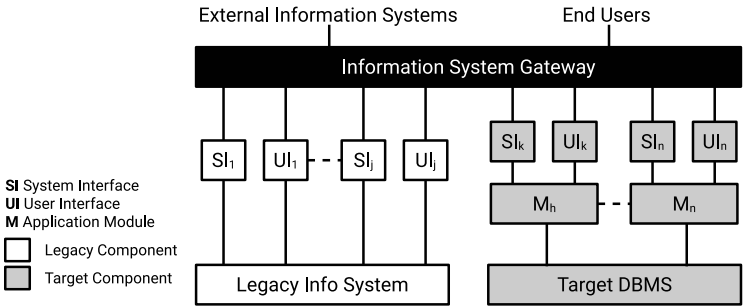
- Semidecomposable LIS:
  - **Application gateway** - located between user and system interfaces and the legacy application.
  - Encapsulates from the applications down (from the perspective of interfaces)



Idaho State University
Computer Science

Chicken Little

- Nondecomposable systems LIS
  - **Information systems gateway** - located between user and other information systems and LIS
  - The entire functionality of the legacy system is encapsulated
    - Differs from an application gateway which only encapsulates from the application module down.



ROAR

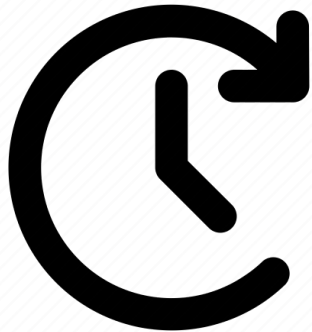
Chicken Little

- The Chicken little methodology proposes an 11-step incremental migration strategy
  - For each component to be migrated all steps are executed
  - The process is repeated for each increment

Step	Description
Step 1:	Incrementally analyze the LIS
Step 2:	Incrementally decompose the structure of the LIS
Step 3:	Design the interfaces of the target system in an incremental manner
Step 4:	Build the target applications in an incremental manner
Step 5:	Design the database in an incremental manner
Step 6:	Install the target environment in an incremental manner
Step 7:	Create and install the necessary gateways in an incremental manner
Step 8:	Migrate the databases in an incremental manner
Step 9:	Migrate the legacy applications in an incremental manner
Step 10:	Migrate the legacy interfaces in an incremental manner
Step 11:	Cut over to the target system in an incremental manner

For Next Time

- Review EVO Chapter 5.5.1 - 5.5.5
- Read EVO Chapter 5.5.6 - 5.6 and 6.1 - 6.2
- Watch Lecture 13
- **4423:**
  - Weekly Quiz due 2/7 @ 11:00 pm
  - Start on Homework 02
- **4423 Project:**
  - Course Project Part 2: System Selection is due 2/5 @ 11:00 pm
  - Team Evaluation Part 2 is due 2/7 @ 11:00 pm
- **5523:**
  - Project Topic Selection Report is due 2/5 @ 11:00 pm
  - Project Topic Selection Lightning Talk is due 2/5 @ 11:00 pm
  - Paper Review 02 is due 2/7 @ 11:00 pm



Are there any questions?