Legacy Systems - Advanced Migration Methods

Idaho State University | Computer Science

Isaac Griffith

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

ROAR

Outcomes

After today's lecture you will:

- Understand and be able to describe the following advanced methods for Legacy System migration
  - Butterfly
  - Iterative

Butterfly Method

CS 4423/5523

ROAR

Butterfly

- Does not require simultaneous access to the legacy and target databases
  - Target system is not operated in the production mode
- During migration:
  - The legacy system remains operational
  - Live data is stored only in the legacy system
- For data migration we introduce the following concepts:
  - Sample DataStore, Legacy SampleData, and Target SampleData
  - TempStore
  - Data-Access-Allocator
  - Data-Transformer
  - Termination-Condition and Threshold Value

Key Concepts

- **Legacy SampleData** - A representative subset of the legacy database
- **Target SampleData** - derived from the **Legacy SampleData**.
- **Sample DataStore** stores the **Target SampleData** using the target system data model
  - Supports the initial development and testing of all components
- **TempStores (TS)** - sequence of temporary data stores used to migrate legacy data
  - Hold the results of manipulations on the legacy data
- **Data-Access-Allocator (DAA)** - directs those manipulations
  - The results are stored/retrieved by the DAA in the most recent **TempStore**.
- **Chrysaliser** - a **Data-Transformer** which migrates the legacy and **TempStore** data to the target system

Butterfly

- The concepts of Termination-Condition and Threshold Value determine whether the new system is ready to be used
  - **Termination-Condition:** size(TS) ≤ ε
    - implies that the time required to migrate the data is negligibly small
    - we can shutdown the legacy system without too much disturbance to the business
  - **Threshold Value**, ε, pre-determined value representing the acceptable quantity of data remaining in the current TS.

Butterfly

- Migration is organized into six phases

Phase	Description
Phase 1:	Readiness for migration
Phase 2:	Comprehend the semantics of the system to be migrated and develop schema(s) for the target database
Phase 3:	Based upon the Target SampleData, construct a Sample DataStore
Phase 4:	Except the data, migrate the components of the legacy system
Phase 5:	Gradually migrate the legacy data
Phase 6:	Roll over to the new system

Idaho State University
Computer Science

1. Readiness for migration

Preparation Phase primary concerns

- Identify user's requirements
- Specify the target system

Activities

ID	Description
1.1	Identify the basic requirements of migration
	1.1.1 Identify the user requirements
	1.1.2 Identify the criteria to measure the success of the migration process
1.2	Design the architecture of the target system
1.3	Set up the target hardware platform

Idaho State University
Computer Science

2. Comprehend Semantics

- Must understand the legacy system
- **Data-Access-Allocator (DAA) tool:** developed to redirect all manipulations of legacy data and data stored in TempStores.

Activities

ID	Description
2.1	Comprehend the existing interfaces, determine redundancies in the existing interface, and determine the new interfaces
2.2	Comprehend the legacy applications, determine redundancies in the legacy applications, and identify the functional requirements of the new applications
2.3	Comprehend the existing legacy data, determine redundancies in the legacy data, and determine what legacy data to migrate
2.4	Identify and comprehend the interactions of the legacy system with its environment
2.5	**Identify the requirements of migration**
2.6	Design and develop the data redirector tool called Data-Access-Allocator
2.7	Design the schema for the new data and identify the rules for mapping of data

Idaho State University
Computer Science

3. Construct Sample DataStore

- **Primary Concerns**
  - Developing the Chrysaliser
  - Determining the legacy SampleData
- The Chrysaliser derives the Sample DataStore from the SampleData
- The Sample DataStore is used to test and develop the target system.

Activities

ID	Description
3.1	Construct the Legacy SampleData
3.2	Develop the Chrysaliser
3.3	Construct the Sample DataStore by using the Target SampleData derived from the Legacy SampleData

Idaho State University
Computer Science

4. Migrate Components

- Forward software engineering principles used to develop new target components
  - The Sample DataStore supports this engineering process

Activities

ID	Description
4.1	Migrate legacy interface
	4.1.1 Migrate/develop a portion of the interface of the target system
	4.1.2 For correctness, test the target interface against Sample DataStore
	4.1.3 Validate the target interface against user's requirements
4.2	Migrate legacy applications
	4.2.1 Migrate/develop a target application
	4.2.2 Test the target application against Sample DataStore for correctness
	4.2.3 Validate the target application against the requirements
4.3	Migrate the reusable components of the legacy system
4.4	**Integrate target components/system** (verify interactions)
4.5	Test target components/system
4.6	Validate target components/system against the user's requirements
4.7	Train the users of the new system

5. Migrate Legacy Data

- Legacy data migration is central to the Butterfly methodology
  - Data is incrementally migrated using:
    - TempStores
    - Chrysaliser
    - Data-Access-Allocator (DAA).

Activities

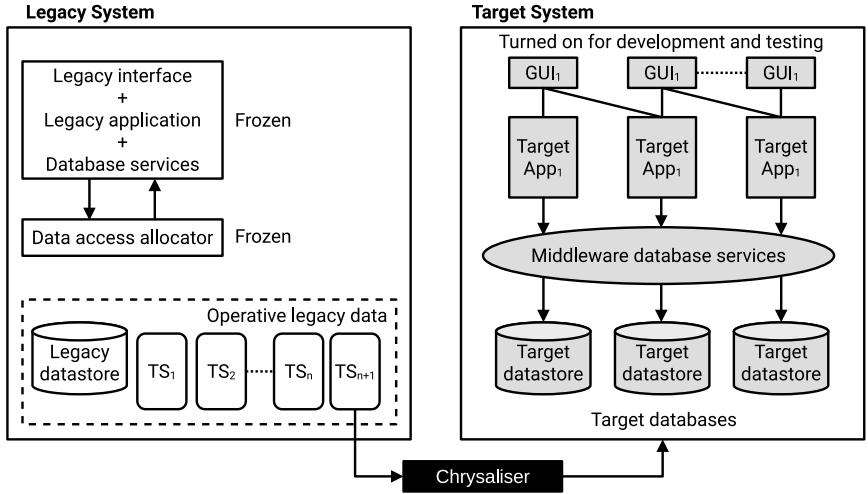
ID	Description
5.1	Integrate the Data-Access-Allocator into the legacy system
5.2	Create TempStore TS1 and set the access mode of the legacy datastore (TS0) to read-only
5.3	The contentes of TS0 are migrated into the new data store by means of the Chrysaliser. Accesses to the legacy data store are redirected by the DAA while migration is continuing, and results of the manipulations are stored in TS1
5.4	Create TempStore TS2; then set TS1 to read-only
5.5	Through the Chrysaliser, migrate TS1 into the target datastore(s). The DAA redirects all accesses to the legacy data, and all mainpulation results are stored in TS2
5.6	Repeat steps 5.4 and 5.5 for TS(n+1) and TS(n) until the Termination-Condition is satisfied
5.7	Do not make changes to the legacy system. Transform TS0 into the new, target datastore(s) by means of the Chrysaliser
5.8	Train the users about the target system.

5. Migrate Legacy Data

- Legacy system remains accessible the entire time
- No changes to the legacy data store are allowed
- **Data-Access-Allocator (DAA)** redirects manipulation operations of legacy data
  - The results are saved in a series of TempStore(s) (TS).
  - DAA selects correct source (legacy or TempStore) when a legacy application needs to access data
- **Chrysaliser:** migrates data to the new system
  - from the legacy store and the series of TempStores

5. Migrate Legacy Data

TempStore Migration



ROAR

6. Roll Over to New System

- Once the new system is built and legacy data migrated
- New system is ready for operation.

Butterfly Drawbacks

- Legacy database is read-only, while modifications are placed in a separate, temporary database
  - All data accesses require the system reads
    - the old database
    - the target database
    - the temporary databases
  - Increases the time to access the required data.

Iterative Method

CS 4423/5523

Iterative

- One component at a time is reengineered.
- Gradually evolving the legacy system over a period of time.
- Enables simultaneous operations of the reengineered and legacy system components
- Reengineered components access either the legacy or new database
  - based upon the location of data needed

Legacy Data Categories

- Legacy data is partitioned into two categories:
  - **Primary data:** data essential to an application's business functions. This is further subdivided into:
    - **Conceptual data:** These data describe concepts specific to the application's domain.
    - **Structural data:** These data are needed to organize and support data structures that are used to correctly access the conceptual data.
  - **Residual data:** other data used by the legacy system, which will eventually be discarded. This is further subdivided into:
    - control data.
    - redundant structural data.
    - semantically redundant data.
    - computationally redundant data.

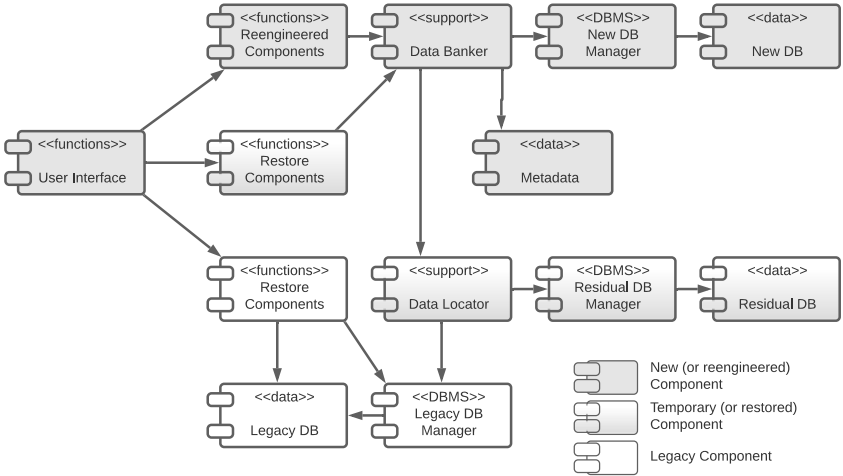
Component States

- Each Component may be in one of the following states during the reengineering process:
  - **Legacy:** This state means that the component has not been reengineered, which implies that it continues to be a legacy component.
  - **Restored:** The structure of the component remains the same and it performs the same functions, but the component accesses data through the new Data Banker.
  - **Reengineered:** The component has already been modified, and a desired level of quality has been achieved.
  - **New:** The component was not part of the legacy system and has been newly added in order to introduce new functions.

Idaho State University
Computer Science

System Architecture

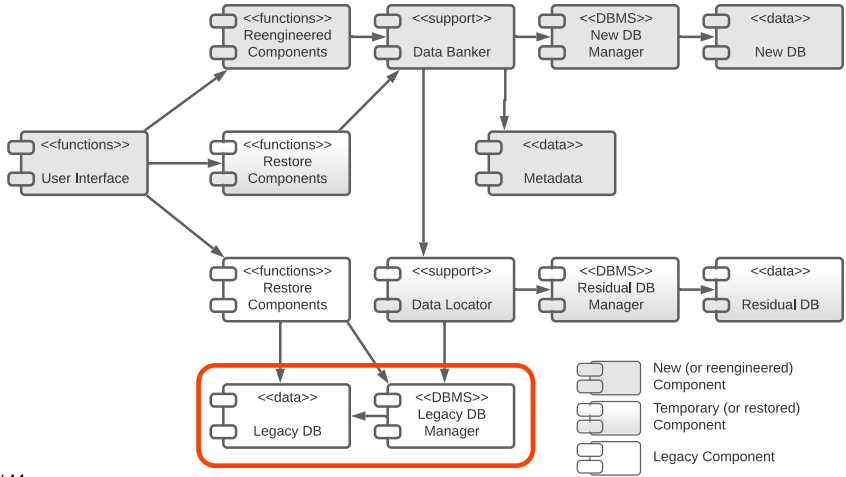
While reengineering is in progress, the figure shows the system architecture of gradual evolution of a legacy system.



ROAR

System Architecture

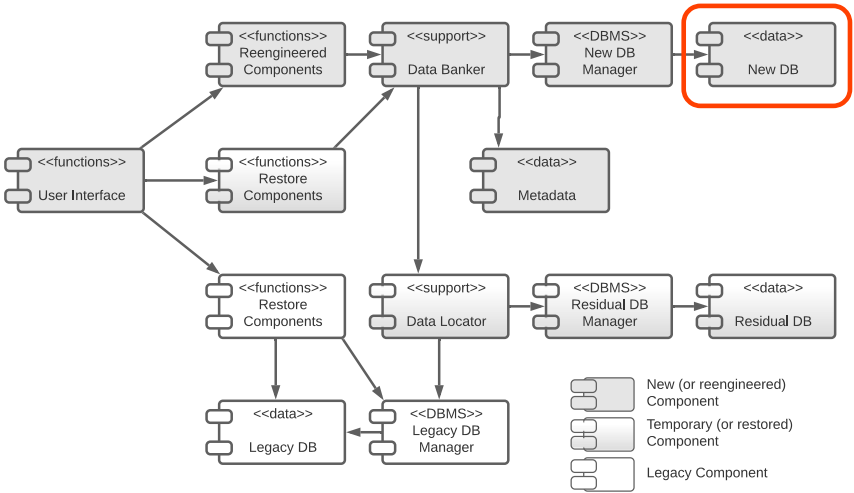
- The **Legacy DB manager** manages all accesses of the **Legacy DB** by the legacy components.



Idaho State University
Computer Science

System Architecture

- The database with the new structure is represented by **New DB**

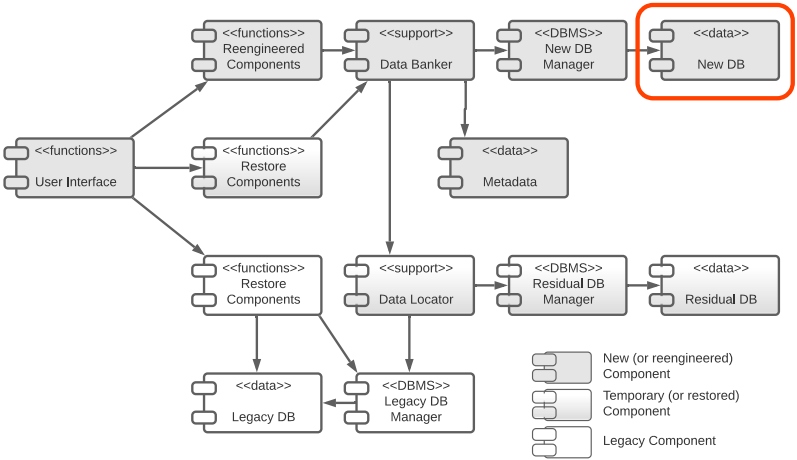


ROAR

Idaho State University
Computer Science

System Architecture

- **NewDB** includes:
  1. all primary data for those functions which have been added to the new system
  2. all primary data which have been migrated from the legacy DB.

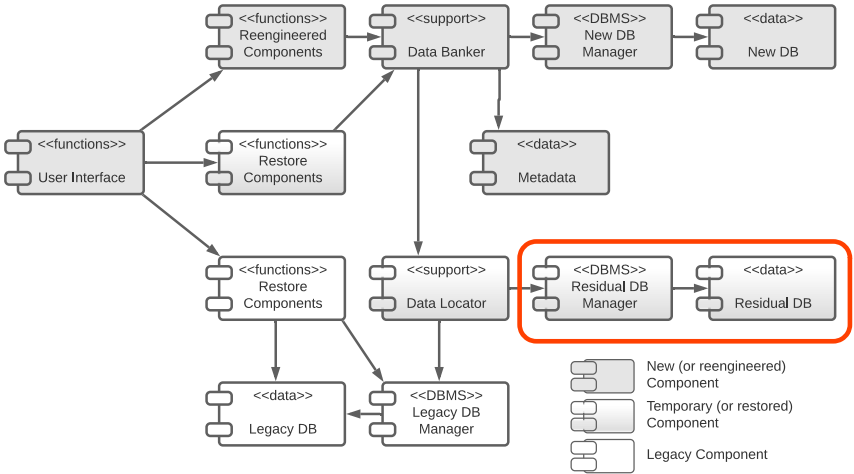


ROAR

Idaho State University
Computer Science

System Architecture

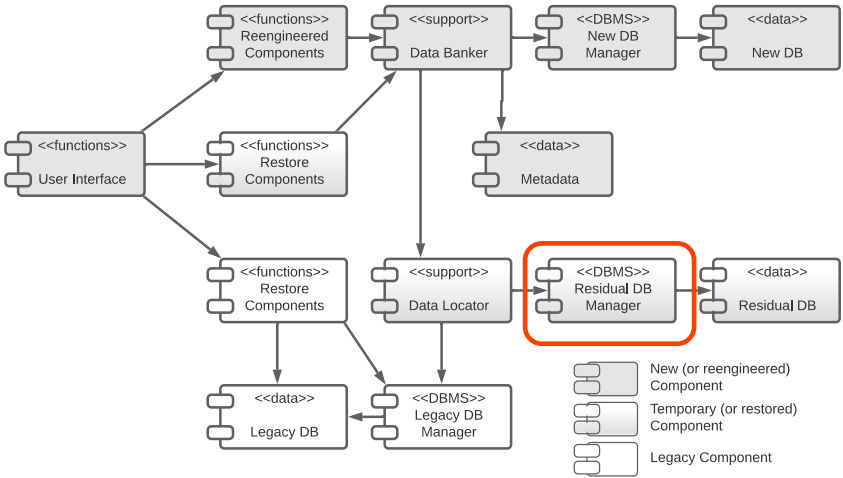
- **Residual DB** stores the residual data from the legacy database
- **Residual DB Manager** is the corresponding data manager



ROAR

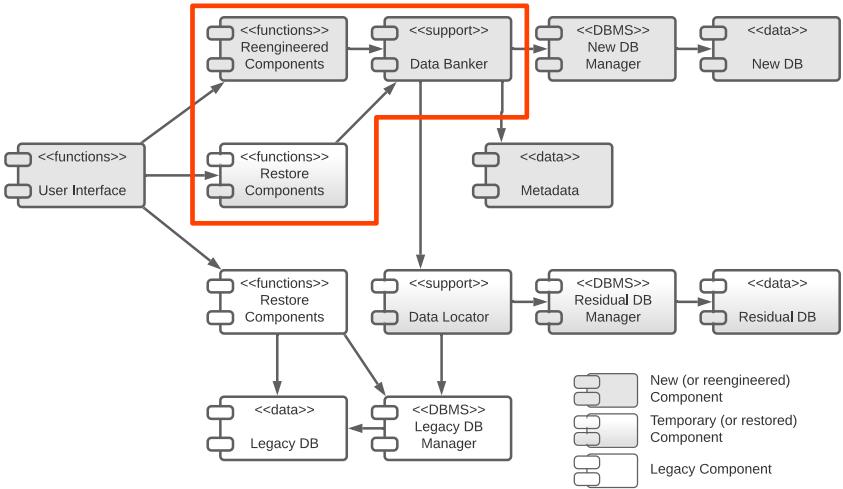
System Architecture

- The **Residual DB Manager** may be the DBMS of the legacy software system or the new database



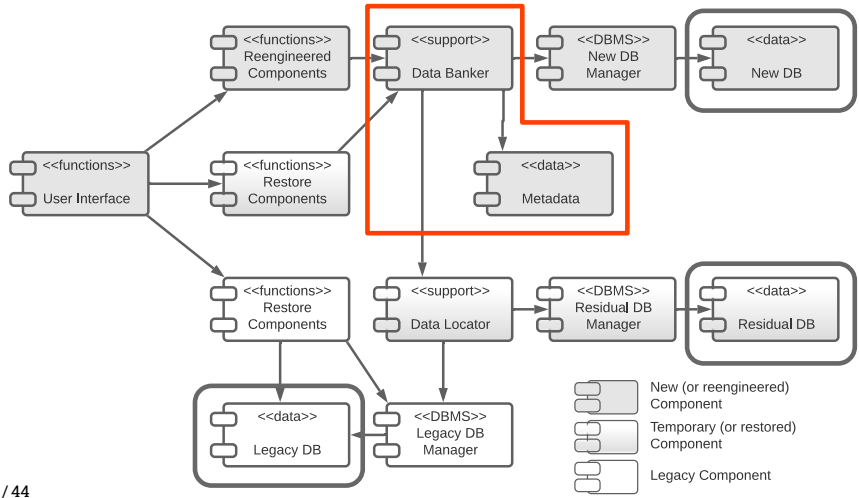
System Architecture

- The **Reengineered Components** and the **Restored Components** get data from **Data Banker**



System Architecture

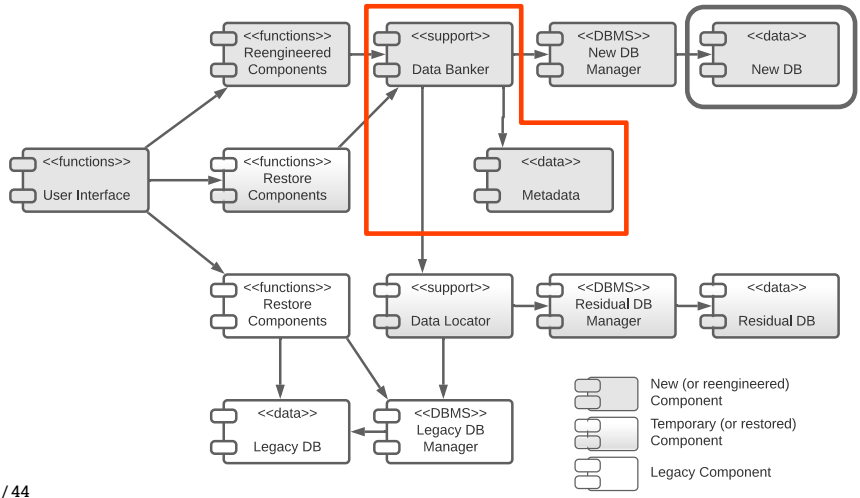
- By means of **Meta Data**, **Data Banker** knows whether or not the required data is available in **New DB**, **Residual DB**, or **Legacy DB**



Idaho State University
Computer Science

System Architecture

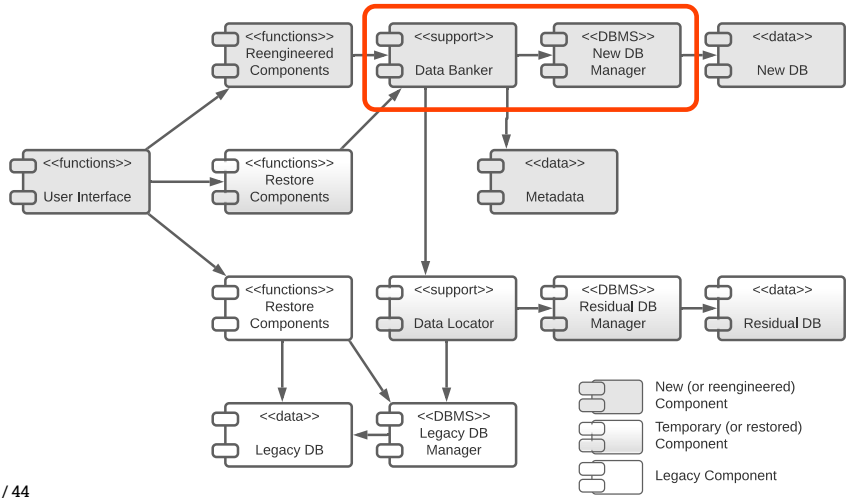
- If the **Data Banker** needs to access the **New DB**, **Metadata** are used to retrieve their physical locations.



ROAR

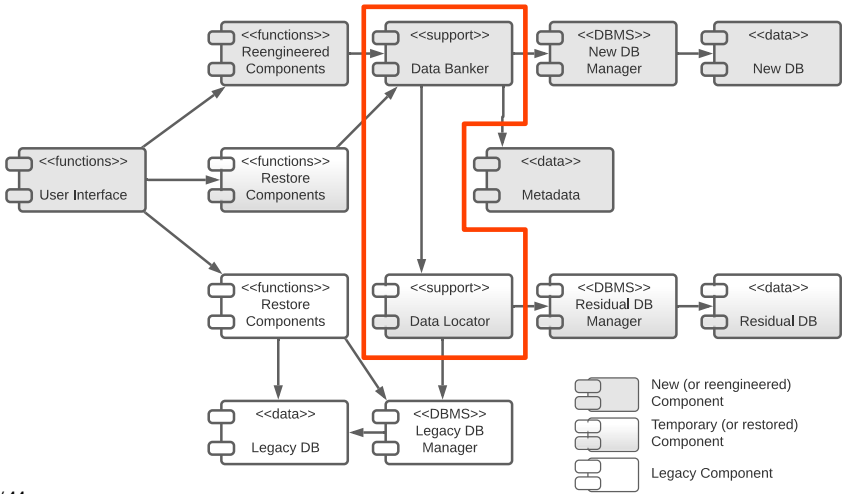
System Architecture

- Next, **Data Banker** uses the knowledge of the physical data locations to obtain data from the **New DB Manager**



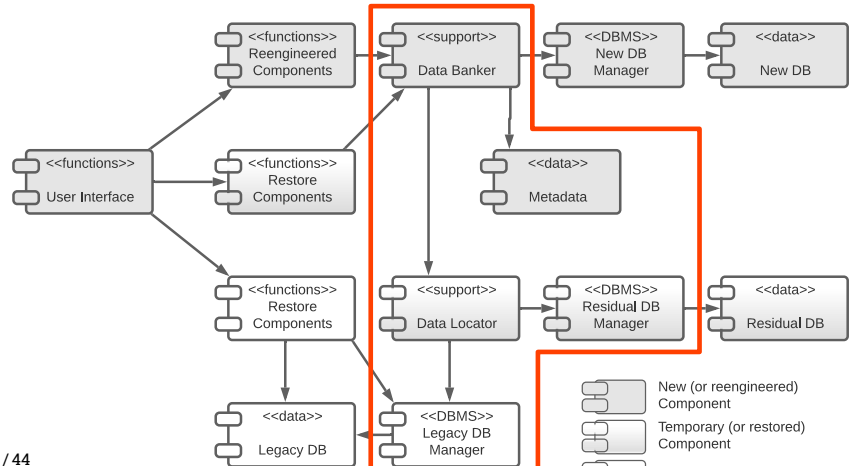
System Architecture

- In the aforementioned two cases, **Data Banker** routes the requests for data to the **Data Locator**



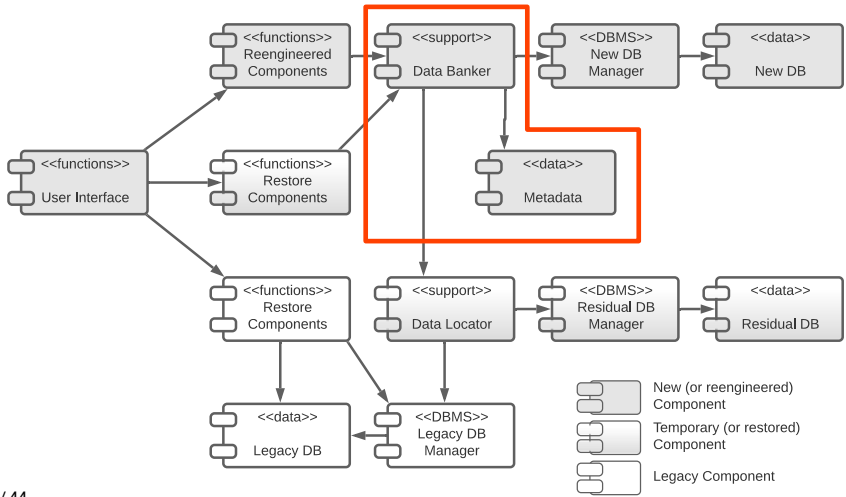
System Architecture

- The **Data Locator**interprets the information that the **Data Banker** has extracted from **Metadata** and gets the required data from the **Residual DB Manager** or the **Legacy DB Manager**



System Architecture

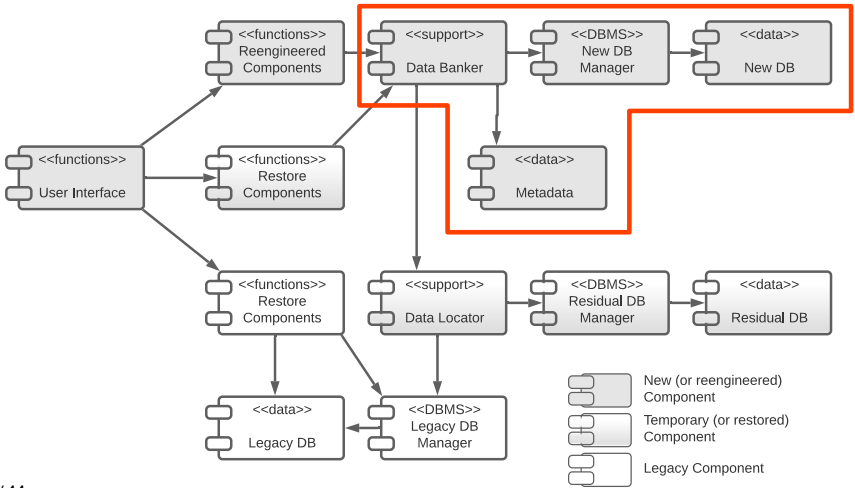
- Both the **Data Banker** and **Metadata** will continue to exist beyond the reengineering effort.



Idaho State University
Computer Science

System Architecture

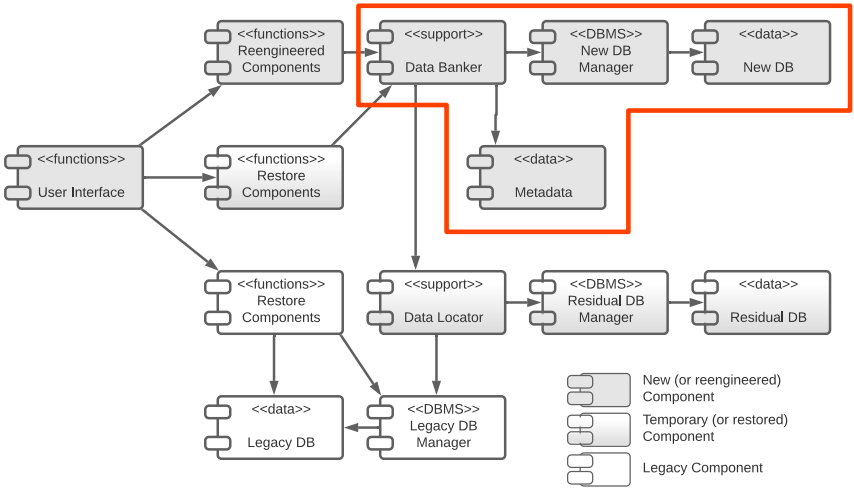
- The structure of **New DB** is known to **Metadata** and the services of the **New DB Manager** are known to **Data Banker**



ROAR

System Architecture

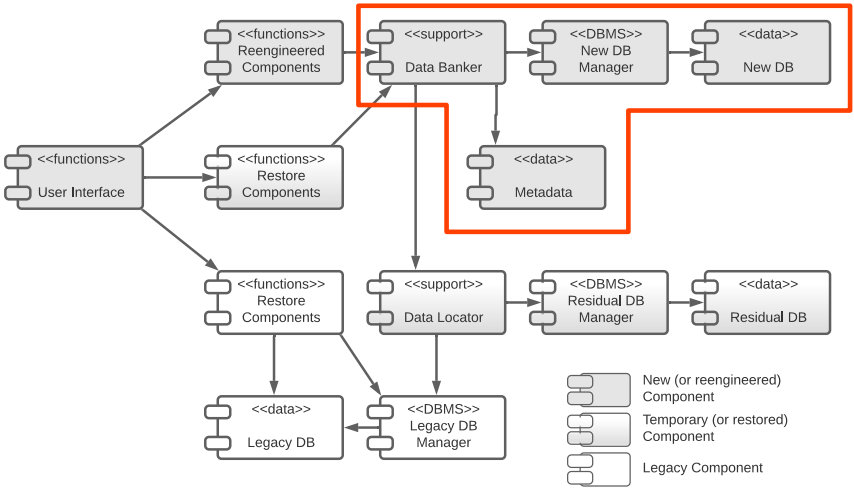
- Therefore, if there are changes to the physical structure of the **New DB**, only the contents of **Metadata** need to be modified.



Idaho State University
Computer Science

System Architecture

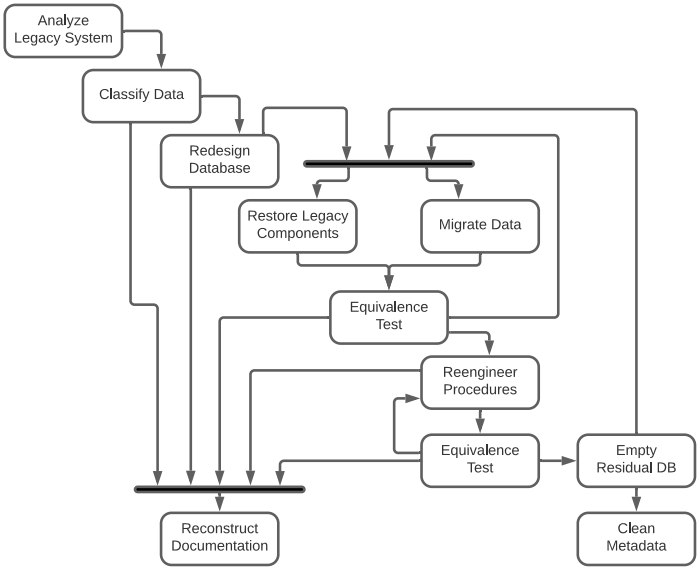
- On the other hand, changes to the **New DB Manager** requires changes to the **Data Banker** alone.



ROAR

Idaho State University
Computer Science

Migration Process

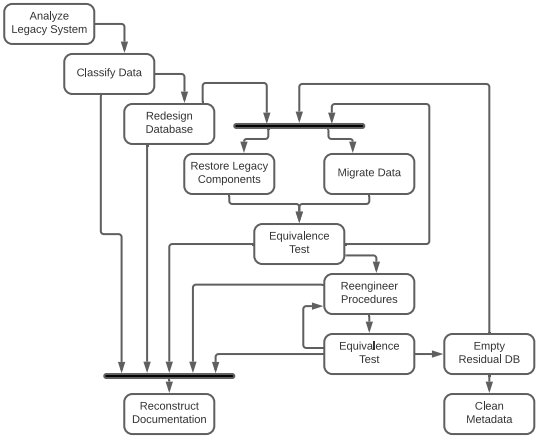


ROAR

Migration Process

Analyze Legacy System

- CRs influencing a component to be reengineered are frozen until that component is reengineered
- System is partitioned into components to minimize their client-supplier relationships
  1. to minimize the number of change requests (CRs) to be frozen
  2. to minimize the time durations for which CRs are frozen
  - Partitioning into components, such as to minimize the impact of reengineering



Idaho State University
Computer Science

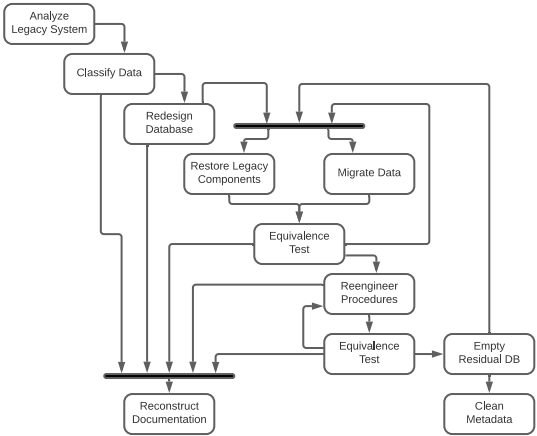
Migration Process

Classify Data

- Categories: primary or residual data.
- All non-duplicated data in the Legacy DB are recorded in a table produced by data classification

Redesign Database

- Data is then restructured
  1. to reorganize the new database for more effective and efficient access
  2. to eliminate all the defects in the legacy database
  3. to eliminate redundant data
- While describing the mode of access for the new database, define the characteristics of the data to be stored in the database for Metadata



ROAR

Idaho State University
Computer Science

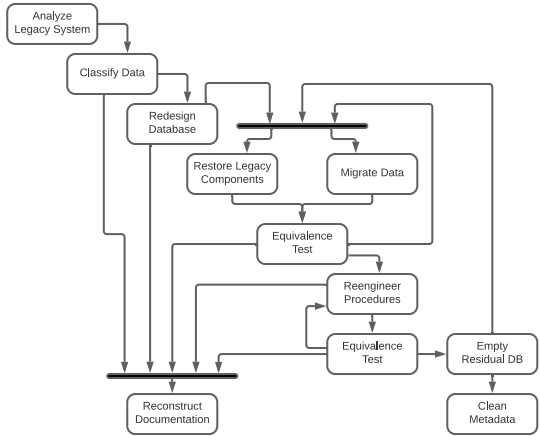
Migration Process

Restore Legacy Components

- Legacy system is made compatible with the new database.
  - Data access code is identified
  - Identified code is replaced with code that calls the Data Banker

Migrate Data

- Legacy database is incrementally migrated to new database
  - Non-essential data -> the residual database.
  - Data migration tool can be developed for this purpose.
- For each data file to be reengineered, the tool must:
  - read all the data it contains
  - copies the data into the Residual DB or New DB based on the Metadata.



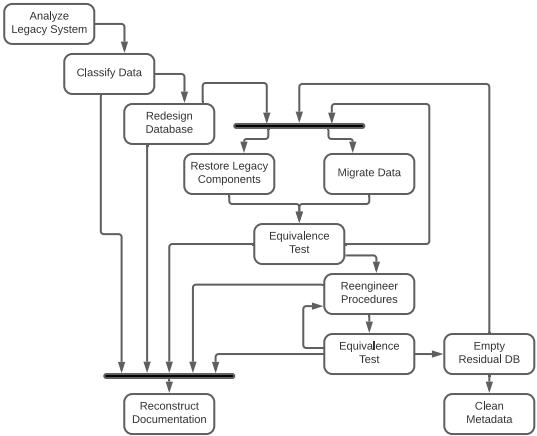
ROAR

Idaho State University
Computer Science

Migration Process

Reengineer Procedures

- Functions are reengineered in this phase, thereby evolving them from the restored state to the reengineered state.
- Quality deficiencies of the procedures are analyzed and suitable remedies are introduced.
- While reengineering, original components of the legacy system are reused as much as possible for two reasons:
  1. reduce the cost of reengineering.
  2. preserve the skills that the maintainers have developed.



ROAR

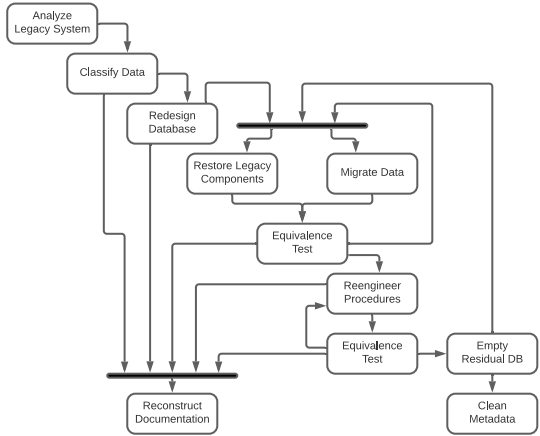
Migration Process

Equivalence Tests

- Ensures that the system continues to execute the same way as it did before
  - Tests are conducted after:
    - procedure restoration
    - data migration.
- Test plan documentation is updated

Empty Residual DB

- In iterative reengineering:
  - Parts of the legacy DB -> the New DB
  - Remaining parts -> Residual DB
  - Useless/redundant data in Residual DB are removed
- At Completion:
  - Residual DB will empty
  - Residual Data Manager and Data Locator are removed from the system.



Idaho State University
Computer Science

Migration Process

Iteration

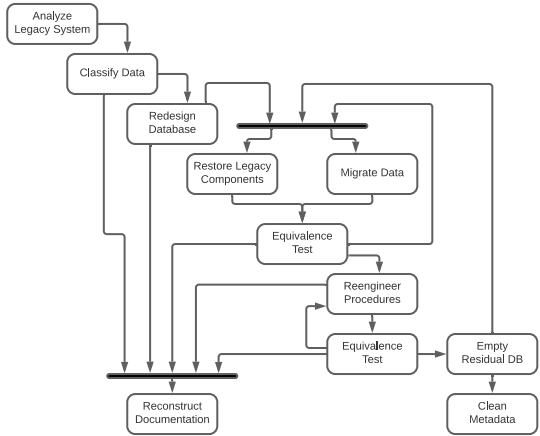
- Legacy components are reengineered one at a time
- Process repeats, for each component, until the entire system is reengineered.

Clean Metadata

- At completion: Metadata only reflects the New DB contents
- Residual DB metadata is not needed and should be eliminated
- All Data Banker procedures are removed

Reconstruct Documentation

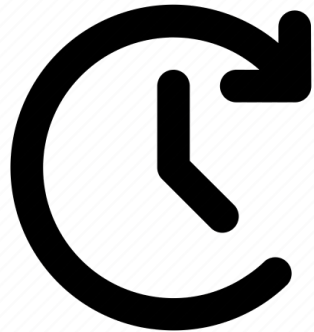
- Proceeds concurrently with the actual migration
- Updating documentation should performed throughout the process



ROAR

For Next Time

- Review EVO Chapter 5.5.6 - 5.6
- Read EVO Chapter 6.1 - 6.3
- Watch Lecture 14



ROAR

Are there any questions?