Module Five: Service Oriented Architecture

Assignment deadlines reminder

- April 10th – web service programming tutorial

Service Design Principles

- The principles of service-orientation provide a means of supporting and achieving a foundation paradigm based upon building of SOA characteristics.

Common Service Design Principles

- Standardized Service Contracts
- Loose Coupling
- Abstraction
- Reusability
- Autonomy
- Statelessness
- Discoverability
- Composability

Standardized Service Contracts

- Services share a formal contract
- Services adhere to a communications agreement as defined collectively by one or more service description documents
- For services to interact, they need not share anything but a formal contract that describes each service and defines the terms of information exchange.

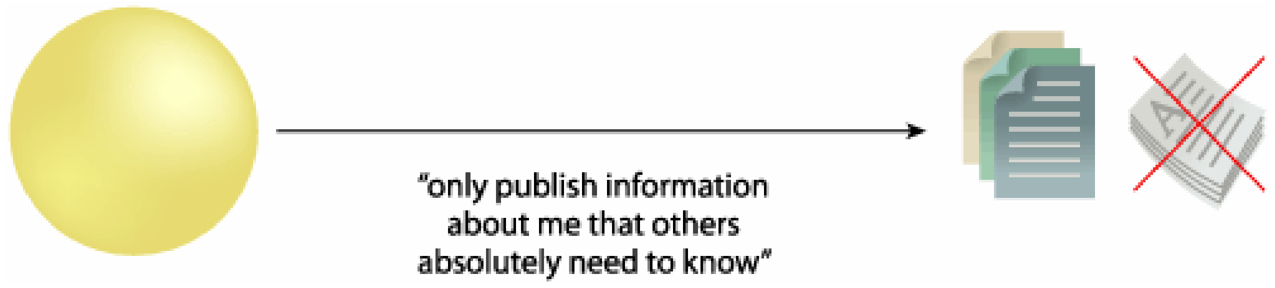
WSDL – standardized service contracts support in Web Services

- WSDL is a specification defining how to describe web services in a common XML grammar.
- WSDL describes four critical pieces of data:
    - Interface information describing all publicly available functions
    - Data type information for all message requests and message responses
    - Binding information about the transport protocol to be used
    - Address information for locating the specified service

Loose Coupling

- Services maintain a relationship that minimizes dependencies and only maintain an awareness of each other
- Services must be designed to interact without the need for tight, cross-service dependencies
- A service is defined solely by an implementation-independent interface
- Services should be able to change their implementation without impacting service consumers

Abstraction


"only publish information about me that others absolutely need to know"

- Services abstract underlying logic
- Beyond what is described in the service contract, services hide logic from the outside world
- The only part of a service that is visible to the outside world is what is exposed via the service contract. Underlying logic, beyond what is expressed in the descriptions that comprise the contract, is invisible and irrelevant to service requestors.

Reusability

- Services are reusable
- Logic is divided into services with the intention of promoting reuse
- Regardless of whether immediate reuse opportunities exist; services are designed to support potential reuse

WSDL Authoring Style Recommendation helps with reusability of Web Services

- Maintain WSDL document in 3 separate parts
  - Data type definitions
  - Abstract definitions
  - Specific service bindings
- Use “*import*” element to import necessary part of WSDL document

Autonomy

- Services are autonomous
- Services have control over the logic they encapsulate
- The logic governed by a service resides within an explicit boundary. The service has control within this boundary and is not dependent on other services for it to execute its governance.

Statelessness

- Services are stateless
- Services should not be required to manage state information, as that can impede their ability to remain loosely coupled. Services should be designed to maximize statelessness even if that means deferring state management elsewhere.
- Service implementations should not hold conversational state across multiple requests.
    - Communicate complete information at each request.
- Each operation should be functionally isolated (separate, independent).

Discoverability

- Services are discoverable
- Services are designed to be outwardly descriptive so that they can be found and assessed via available discovery mechanisms
- Services should allow their descriptions to be discovered and understood by humans and service requestors that may be able to make use of their logic.

Composability

- Services are composable
- Collections of services can be coordinated and assembled to form composite services
- Services may compose other services. This allows logic to be represented at different levels of granularity and promotes reusability and the creation of abstraction layers.

Poll(Anonymous) 1 answer(s) at most

Setting

Which service design principle is in your opinion the most important?

- A Services share a formal contract
- B Services are loosely coupled
- C Services abstract underlying logic
- D Services are reusable
- E Services are autonomous
- F Services are stateless
- G Services are discoverable
- H Services are composable

Submit

Service Design Principles

Do service orientation principles interrelate?

- How principles can relate to and affect each other?
- What is the cause and effect of applying each of the common principles?

- Standardized Service Contracts
- Loose Coupling
- Abstraction
- Reusability
- Autonomy
- Statelessness
- Discoverability
- Composability

Open Question Points: 10

Setting

Can reusability be affected by applying other principles? How?

Answer

Service reusability relationship with other principles

- Service autonomy establishes an execution environment that facilitates reuse because the service achieves increased independence and self-governance.
- The less dependencies a service has, the broader its reuse applicability.

Service reusability relationship with other principles

- Service statelessness supports reuse because it maximizes the availability of a service and typically promotes a generic service design that defers state management and activity-specific processing outside of the service boundary.

Service reusability relationship with other principles

- Service abstraction fosters reuse because it establishes the black box concept.
- Proprietary processing details are hidden and potential consumers are only made aware of an access point represented by a generic public interface.

Service reusability relationship with other principles

- Service discoverability promotes reuse as it allows those that build consumers to search for, discover and assess services offering reusable functionality.

Service reusability relationship with other principles

- Service loose coupling establishes an inherent independence that frees a service from immediate ties to others. This makes it a great deal easier to realize reuse.

Open Question Points: 10

Setting

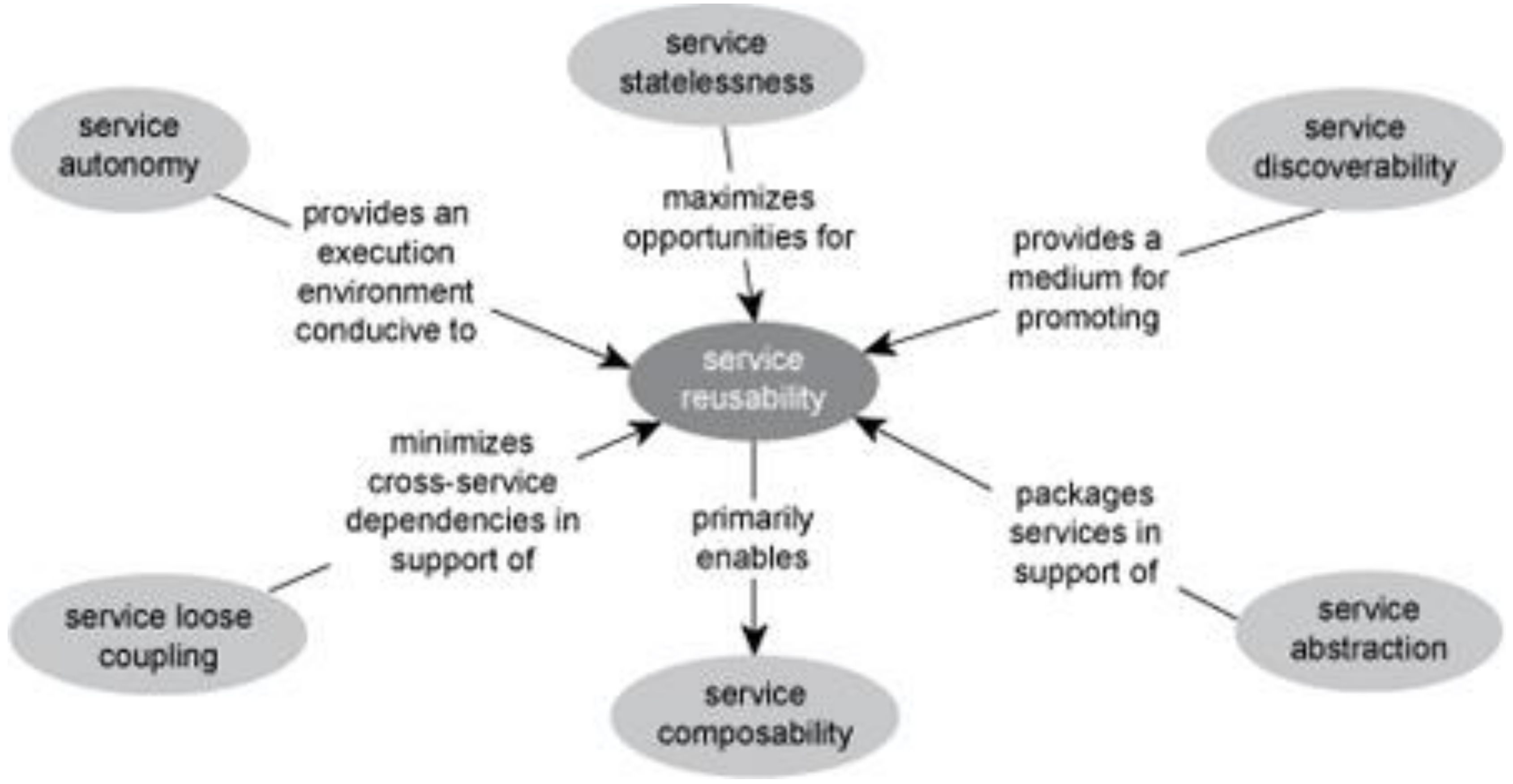
Will applying the principle of reusability have impact on any other principle? Which one?

Answer

Service reusability relationship with other principles

- Service composability is primarily possible because of reuse.
- The ability for new automation requirements to be fulfilled through the composition of existing services is feasible when those services being composed are built for reuse.

Service reusability relationship with other principles


service statelessness
maximizes opportunities for
service autonomy
provides an execution environment conducive to
service discoverability
provides a medium for promoting
service reusability
minimizes cross-service dependencies in support of
primarily enables
packages services in support of
service loose coupling
service composability
service abstraction

Common Service Design Principles

- Standardized Service Contracts
- Loose Coupling
- Abstraction
- Reusability
- Autonomy
- Statelessness
- Discoverability
- Composability

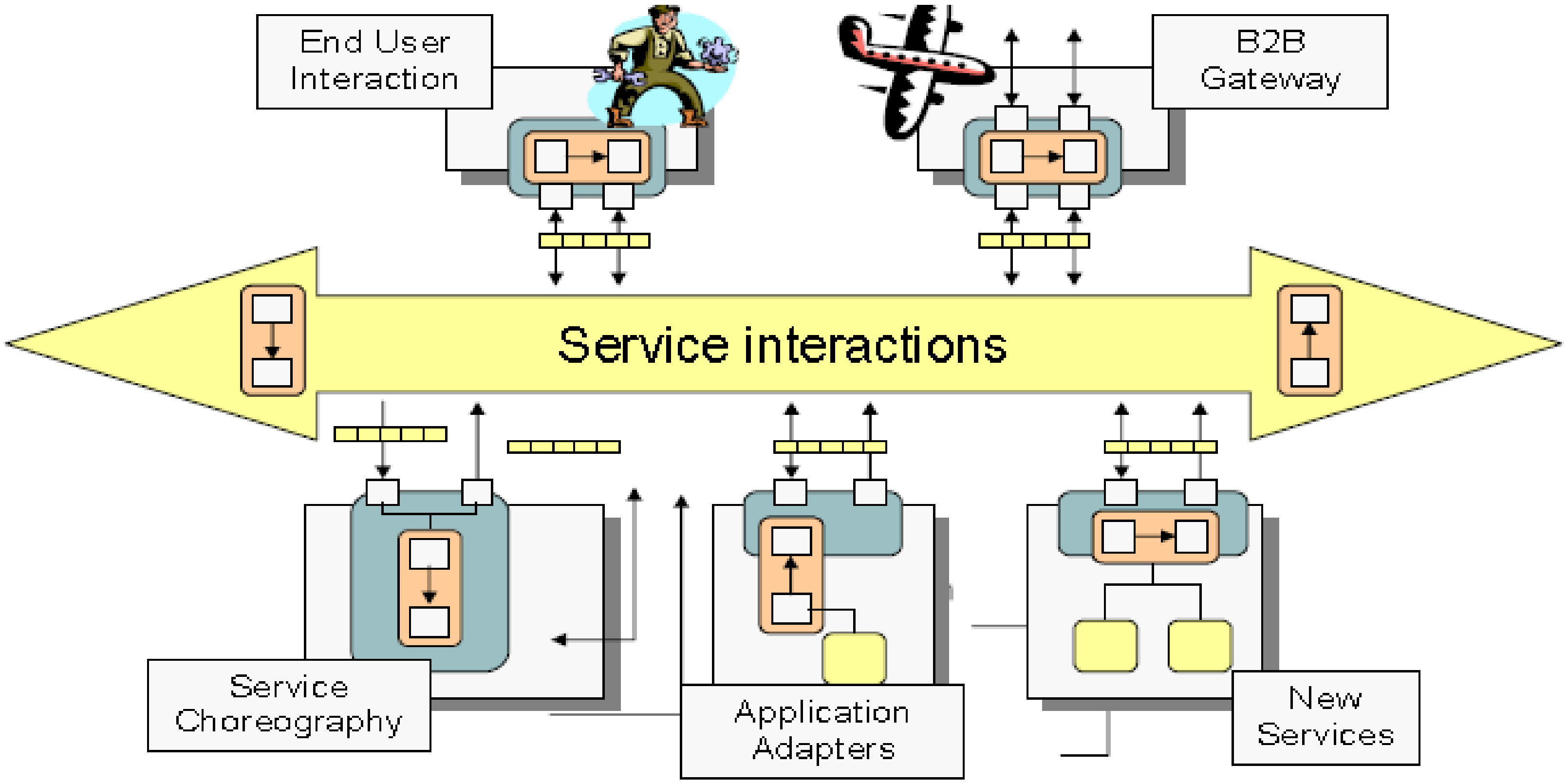
How do you apply these principles?

You can work towards this goal in a number of ways:

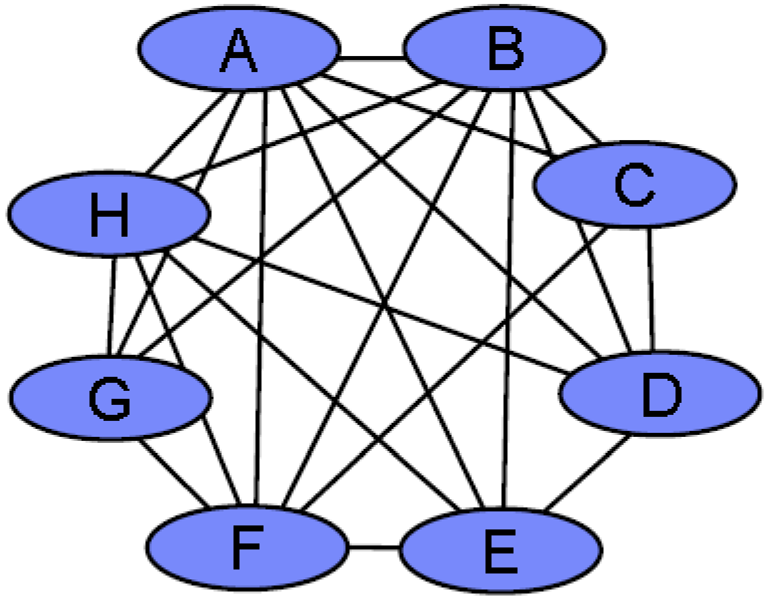
- Declarative techniques
  - Provide separation of concerns between the application logic and the configuration of middleware
- Code generation
  - WSDL code generation can hide the complexities of SOAP, HTTP and JMS from the developer
- Tooling
  - Can help reduce complexity for the developer
- Model-driven development
  - Exploiting both tooling and code generation capabilities to simplify development

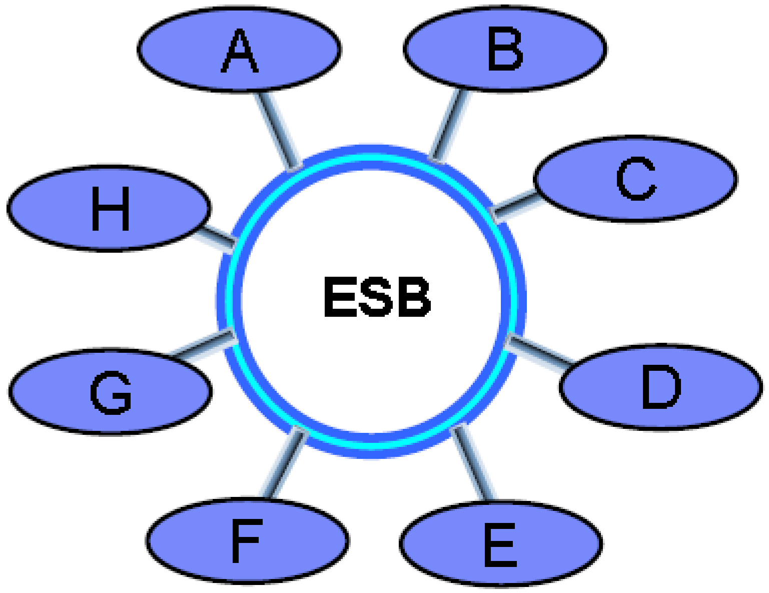
Loose coupling is enabled by an ESB

- An ESB (enterprise service bus) is an intermediary-oriented approach



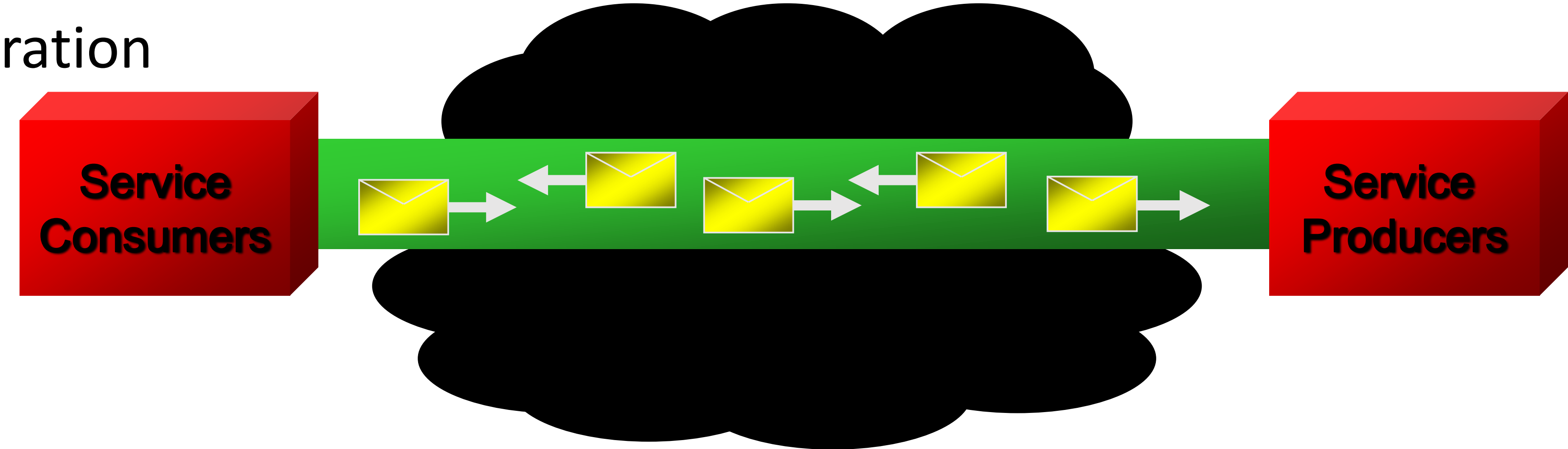
Point-to-point service interactions





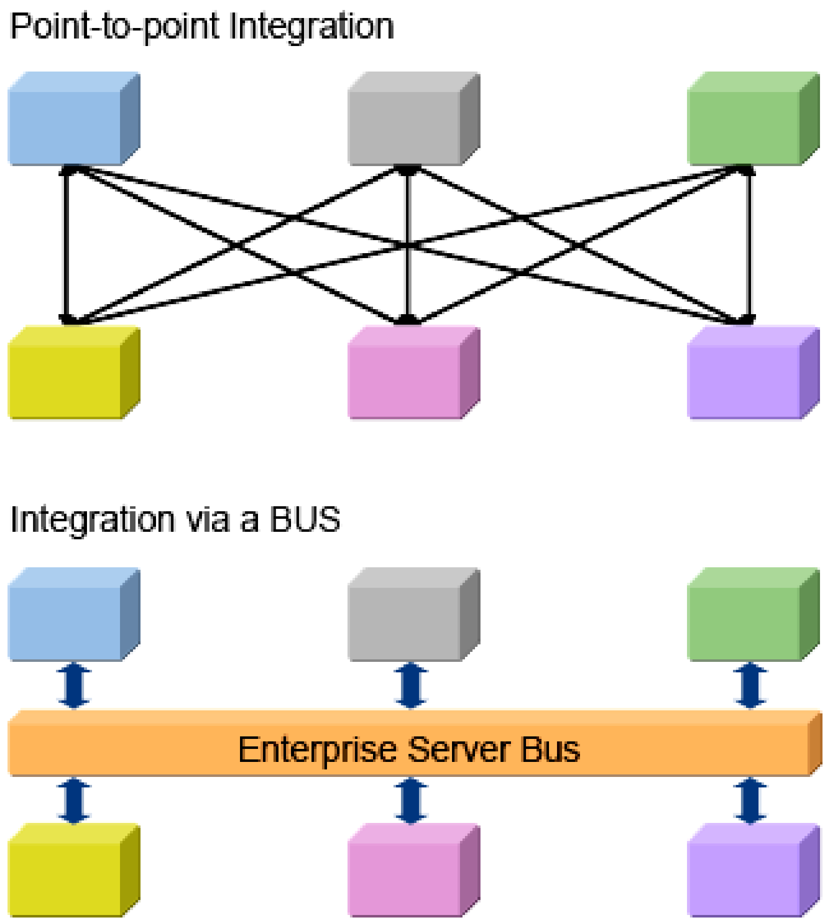
Service Communication

- Communicate with messages
- No knowledge about partner
- Likely heterogeneous
- Providing reliability and security to messages
- Sending messages across consumers and producers
- Service Orchestration

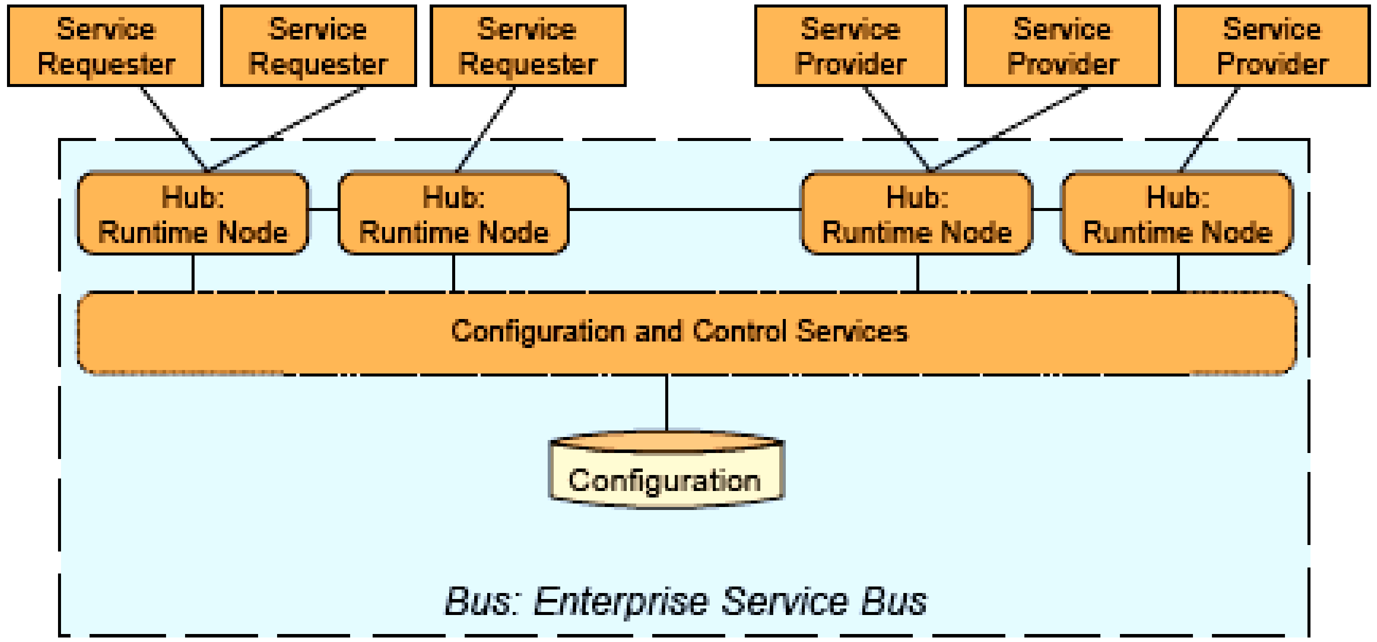


Enterprise Service Bus (ESB)

Overcoming the Problems of Point-to-point Integration with ESB

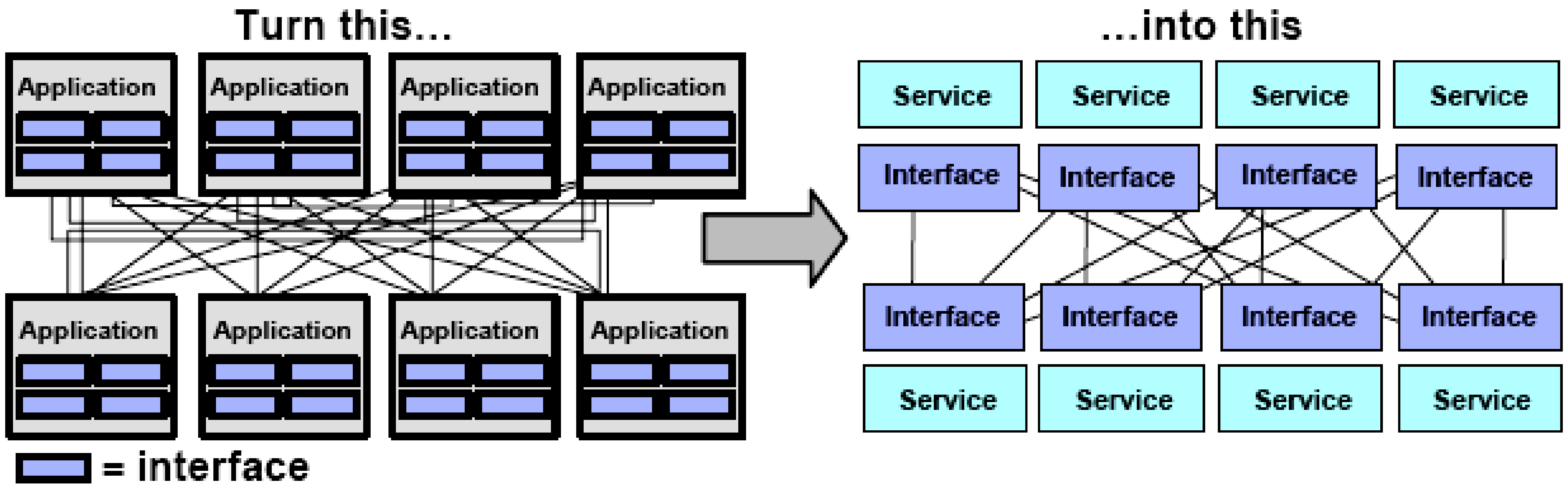


ESB as a Distributed Infrastructure with Centralized Control



SOA without ESB

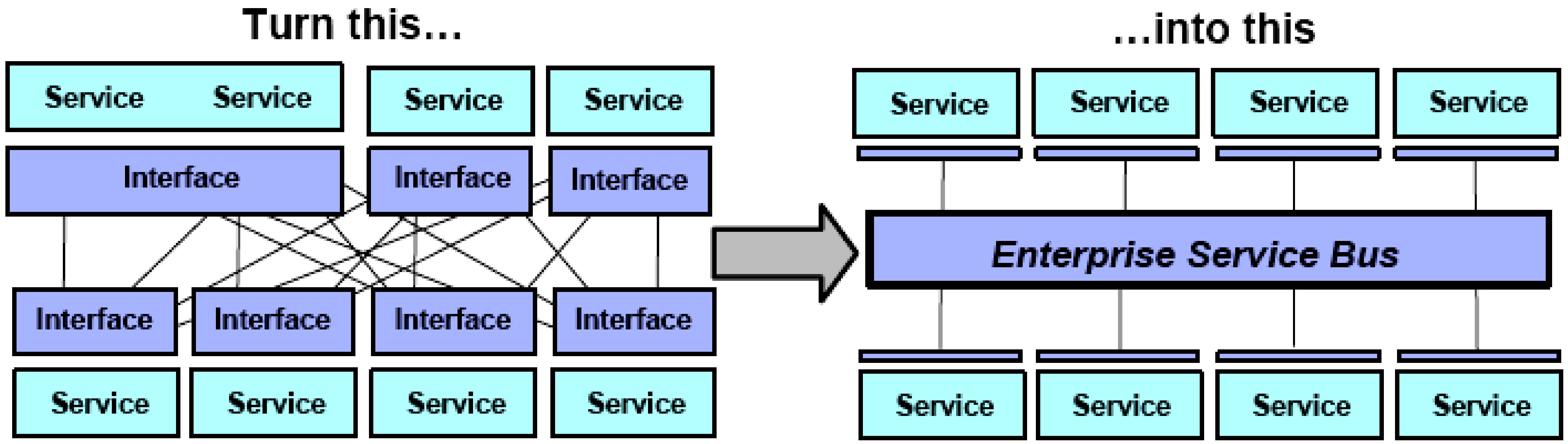
- *Decouples interfaces from applications*
- Separate connection points still leave bloated interfaces



- ✓ Rich business abstractions describe the application interface
- ✓ Decouples the interfaces from the business applications
- ✓ The number and complexity of the interfaces is reduced
- ✓ Business applications and their interfaces become reusable

SOA with ESB

- Shrinks the interfaces further



- ✓ Decouples the point-to-point connections from the interfaces
- ✓ Allows for dynamic selection, substitution, and matching
- ✓ Enables more flexible coupling and decoupling of the applications
- ✓ Enables you to find both the applications and the interfaces for re-use

The ESB Architecture

- The purpose of the ESB is so that common specifications, policies, etc can be made at the bus level, rather than for each individual service.

Where not to use SOA?

- We discussed many benefits of SOA. Can you think of any scenario when SOA would NOT be a preferred choice?

Open Question Points: 10

Setting

We discussed many benefits of SOA. Can you think of any scenario when SOA would NOT be a preferred choice?

Open Question is only supported on Version 2.0 or newer.

Answer

Applying SOA - Challenges

- Service Orientation

Business functionality has to be made available as services. Service contracts must be fixed

- Reuse

Implemented services must be designed with reuse in mind. This creates some overhead.

- Sharing of Responsibilities

Potential service users must be involved in the design process and will have influence on the service design

- Increased complexity!

Where not to use SOA?

- When you have homogenous IT environment

Where not to use SOA?

- When you have homogenous IT environment
- When real time performance is critical

Where not to use SOA?

- When you have homogenous IT environment
- When real time performance is critical
- When tight coupling is a pro not a con

Where not to use SOA?

- When you have homogenous IT environment
- When real time performance is critical
- When tight coupling is a pro not a con
- When things don’t change

Module 5 Summary

- Architectural patterns of SOA
- Designing distributed systems with SOA

Establishing a SOA in an Enterprise– a case study (part 1)

(Module 5 Learning Activities)

Prepare for the discussion

- In this example, we will study an enterprise that was faced with some business and technical problems
- When listening to this case, think about the following question:
  - How can SOA address these issues?

Case Study: RailCo Ltd.

- Established in the early 90s

Case Study: RailCo Ltd.

- Established in the early 90s
- The company has gradually grown from a staff of 12 to 40

Case Study: RailCo Ltd.

- Established in the early 90s
- The company has gradually grown from a staff of 12 to 40
- Started out as a brokerage for various railway wholesalers, but then came to specialize in air brakes.

Case Study: RailCo Ltd.

- Established in the early 90s
- The company has gradually grown from a staff of 12 to 40
- Started out as a brokerage for various railway wholesalers, but then came to specialize in air brakes.
  - The narrowed business focus resulted in increased opportunities, as RailCo was able to become a wholesaler in its own right by dealing directly with air brake parts manufacturers.

Technical infrastructure

- 5 employees and 1 manager are dedicated to full-time IT duties
  - responsible primarily for maintaining client workstations and back-end servers.

Technical infrastructure

- 5 employees and 1 manager are dedicated to full-time IT duties
  - responsible primarily for maintaining client workstations and back-end servers.
- Custom development tasks have typically been outsourced to local consulting firms

Technical infrastructure

- 5 employees and 1 manager are dedicated to full-time IT duties
  - responsible primarily for maintaining client workstations and back-end servers.
- Custom development tasks have typically been outsourced to local consulting firms
- Periodic upgrades and maintenance fixes have been the responsibility of in-house staff

RailCo's automated environment (1)

- A two-tier client-server system governing all accounting and inventory control transactions

RailCo's automated environment (1)

- A two-tier client-server system governing all accounting and inventory control transactions
  - Two administrative clerks manually feed this solution with standard transaction document data (primarily incoming and outgoing purchase orders and invoices)

RailCo's automated environment (1)

- A two-tier client-server system governing all accounting and inventory control transactions
  - Two administrative clerks manually feed this solution with standard transaction document data (primarily incoming and outgoing purchase orders and invoices)
  - Receipt and submission of these documents typically initiates corresponding inventory receiving and order shipping processes

RailCo's automated environment (2)

- A contact management system in which customer and business partner profile information is stored and maintained

RailCo's automated environment (2)

- A contact management system in which customer and business partner profile information is stored and maintained
  - This simple application consists of a database fronted by Web-based data entry and reporting user-interfaces

RailCo's automated environment (2)

- A contact management system in which customer and business partner profile information is stored and maintained
  - This simple application consists of a database fronted by Web-based data entry and reporting user-interfaces
  - Users range from managers to administrative assistants and accounting personnel

Business Problems

- Profit margins have been noticeably declining over the past year.
  - Clients have been switching to a competitor providing the same products in a more efficient manner and at a lower cost

Business Problems

- Profit margins have been noticeably declining over the past year.
  - Clients have been switching to a competitor providing the same products in a more efficient manner and at a lower cost
- Further investigation led to the discovery that this competitor has implemented an extension to their existing accounting system, allowing them to perform various transactions online via B2B solutions provided by some of the larger clients

Business Problems

- Profit margins have been noticeably declining over the past year.
    - Clients have been switching to a competitor providing the same products in a more efficient manner and at a lower cost
- Further investigation led to the discovery that this competitor has implemented an extension to their existing accounting system, allowing them to perform various transactions online via B2B solutions provided by some of the larger clients
- A further unpleasant revelation was that RailCo's primary client, Transit Line Systems (TLS), has started an online relationship with this competitor as well

Business Problems

- Profit margins have been noticeably declining over the past year.
    - Clients have been switching to a competitor providing the same products in a more efficient manner and at a lower cost
- Further investigation led to the discovery that this competitor has implemented an extension to their existing accounting system, allowing them to perform various transactions online via B2B solutions provided by some of the larger clients
- A further unpleasant revelation was that RailCo's primary client, Transit Line Systems (TLS), has started an online relationship with this competitor as well
- RailCo is a company with outdated technology automating inefficient business processes
    - Need to better respond to new business trends and automation requirements

RailCo-to-TLS initial Web Services

- To remain competitive and minimize losses, RailCo must upgrade its automation environment as soon as possible.

RailCo-to-TLS initial Web Services

- To remain competitive and minimize losses, RailCo must upgrade its automation environment as soon as possible.
- Its top priority is to participate in online transactions with TLS.

RailCo-to-TLS initial Web Services

- To remain competitive and minimize losses, RailCo must upgrade its automation environment as soon as possible.
- Its top priority is to participate in online transactions with TLS.
- Before our storyline begins, RailCo has already hurried to build a pair of Web services
  - RailCo-to-TLS Invoice Submission Process
  - TLS-to-RailCo Purchase Order Submission Process

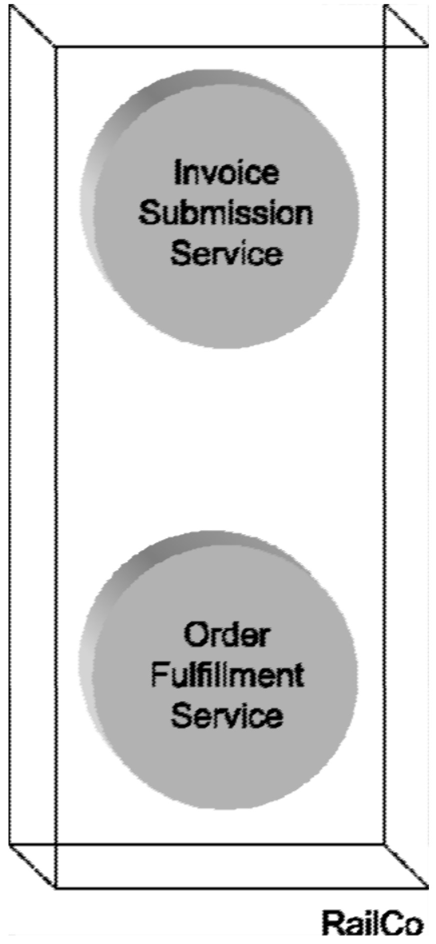


Figure 2.1: RailCo's initial set of Web services, designed only to allow RailCo to connect to TLS's B2B solution.

RailCo-to-TLS initial Web Services

- By that point RailCo runs into some limitations and decides to re-evaluate its environment in consideration of establishing an SOA. Further, RailCo realizes that it must also seek new clients to make up for the lost sales to TLS. This new requirement ends up also affecting the design of its SOA.

Prior solution (1)

- RailCo's accounting solution exists as a two- tier client-server application, where the bulk of application logic resides within an executable deployed on client workstations.

Prior solution (1)

- RailCo's accounting solution exists as a two- tier client-server application, where the bulk of application logic resides within an executable deployed on client workstations.
- Two primary tasks:
  - Enter Customer Purchase Order
  - Create Customer Order

Prior solution (2)

- The completion of each task involves a series of steps that constitute a business process.

Prior solution (2)

- The completion of each task involves a series of steps that constitute a business process.
- This process was originally modelled using standard workflow logic and then implemented as part of a packaged solution.

Prior solution (2)

- The completion of each task involves a series of steps that constitute a business process.
- This process was originally modelled using standard workflow logic and then implemented as part of a packaged solution.
- Within the application, the process may or may not be separately represented by individual sets of programming routines.

Prior solution (2)

- The completion of each task involves a series of steps that constitute a business process.
- This process was originally modelled using standard workflow logic and then implemented as part of a packaged solution.
- Within the application, the process may or may not be separately represented by individual sets of programming routines.
- Regardless, it is compiled into a single executable that provides a fixed manner in which the process is automated.

RailCo's accounting system

- RailCo's accounting system is a classic two-tier client- server application.
- Its GUI front-end consists of a single executable designed for deployment on old Windows workstations.
  - It provides user-interfaces for looking up, editing, and adding various accounting records
  - It also offers a financial reporting facility that can produce a fixed amount of statements with detailed or summarized accounting data

RailCo's accounting system

- RailCo's accounting system is a classic two-tier client- server application.
- Its GUI front-end consists of a single executable designed for deployment on old Windows workstations.
    - It provides user-interfaces for looking up, editing, and adding various accounting records
    - It also offers a financial reporting facility that can produce a fixed amount of statements with detailed or summarized accounting data
- Considering it's only ever had two to three users, there have never really been performance problems on the database end. The now outdated RDBMS (Relational Database Management System) that has been in place for the past decade has been reliable and has required little attention.

Problems with this application have surfaced:

- Operating system upgrades have introduced erratic behavior on some screens, resulting in unexplainable error messages.
  - It is uncertain if these are caused by other programs that have been installed on the workstations.

Problems with this application have surfaced:

- Operating system upgrades have introduced erratic behavior on some screens, resulting in unexplainable error messages.
  - It is uncertain if these are caused by other programs that have been installed on the workstations.
- The workstations themselves have been rarely upgraded and have not kept pace with the hardware demands of recent software upgrades.
  - After the accounting system launches, there is little more the user can do with the computer. As a result, employee productivity has been affected somewhat.

Problems with this application have surfaced:

- Operating system upgrades have introduced erratic behavior on some screens, resulting in unexplainable error messages.
  - It is uncertain if these are caused by other programs that have been installed on the workstations.
- The workstations themselves have been rarely upgraded and have not kept pace with the hardware demands of recent software upgrades.
  - After the accounting system launches, there is little more the user can do with the computer. As a result, employee productivity has been affected somewhat.
- Following a new records management policy and some billing procedure changes, a modification to the overall billing process was imposed on the accounting personnel.
  - Because the accounting system was not designed to accommodate this change, employees are required to supplement the automated billing process by manually filling out supplementary forms.

Discussion

- Fundamentally, this accounting system has been getting the job done.
- However, the actual accounting tasks performed by the users have become increasingly convoluted and inefficient. This is due to the questionable stability of the workstation environments and also because the system itself is not easily adaptable to changes in the processes it automates.
- How can SOA address these issues?

- We will continue this discussion on Monday