Evolutionary Design:
System Scale (Part 2)

Andy Carpenter

School of Computer Science

(Andy.Carpenter@manchester.ac.uk)

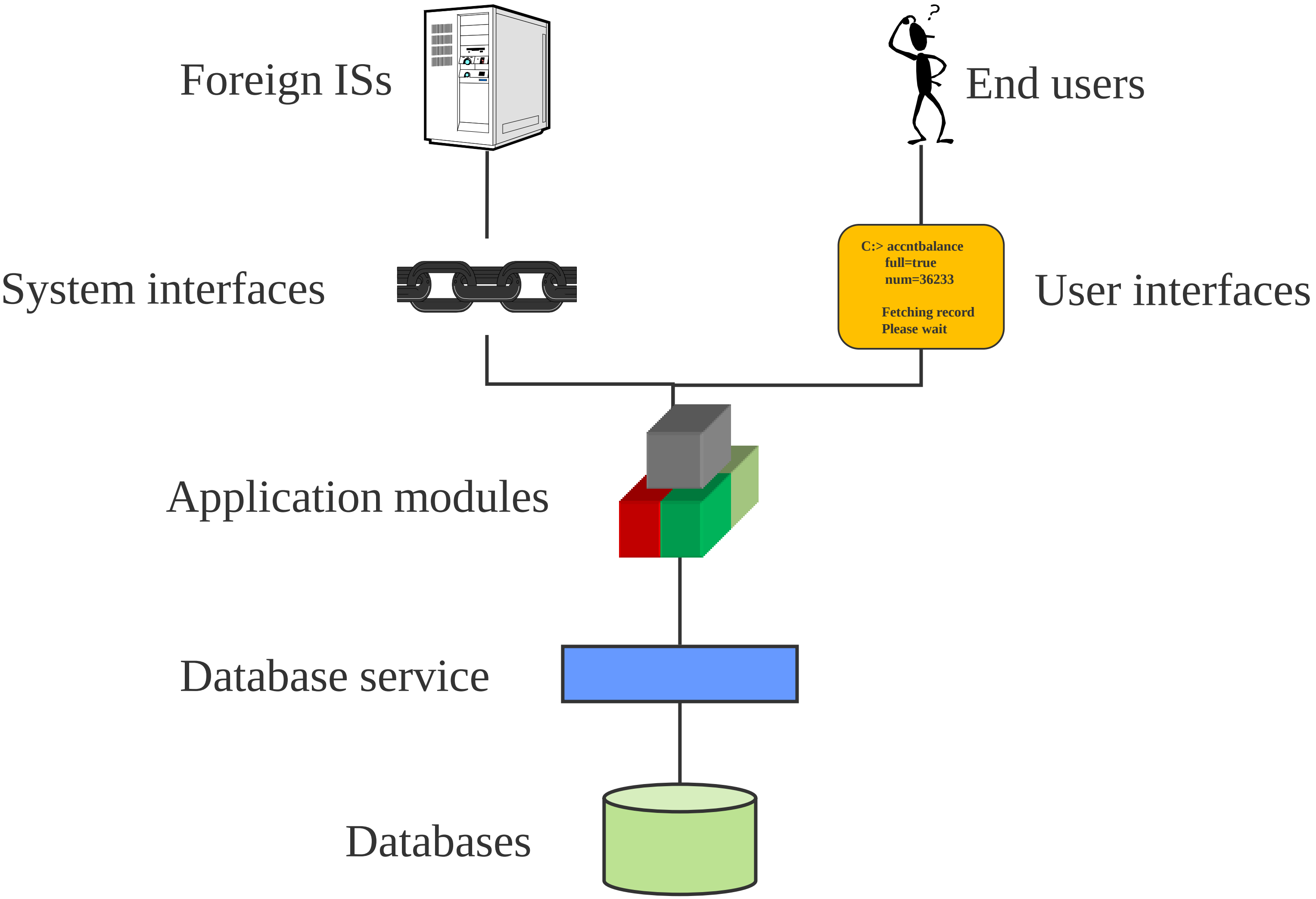
Based on slides from Suzanne Embury

Incremental Migration

- When we replace a component
  - other existing (legacy) components must continue to operate as before
  - new component must operate as if in the completed target environment
- Potential IS components
  - interfaces: with humans, with other systems
  - applications: performing business functions
  - data management service
- Components may be modularised or mixed up together
- Ease of migration depends on architecture of legacy IS

Software Architecture

Decomposable IS Architecture



Foreign ISs

End users

System interfaces

```
C:> accntbalance
    full=true
    num=36233

    Fetching record
    Please wait
```

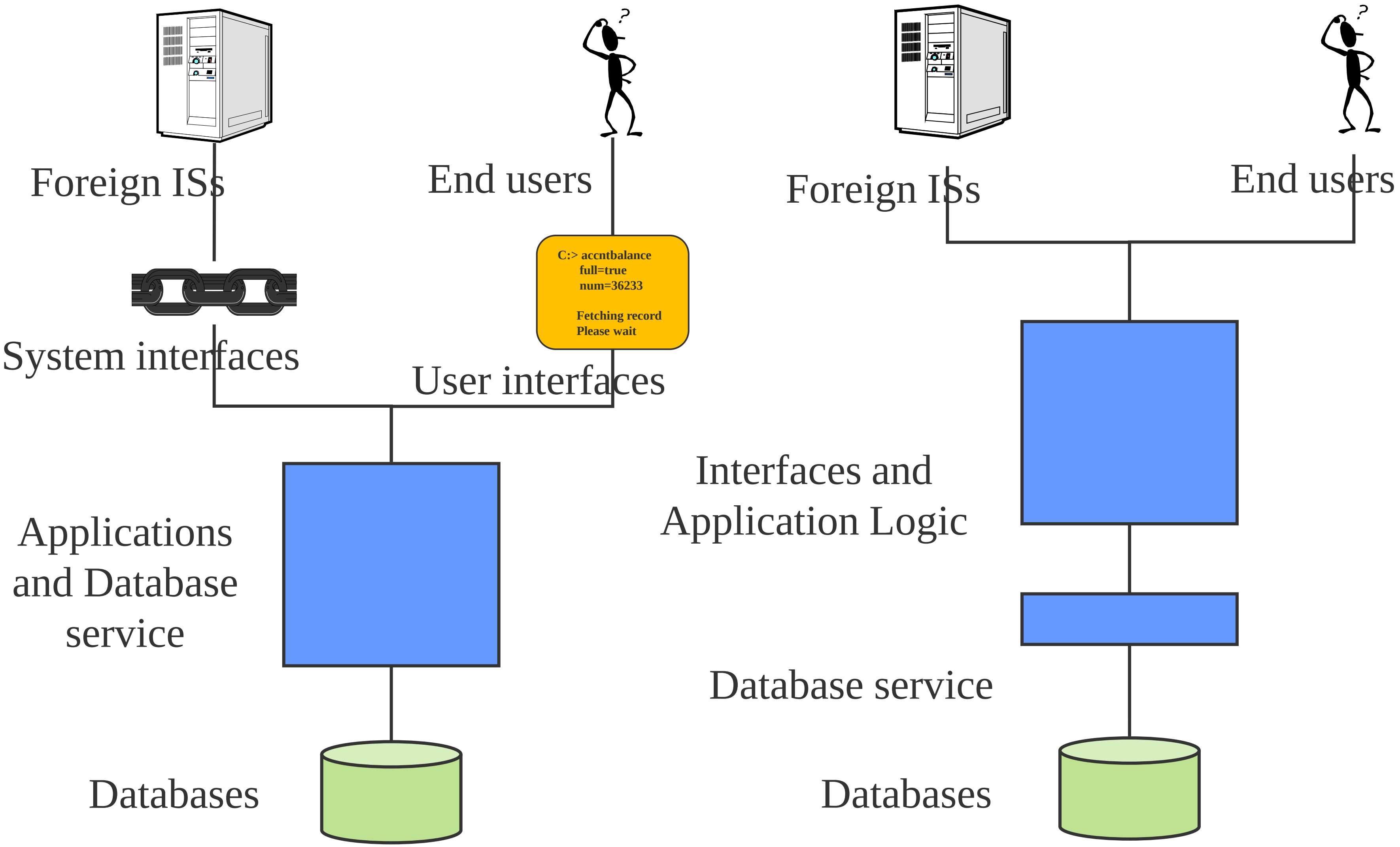
User interfaces

Application modules

Database service

Databases

Semi-Decomposable



Foreign ISs

End users

System interfaces

```
C:> accntbalance
     full=true
     num=36233

    Fetching record
    Please wait
```

User interfaces

Applications and Database service

Databases

Foreign ISs

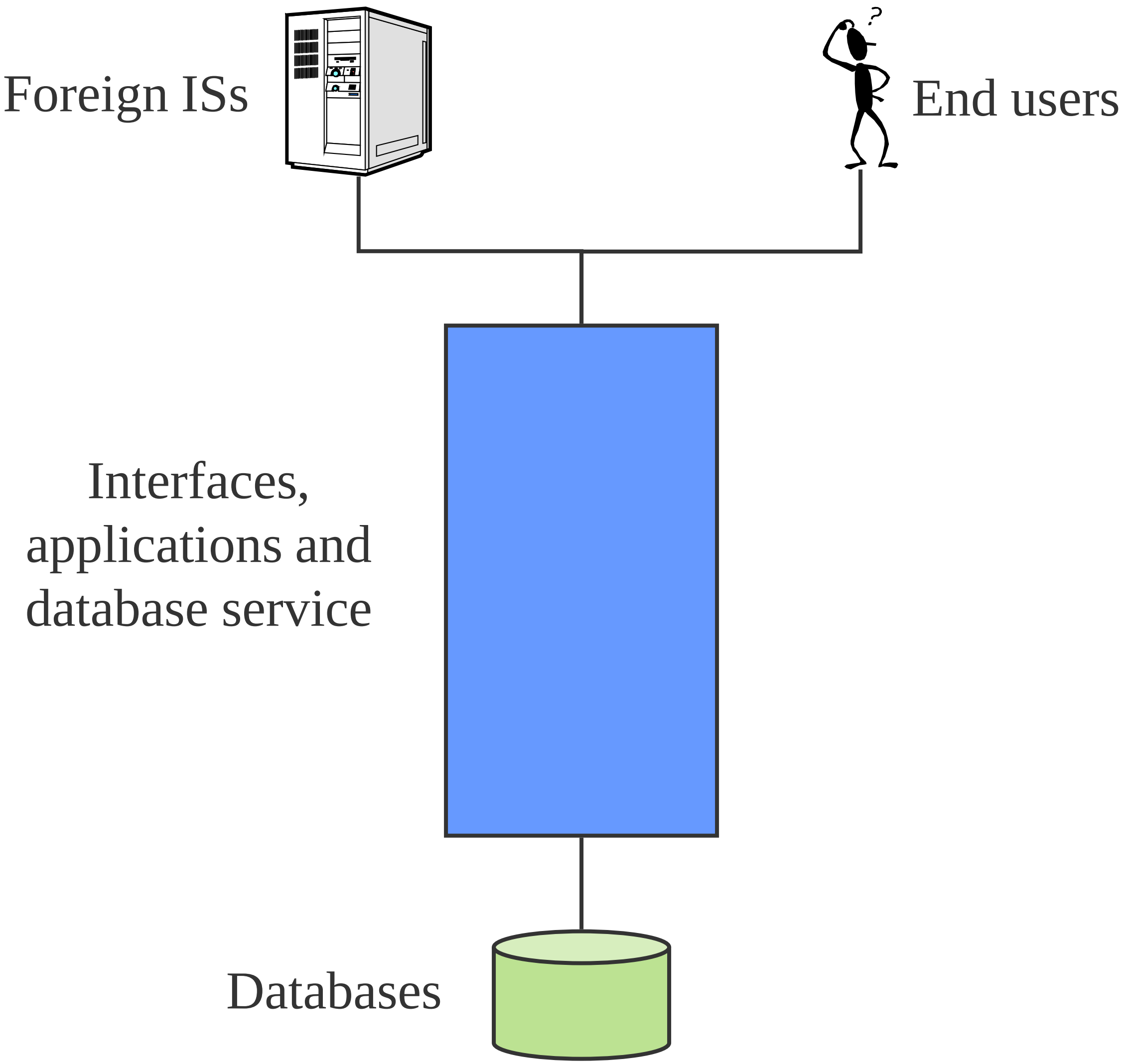
End users

Interfaces and Application Logic

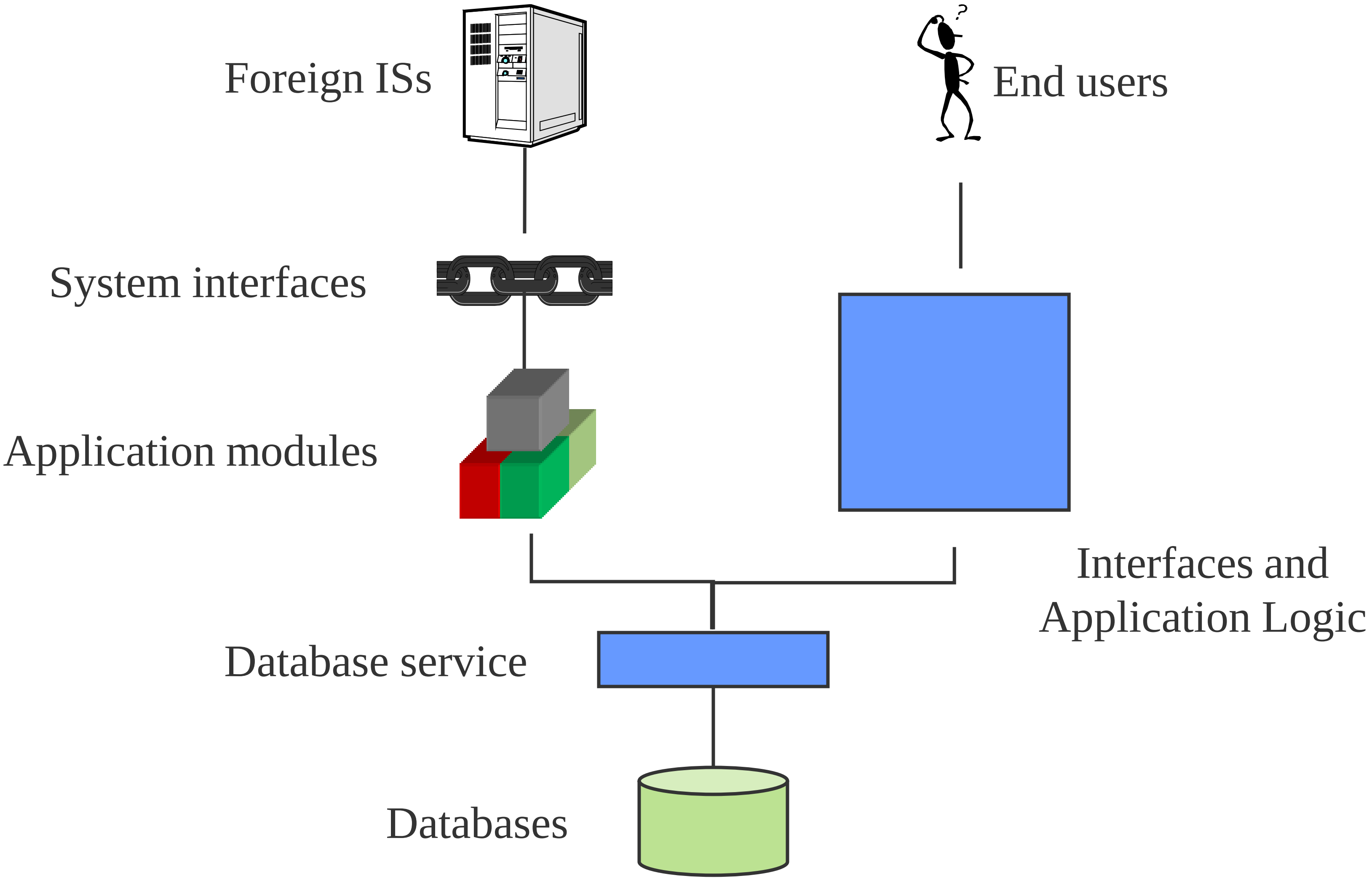
Database service

Databases

Non-Decomposable IS Arch.



Hybrid IS Architecture

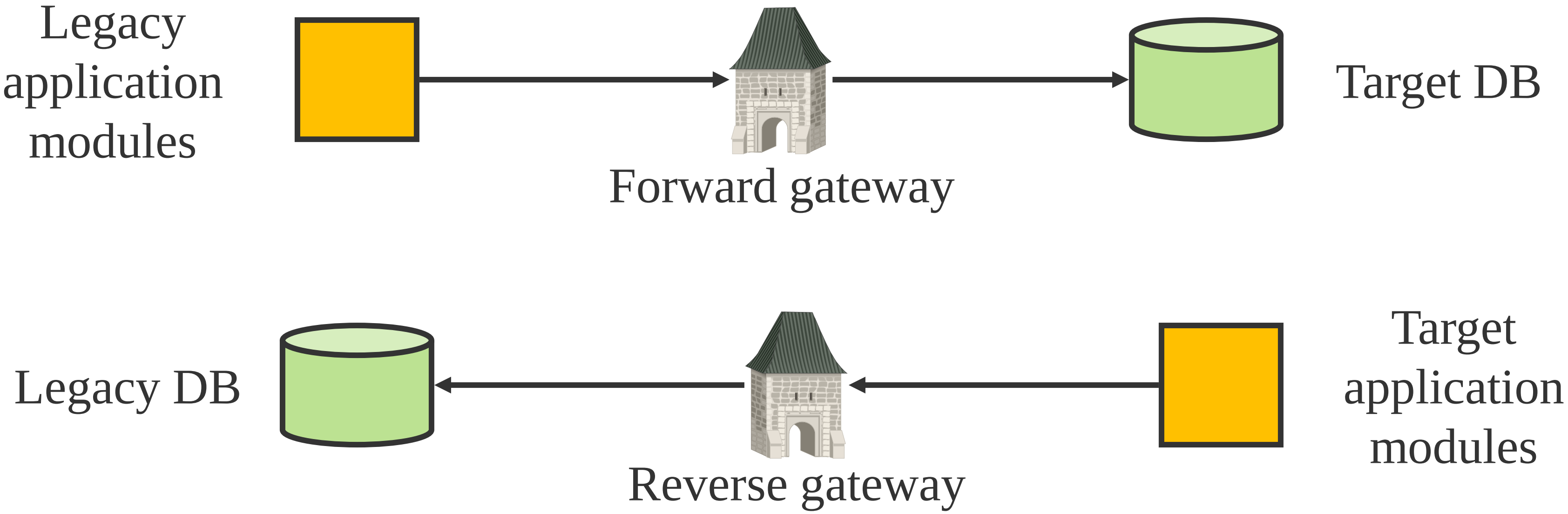


The Target IS

- Designed to be fully decomposable
- During migration, we select elements of the legacy IS, for incremental re-engineering in the target IS
  - Business benefits/urgency of change
  - Technical difficulty of change
  - Costs of maintaining the migration
- Until migration is complete, organisation’s IS is hybrid of
  - legacy IS and partially constructed target IS
- Both work together to keep the business functioning during the migration
- How?

Key Component: Gateway

- Gateways decouple components from their context of use.
- Allows component to operate, even while components around it are changing.



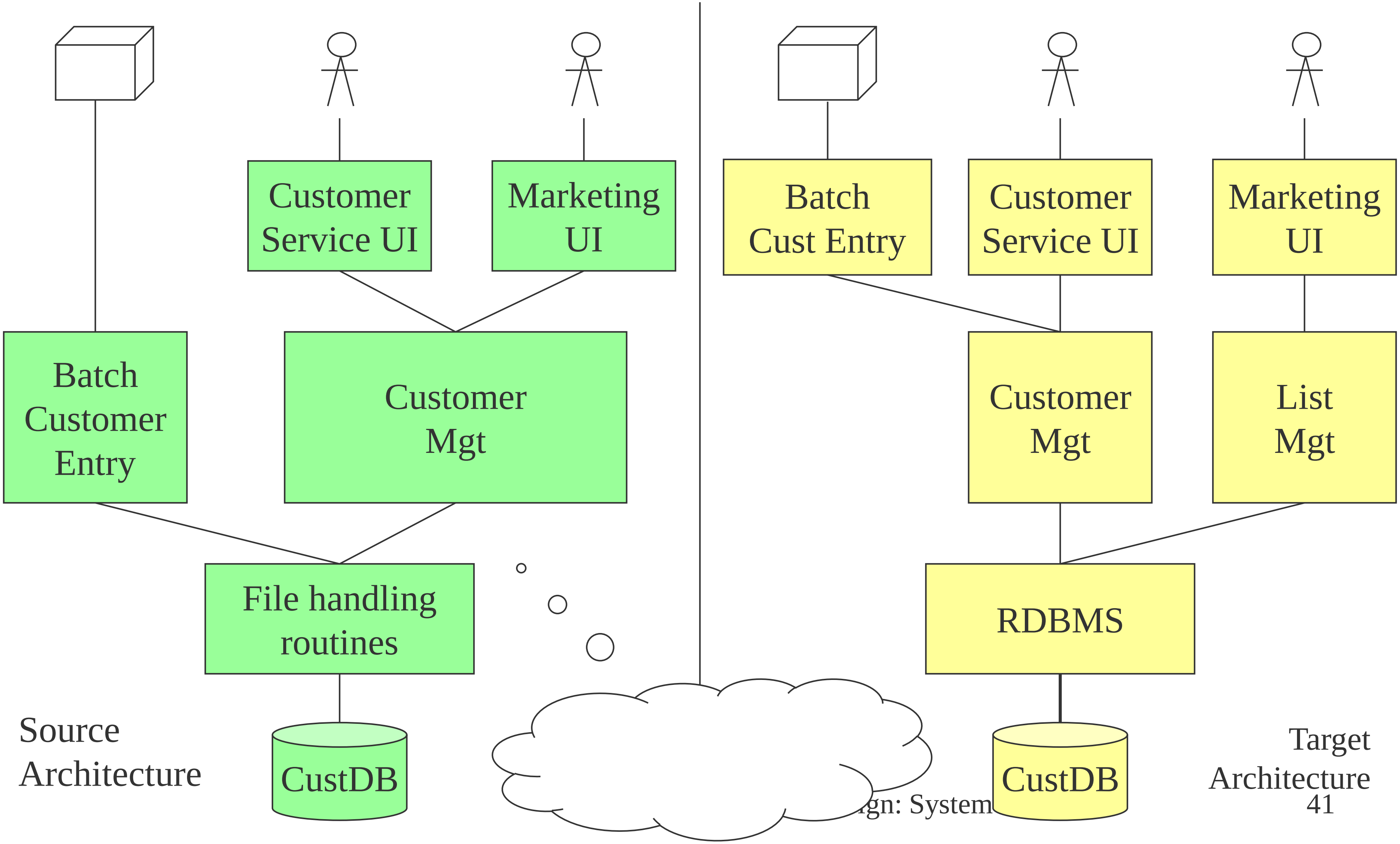
Role of Gateways

- Insulate components from changes made in other components
  - e.g. legacy UI used with target application modules
- Translate requests/responses between formats required by components
  - e.g. COBOL data structures to relational data
- Coordinate updates made through different components, to maintain consistency
  - e.g. propagate updates made through target UI to legacy IS database

Example Migration

- XYZ plc wishes to migrate its customer information system away from COBOL and IDMS, towards a layered, fully decomposable architecture based on an RDBMS
- This requires
  - Replacement of UIs with web-based GUIs
  - Additional marketing functionality to be added to standard account management applications
  - New database, based on relational technology, but
    1. no new data is to be stored as yet
    2. new data is required for the new functionality

Incremental Migration Example



Migration Strategies

- The first task is to determine whether the legacy IS is decomposable or not
  - In this example: a hybrid of fully decomposable and semi-decomposable

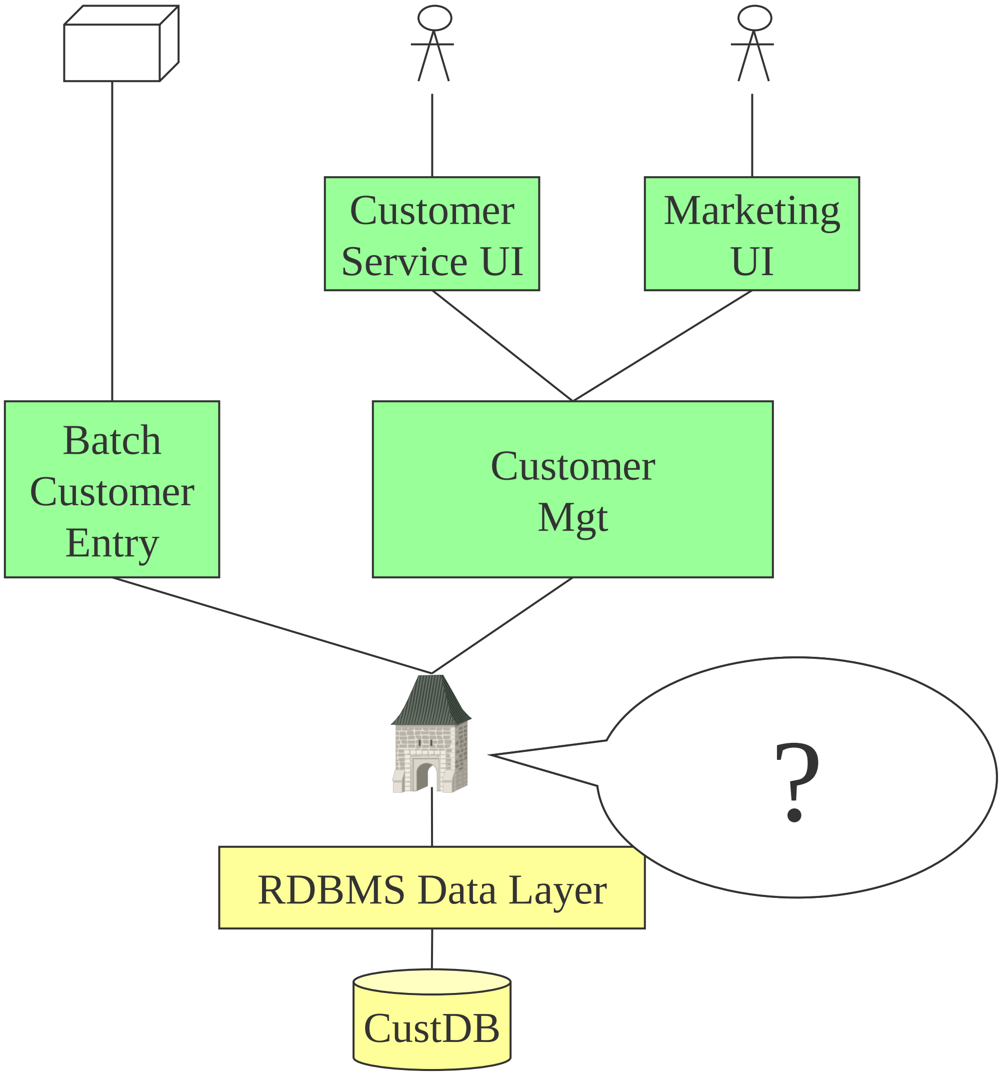
- Next, we must select a migration strategy
  - Forward migration
  - Reverse migration
  - General migration

Forward Migration

- First, migrate the DB in one step
  - Create and load target DB
  - Create forward gateway
  - Install gateway to divert legacy applications to target DB, and away from legacy DB

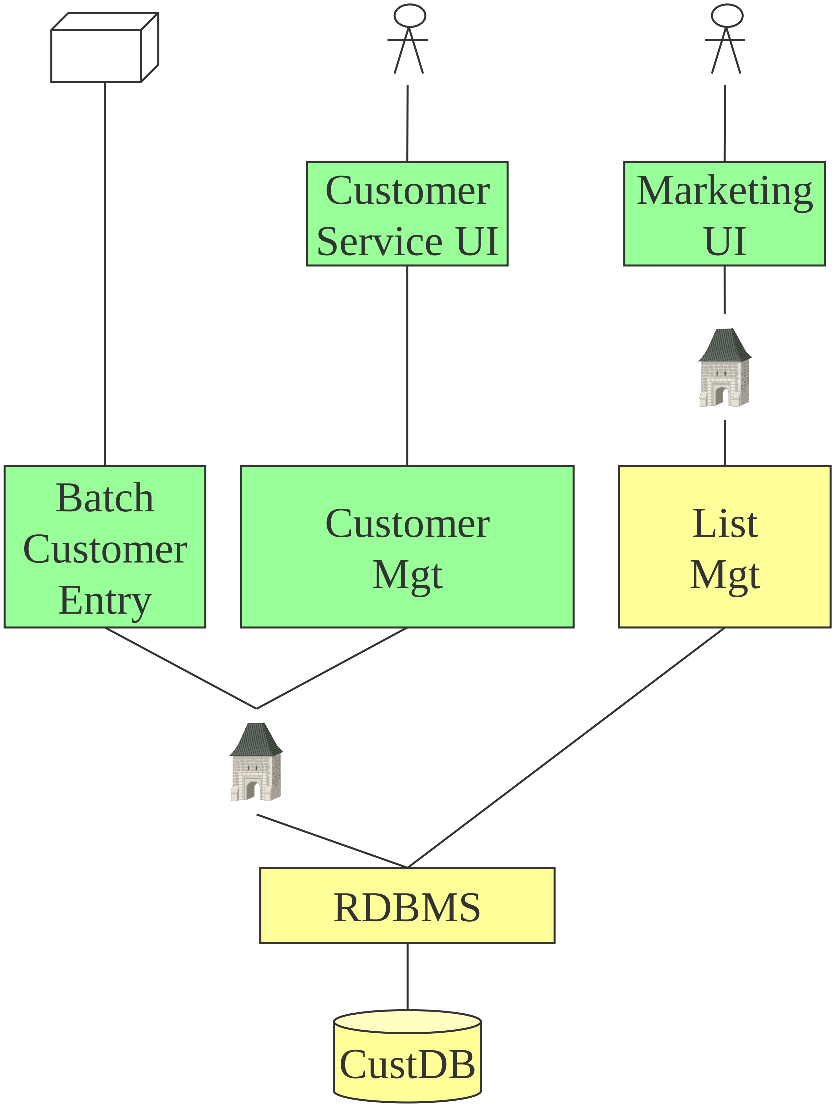
  - Now entire IS works as before, but accesses and stores data in the target DBMS, not in the legacy DBMS

Forward Migration: Step 1



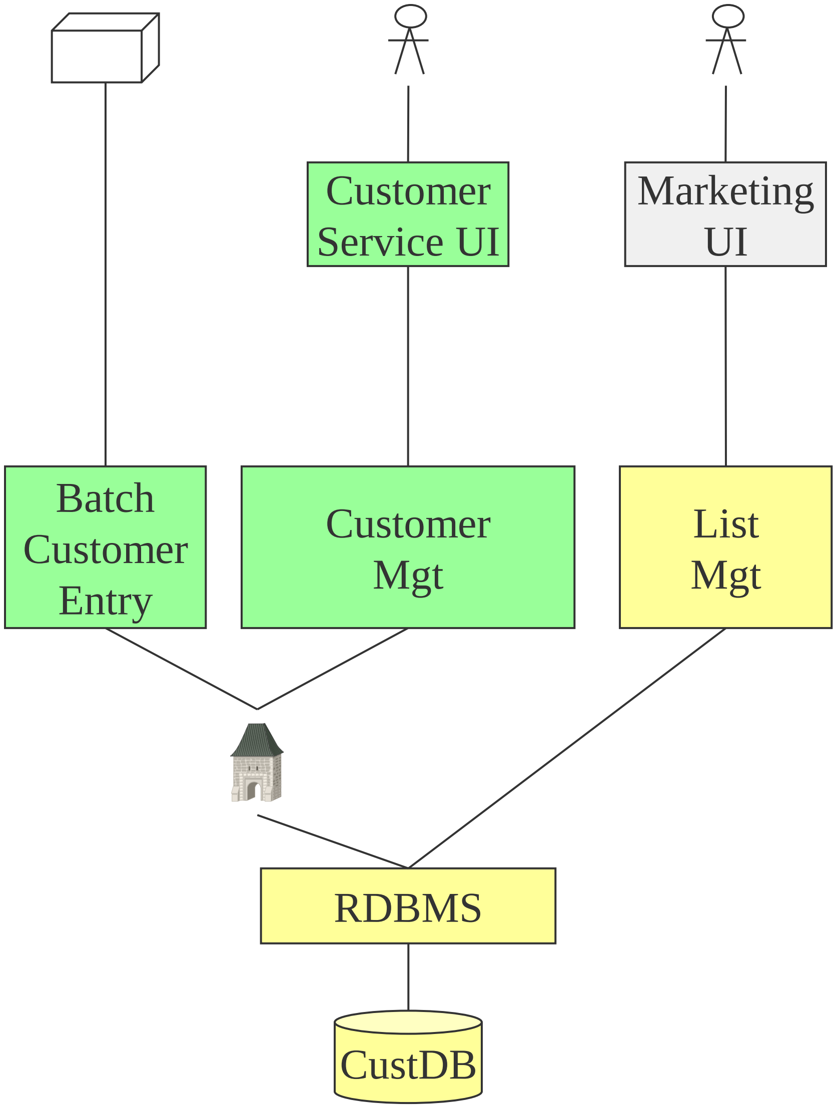
Forward Migration: Step 2

- Next, incrementally migrate the application modules and their interfaces, removing legacy applications as you go
- Which module to migrate first?



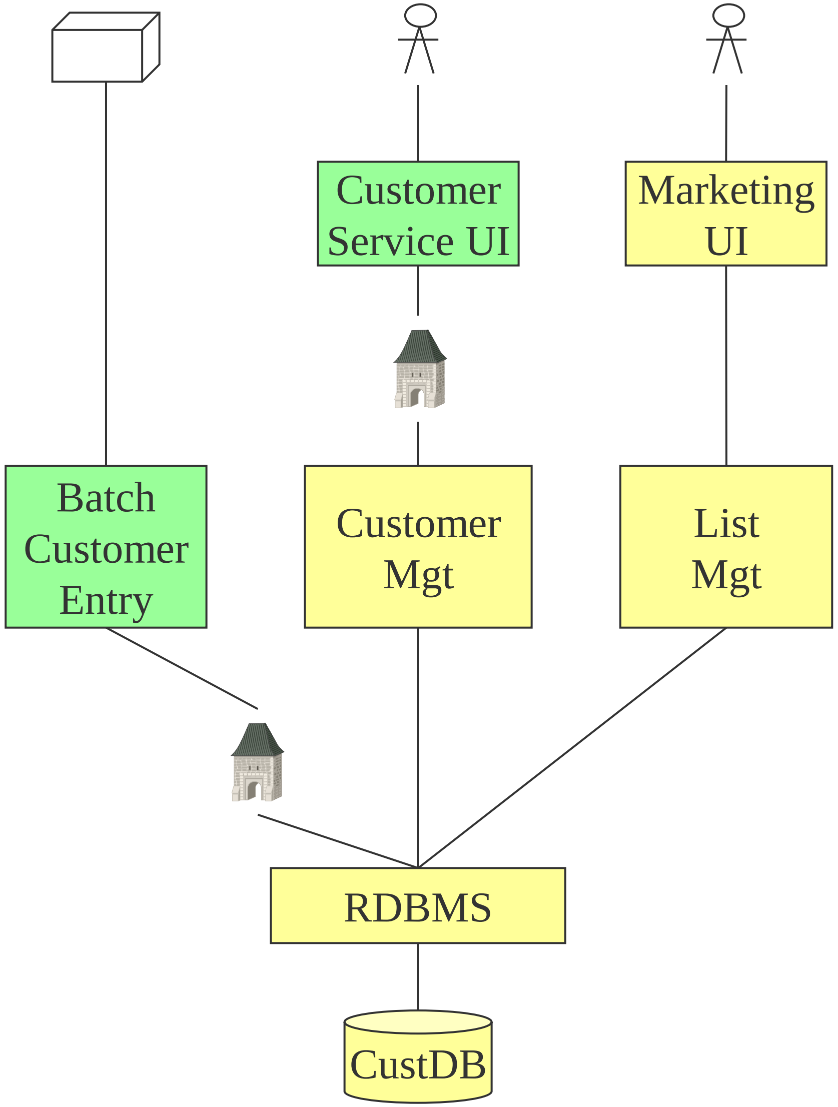
Forward Migration: Step 3

- Which module to migrate next?



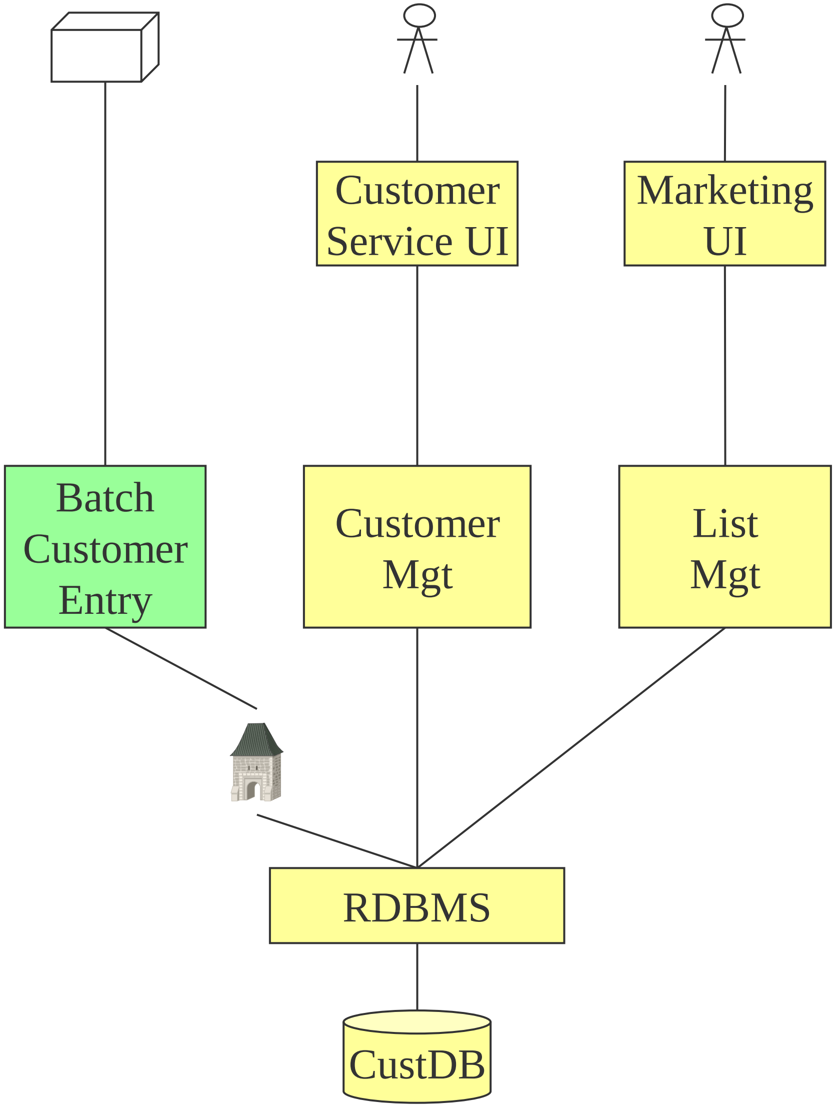
Forward Migration: Step 4

- Which module to migrate next?



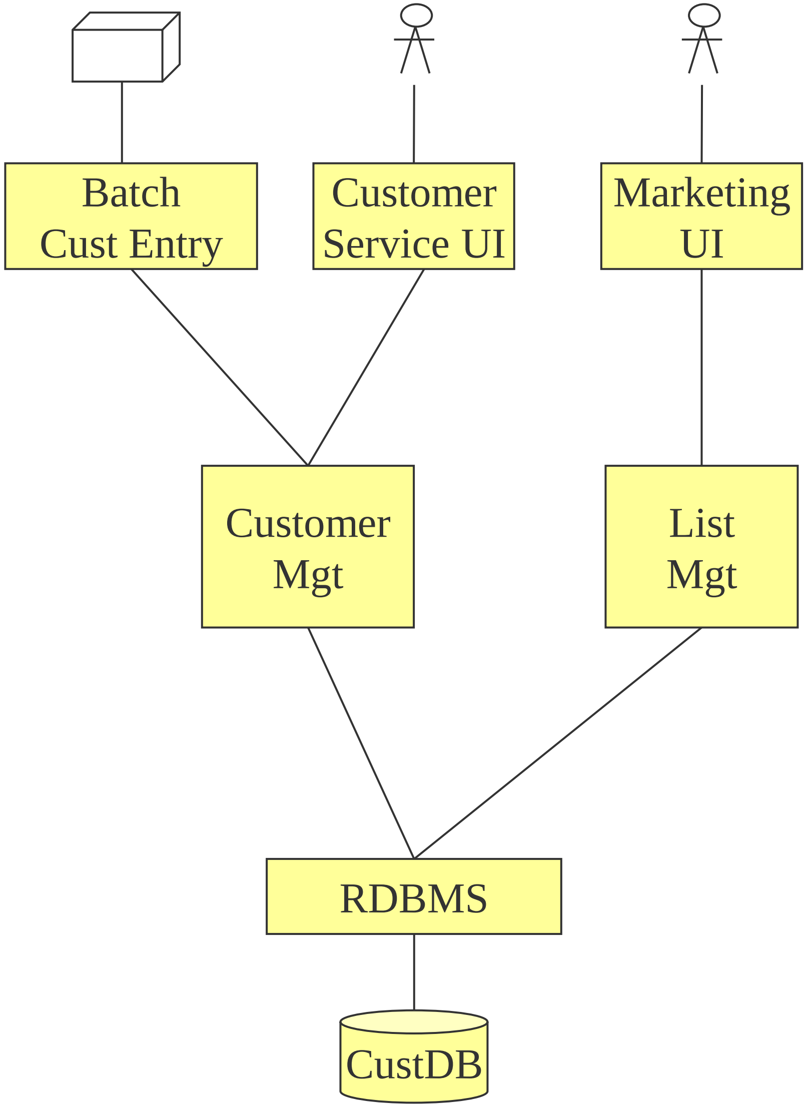
Forward Migration: Step 5

- Which module to replace next?



Forward Migration: Step 6

- Finally, when all legacy components are replaced and all gateways are removed, the migration is complete



Reverse Migration

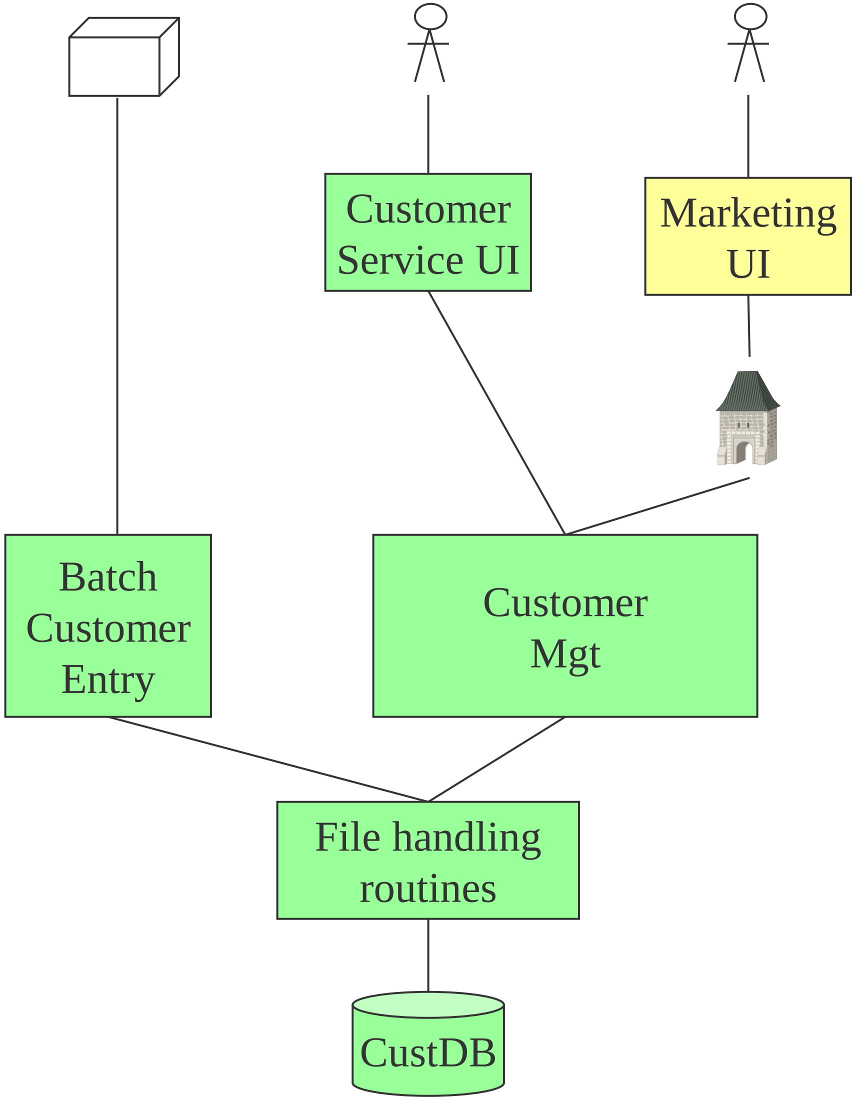
- First, migrate the target interfaces and applications
  - Create and install target applications + interfaces
  - Create reverse gateway
  - Install gateway to divert target applications to the legacy database

- Now entire IS provides some new functionality, but is still operating to the original legacy database

Reverse Migration: Step 1

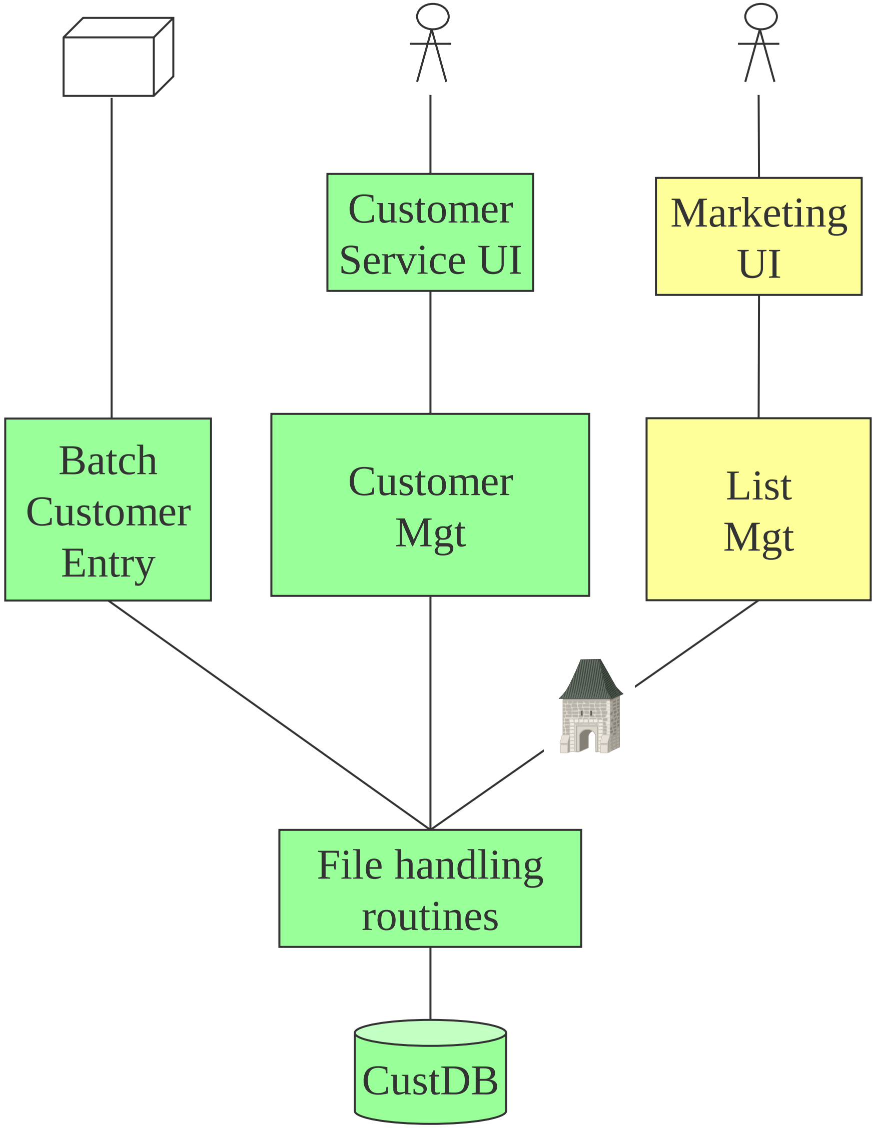
- First, migrate application interfaces
- Install reverse gateways to translate requests for legacy components

- Which component to migrate first?



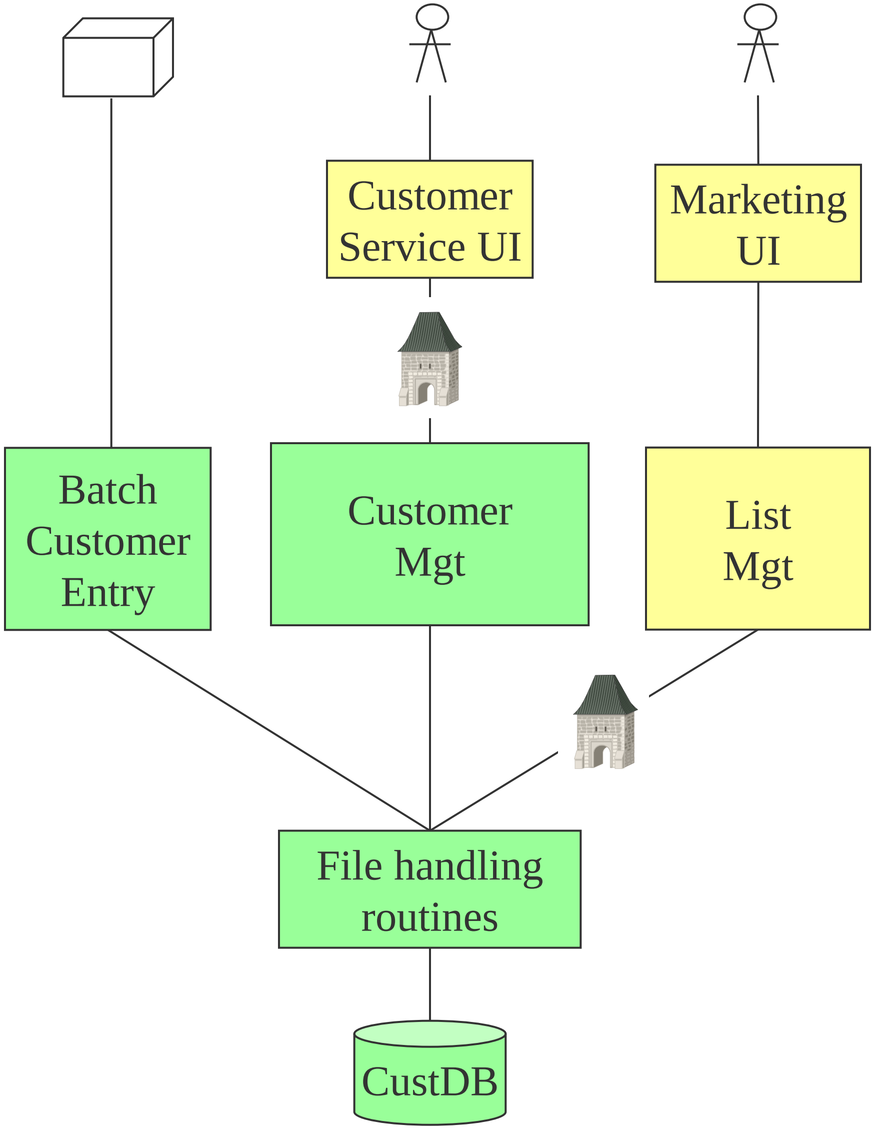
Reverse Migration: Step 2

- Next, incrementally migrate downwards, installing more reverse gateways as required



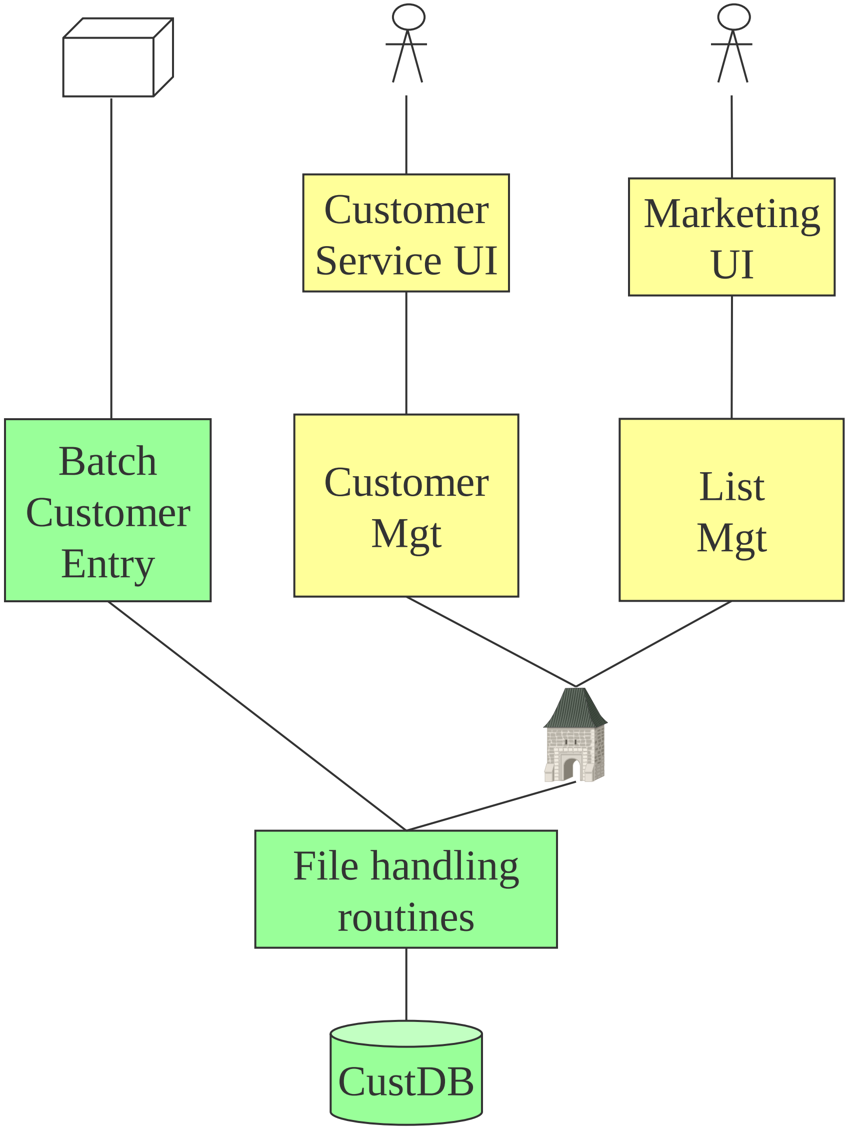
Reverse Migration: Step 3

- Which component to migrate next?



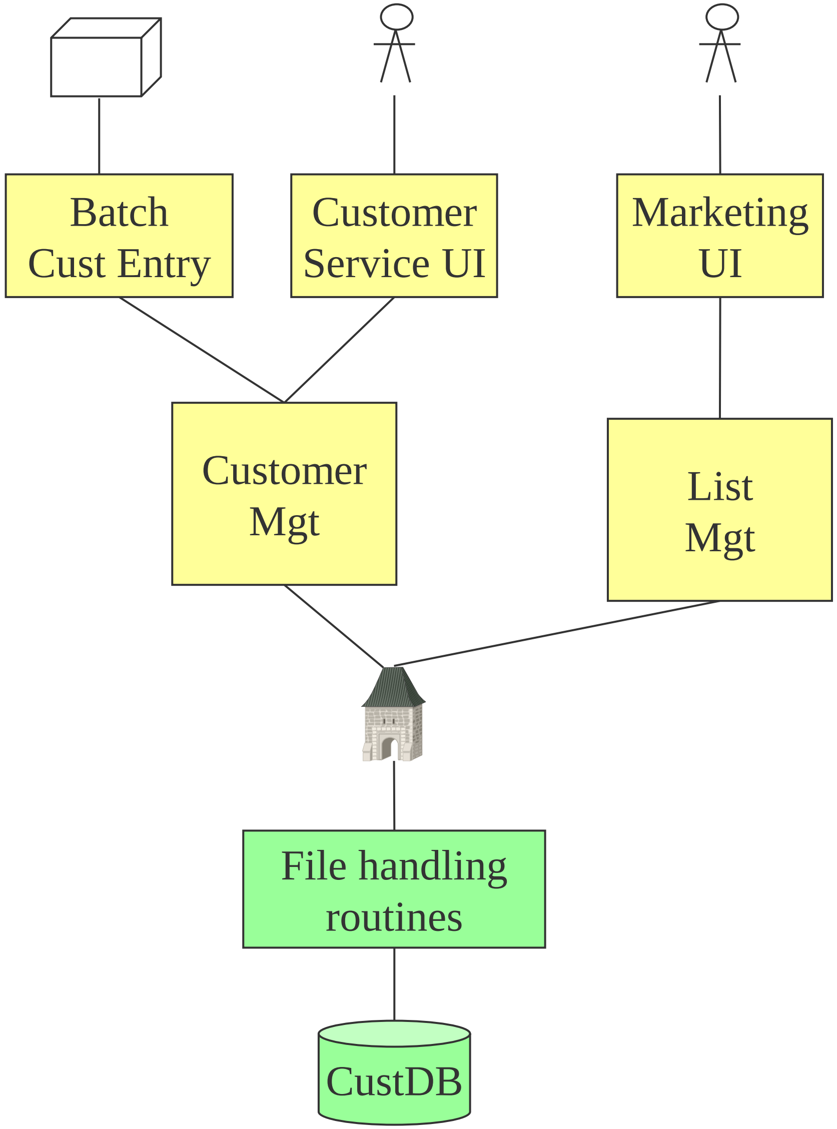
Reverse Migration: Step 4

- Which component to migrate next?



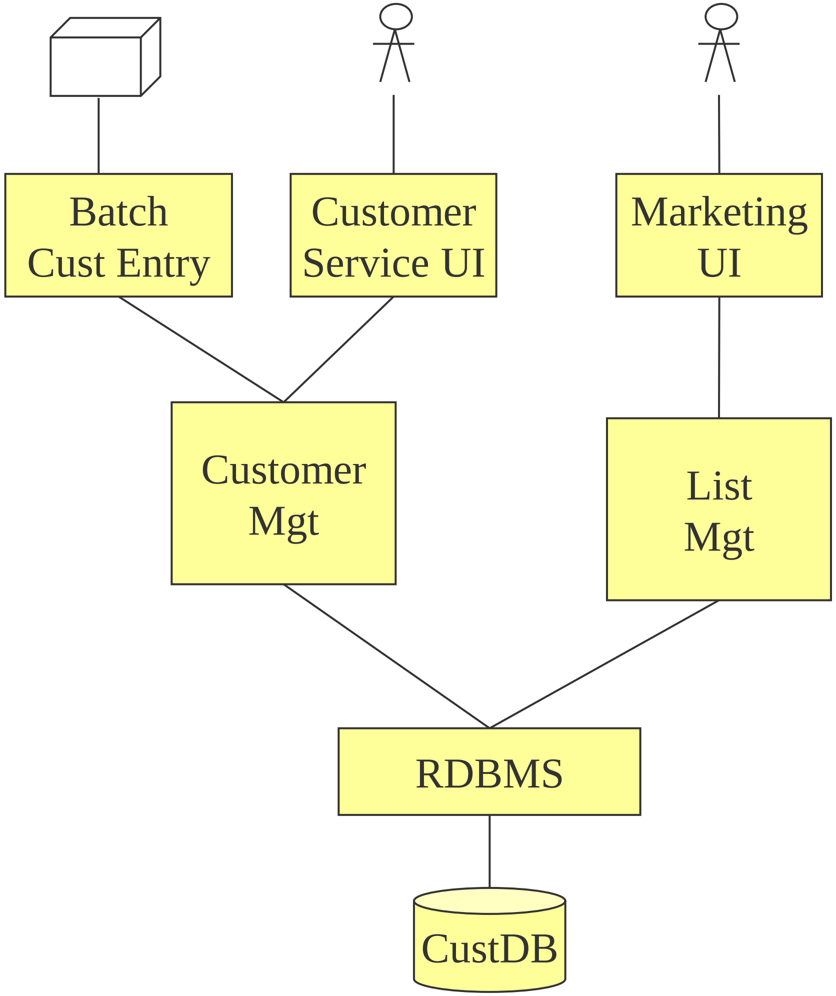
Reverse Migration: Step 5

- Which component to migrate next?



Reverse Migration: Step 6

- Finally, migrate the legacy data to the new data management system, and remove the last gateway



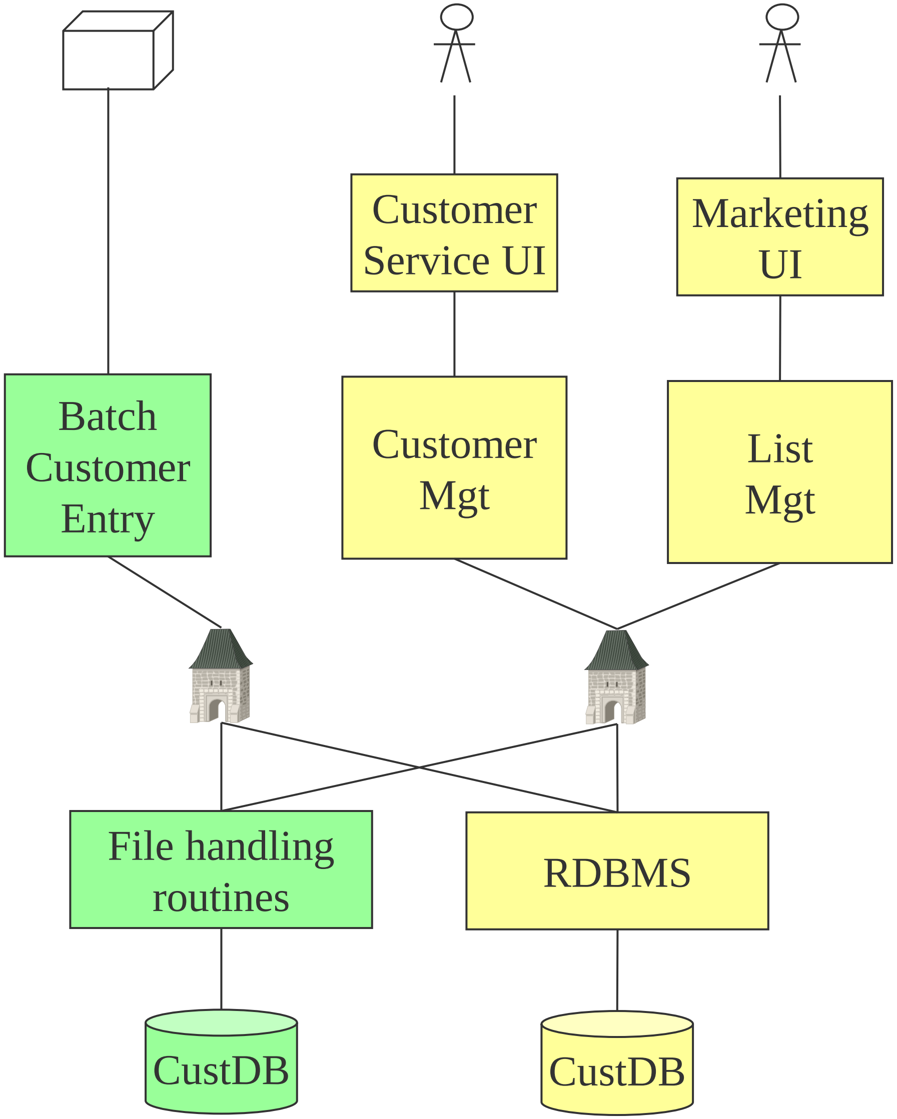
Forward vs Reverse Migration

- Both migration strategies took the same number of steps
- Both produced the same final system
- So, what’s the difference?
  - Benefits
  - Costs
  - Risks

General Migration

- Both forward and reverse migration require that the DB be migrated in one step
- This is risky – we would prefer to be able to migrate the DB incrementally, as well as the applications and interfaces
- By using a combination of these strategies, we can achieve this, at the expense of greater complexity of the system (plus additional effort writing and maintaining gateways)

Example Architecture



Evolution in the Long Term

- In the examples, the gateways were always removed for the final system
- In reality, some gateways may become permanent parts of the system
  - Because we don’t dare to retire the legacy component
  - Because we never complete the planned migration due to changing business requirements
- Allows systems to be continuously evolved, over the long term