System Integration

Mini Case Studies © 2010

Integration Styles

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

Objectives

- Be able to identify the four different styles of application integration
- Understand the elements of a messaging system
- Become familiar with some of the problems associated with senders and receivers

Assumptions

- The application developers may not have designed with integration in mind
- Even if they did, they probably wouldn’t have anticipated your requirements
- The documentation is perfectly understandable in the minds of the original developers
- The interface documents long ago disappeared

How to Integrate Two (or more) Applications

Application A
Application B

Integration Guidelines

- Application Coupling – Minimize dependencies
  - Tightly coupled applications have known and unknown assumptions
  - Applications can evolve independently without problems
- Intrusiveness – Minimize changes in each application
  - Changes are often necessary
  - Tradeoff between intrusiveness and best integration design

Integration Guidelines Cont.

- Technology Selection – A variety of HW and SW integration tools
  - Expensive
  - Vendor Lock-in
  - Tradeoff between tools and reinventing
- Data Format – Agree on format of data exchange
  - Change the application or create a translator
  - How will the data change over time?
- Data timeliness – When does the exchange take place
  - Tradeoff large chunks versus small chunks of data
  - Real time - near real time – in time

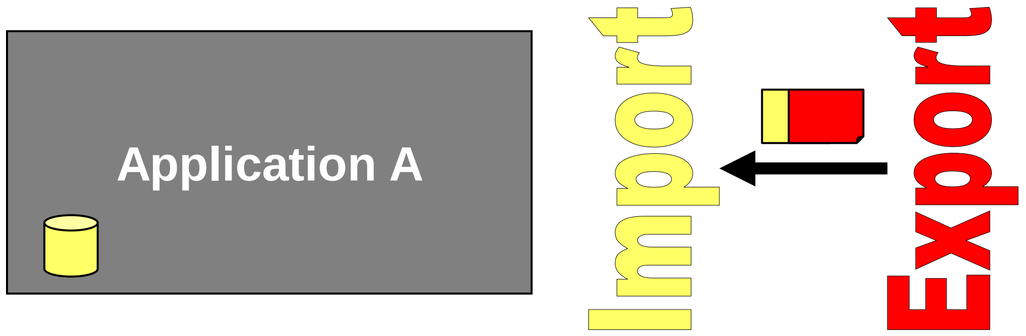
Integration Guidelines Cont.

- Data or Functionality – functionality provides better abstraction between applications
  - Local versus remote invocation
- Remote Communication
  - Synchronous and Asynchronous
  - Slower than local
- Reliability – Communication across networks is not as reliable as inter-application communication

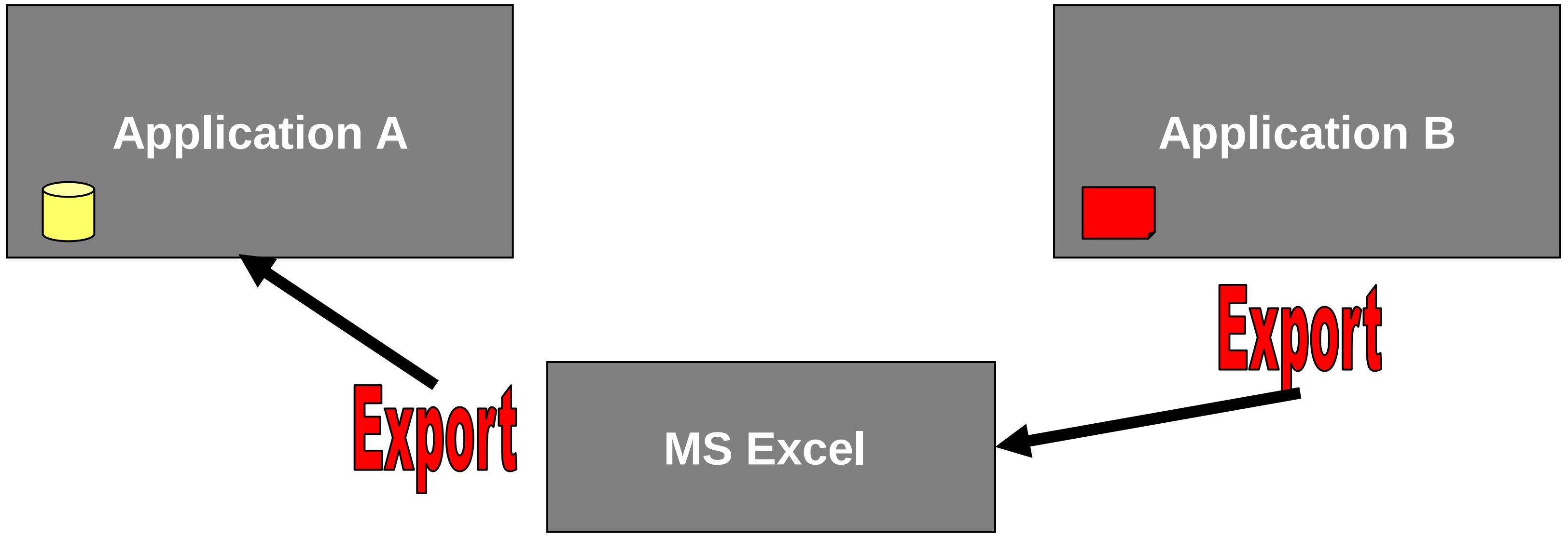
Integration Styles

- File Transfer

File Integration



File Integration



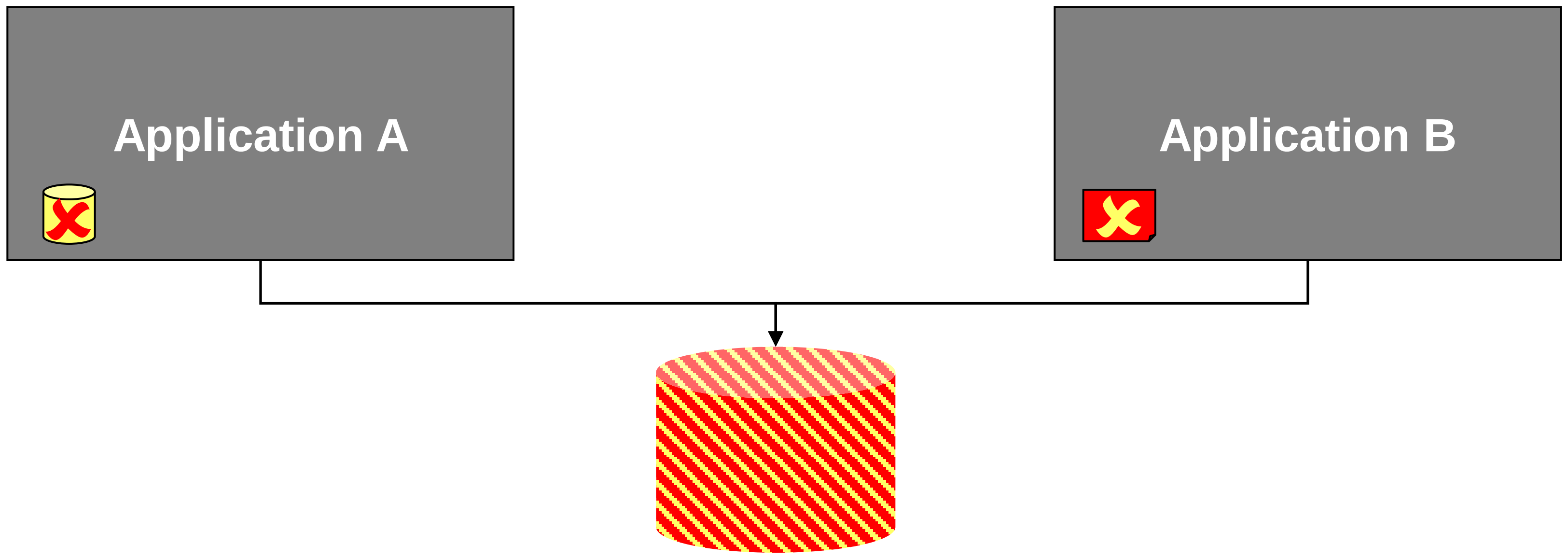
File Integration

- Advantages
  - Almost all applications use/produce files
  - Standard formats often available
  - No knowledge of the internal application
  - No special tools necessary
  - Receiving application can manipulate file data
- Disadvantages
  - Integrator workload
    - Formats
    - File management
    - Freshness/staleness
    - Timing and locking mechanisms
    - Bad data
  - Update latency
  - Computing resources

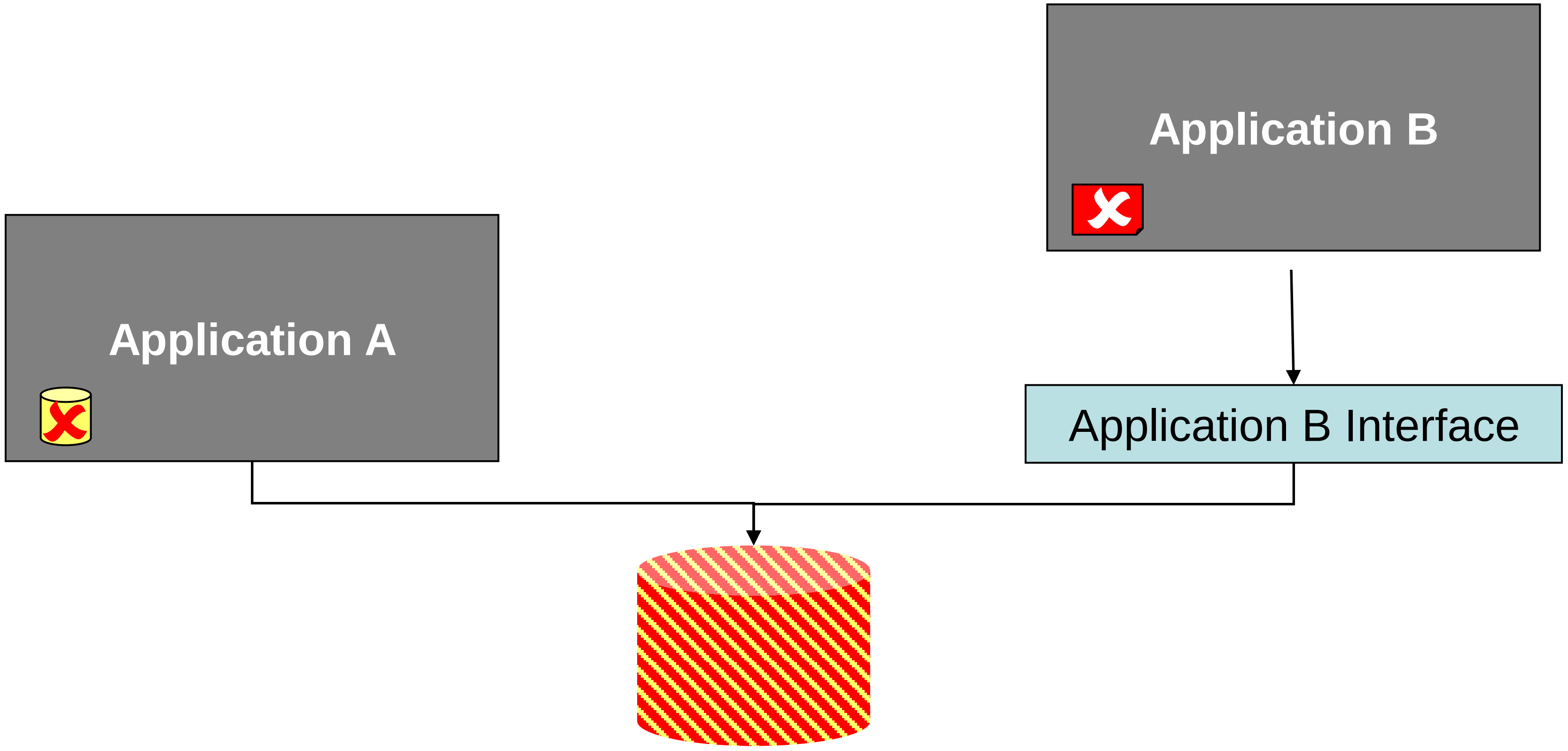
Integration Styles

- File Transfer
- Shared Databases

File Integration



File Integration



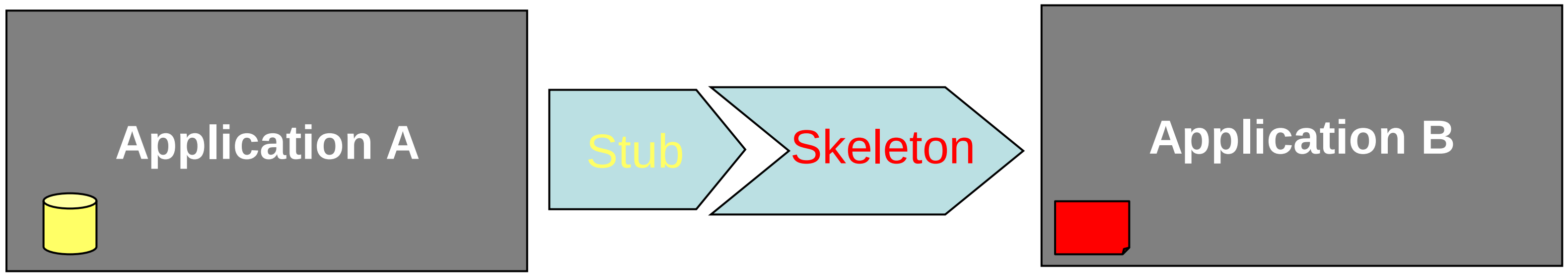
Shared Database Integration

- Advantages
  - Consistency
  - Standard Query Language (SQL)
  - No multiple file formats
  - Single technology
  - Semantic Dissonance resolved
- Disadvantages
  - Semantic Dissonance hard to resolve
  - Finding a workable schema difficult
  - Changed schemas break COTS applications
  - Database may become performance bottleneck
  - Database changes impact performance
  - Not-so-standard SQL
  - Ripple effect of changes

Integration Styles

- File Transfer
- Shared Databases
- Remote Procedure Invocation

How to Integrate Two (or more) Applications - RPC



RPC

- Advantages
  - Data changes can trigger other changes
  - Many existing tools
  - Easier to deal with semantic dissonance
  - Applications are less coupled than with Data Sharing style
- Disadvantages
  - Co-applications have to negotiate exchange interface
  - Interfaces evolve and grow

Integration Styles

- File Transfer
- Shared Databases
- Remote Procedure Invocation
- Messaging Systems

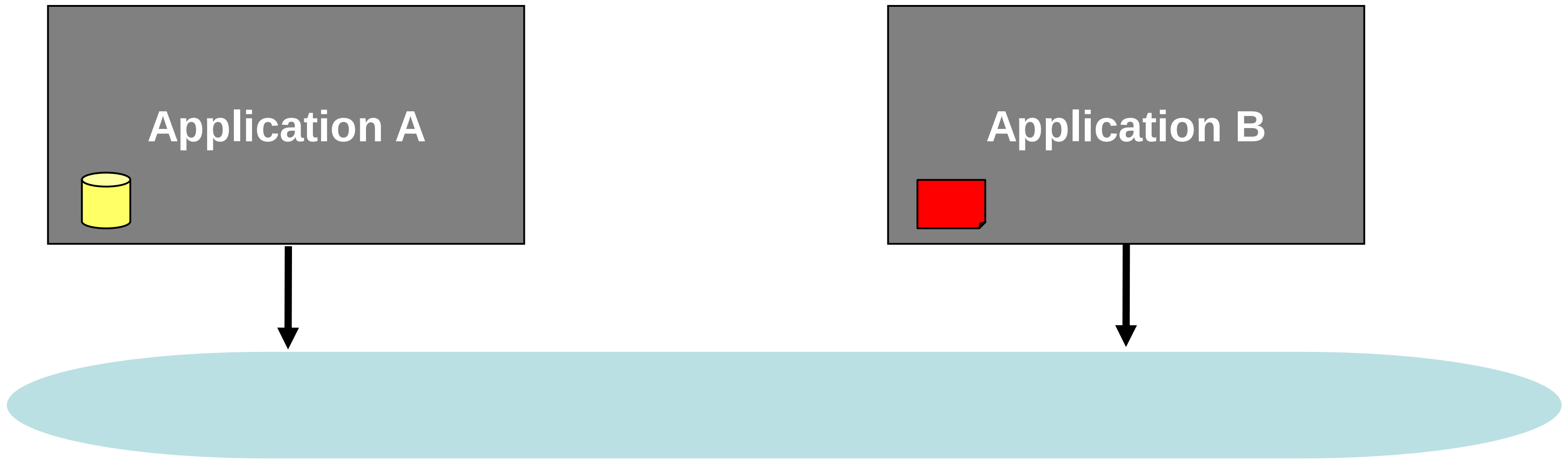
Shared Data Only

Shared Data - Coupling

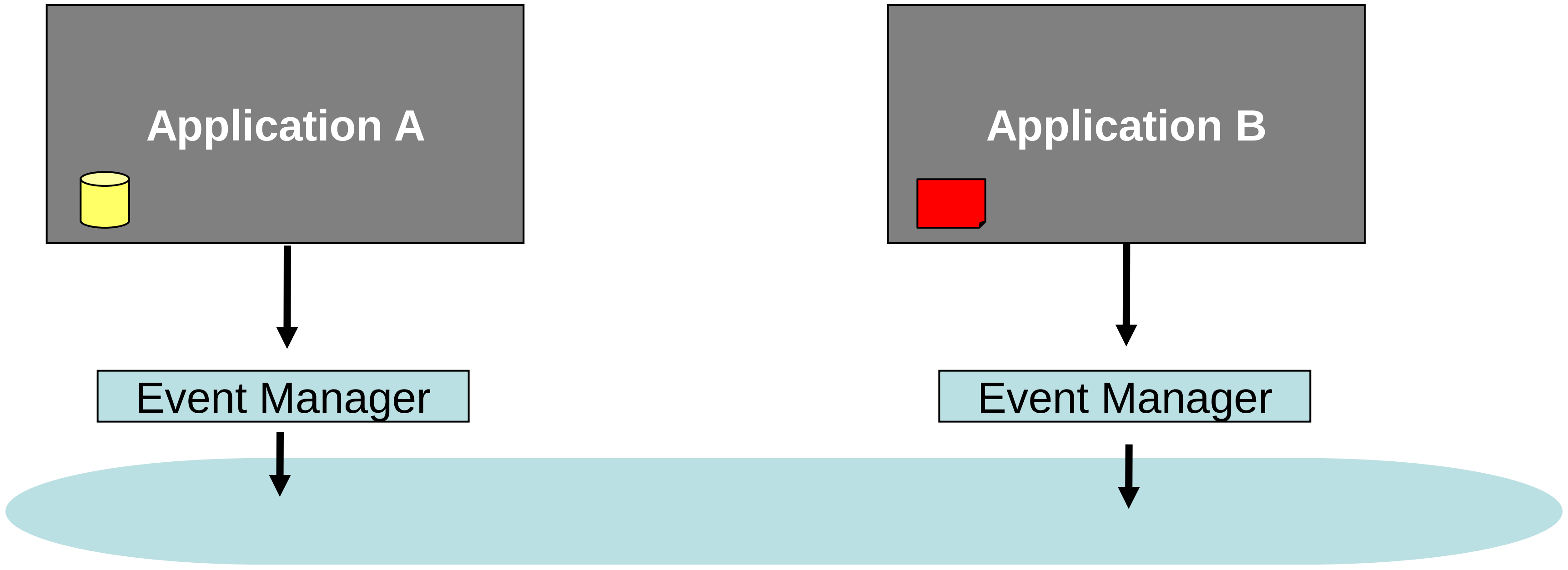
Messaging Services and Concepts

- Channels – Logical addresses
  - Different logical addresses for different purposes
  - How does the application find the relevant addresses?
- Messages
  - Headers – type of data, origin, destination…
  - Body – the data
- Pipes and Filters – preprocesses data
- Routing – broadcast, point to point, context based, publish subscribe…
- Transformation
  - Aggregation
  - Data types
  - Data Representations (XML, ASCII..)
  - Transport (HTTP, SOAP, JMS…)
- Endpoints – Message endpoints interface between the application and the messaging system

How to Integrate Two (or more) Applications - Messaging



How to Integrate Two (or more) Applications - Messaging



Messaging

- Advantages
  - Asynchronous data transfer
  - Decoupled applications
  - Choice among topologies
  - Applications have different conceptual models
  - Timeliness
  - Reliability
- Disadvantages
  - Asynchronous learning curve
  - Testing and debugging more difficult
  - Semantic dissonance still there

How to think about messaging?

- How is data transferred between applications?
- How does an application know where to send the data?
- How or when does the data get transformed?
- What happens when an application isn’t sending (receiving)?
- How do we know if the system is working appropriately?
- Etc..

Summary

- System integration would be easier if applications were designed to be integrated, had similar formats, and had complete and understandable documentation
- Each integration style has advantages and disadvantages
- Messaging systems have become the preferred method of integration, but it isn’t the only method of integration
- Messaging products provide the foundation, but there is still a lot of design to be done