System Integration

Mini Case Studies © 2010

Architectural Mismatch

Shawn A. Butler, Ph.D.
Senior Lecturer, Executive Education Program
Institute for Software Research
Carnegie Mellon University

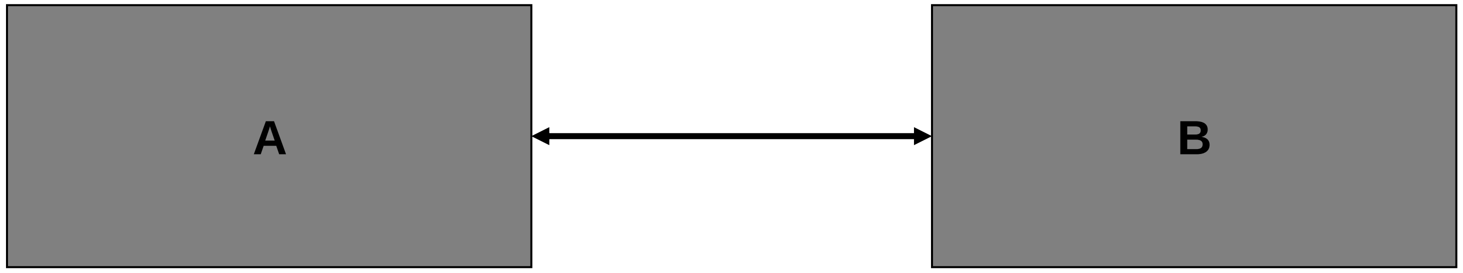
Objectives

- Understand the types of architectural mismatches that can occur during a system integration project
- Understand some of the advantages and disadvantages of reconciling mismatches
- Discuss security issues separately

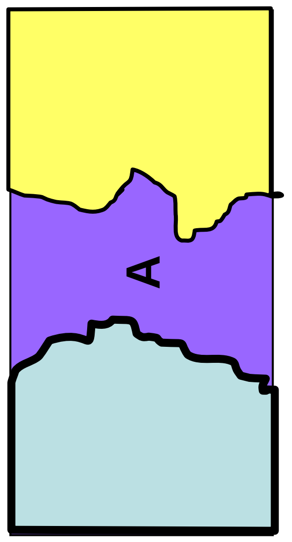
Assumptions

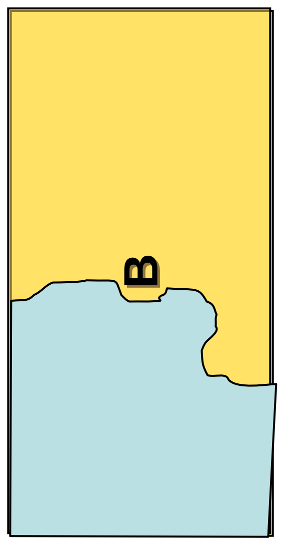
Students should have completed the readings before viewing this lecture

Basic Components and a Connector



Two Applications with Overlapping Functionality





Two Applications with Overlapping Functionality

- Applications may have very similar, but slightly different functionality
- Application design has tightly coupled sub-components – Not easy to break apart
- Each sub-component may make assumptions about other subcomponents
- Redundancy in functionality adds maintenance overhead
- Sub-component not designed to be reused
- Especially problematic with redundant services

Possible Solutions

- Choose one application and extend functionality to meet total functionality
- Refactor one (or both) of the applications so that specific functionality can be extracted and then integrated
- Create API’s to increase sub-component independence
- Wrap components and create new interfaces or clients
- Ignore functionality

There aren’t any easy or cheap solutions to this problem

Platform Compatibility Problems

- Big-Endian/Little-Endian
- System/file calls
- Assumptions about environment
  - Existing applications
  - Order of installation
  - System variables
  - Passwords

Big-Endian/Little-Endian

How are you?

$\longrightarrow$

or
こんにちは

$\longleftarrow$

Big Endian => big bytes first

Little Endian => small bytes first

Number represented = 024F32D1

02|4F|32|D1	D1|32|4F|02
024F|32D1	32D1|024F

Possible Solutions

- Software switch on computer, if feasible
- Marshall data before transmitting – unmarshall at the receiving end

Platform Compatibility Problems

- Big-Endian/Little-Endian
- System/file calls
- Assumptions about environment
  - Existing applications
  - Order of installation
  - System variables

System Calls and File Systems

- System Calls
  - C:/MyDocuments/MyPictures/dog.jpg
  - \myunixfiles\dog.jpg
  - ls versus dir

Possible Solutions

- Middleware that maps physical locations to logical locations
- System level checks to ensure platform compatibility

Platform Compatibility Problems

- Big-Endian/Little-Endian
- System/file calls
- Assumptions about environment
  - Existing applications
    - Does an application assume that a specific version of interpreter, OS, etc. exists
  - Order of installation
    - Must an application have another application loaded first
    - Complete clean up when de-installation occurs
  - System variables
    - Conflicting System Variable names
  - Passwords
    - Default passwords anticipated

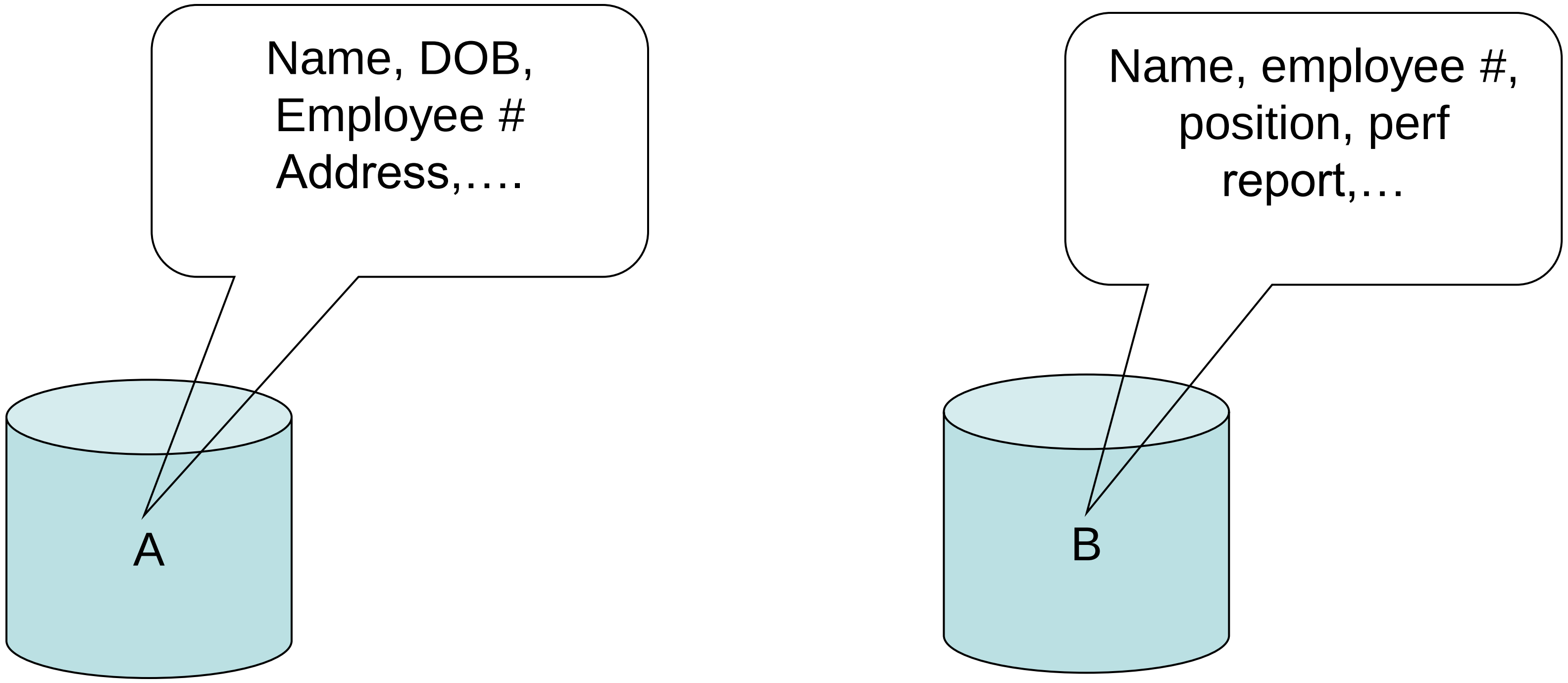
Possible Solutions

- Integration & Run Time Specification (DII COE)
  - Defines how modules behave during runtime
  - Resolves run time conflicts
  - Can be expensive to convert legacy
- Develop local guidelines on run time behaviors

Data Redundancy Issues

- Maintain same data in several places
- Synchronization of data
- Extra storage requirements
- Data formats differ
- Must discover all the data locations – Not all formats are in RDBMS
- Transition from old system to new system – Incompatible name servers

Data Overlap


Name, DOB, Employee # Address,….
A
Name, employee #, position, perf report,…
B

Possible Solutions

- Consolidate into one data base
- Change databases to common storage formats
  - May have to negotiate with several data owners
- Change data into common formats dynamically
- Develop a middleware component with business logic that knows where the data is and how to access data (consistent versions)

Data Transmission

- Synchronous versus asynchronous
- Different Formats
- Distribution
  - Broadcast
  - Point to point
- Different rates of transmission
  - Faster versus slower
  - Periodic versus aperiodic

Possible Solutions

- Data translators
- Develop synchronous/asynchronous mechanisms to match requirement
- Create message service

Other Issues

- Interfaces
  - User
  - Application
- Control over mismatched components
- Complexity
- Enterprise constraints conflicting with commercial product functionality

Summary

- Application overlapping functionality is expensive to resolve
- Architectural mismatches are most commonly found when integrating legacy systems
- Current technologies reduce integration problems, but they are not eliminated
- Architectural mismatches add complexity and cost to the integration project