Understanding Service-Orientation

hungdn@ptit.edu.vn

3.1 Introduction to Service-Orientation

Service

- **Individuals**

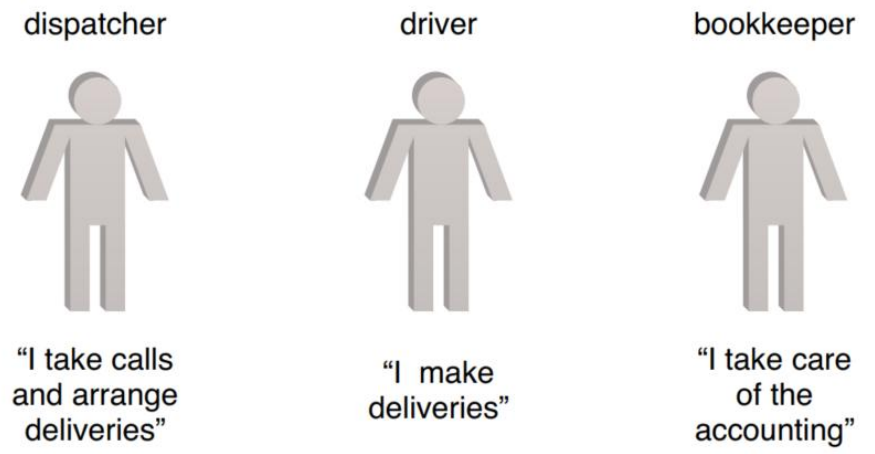

dispatcher
driver
bookkeeper
"I take calls and arrange deliveries"
"I make deliveries"
"I take care of the accounting"

Figure 3.1
Three individuals, each capable of providing a distinct service.

- **Organization**

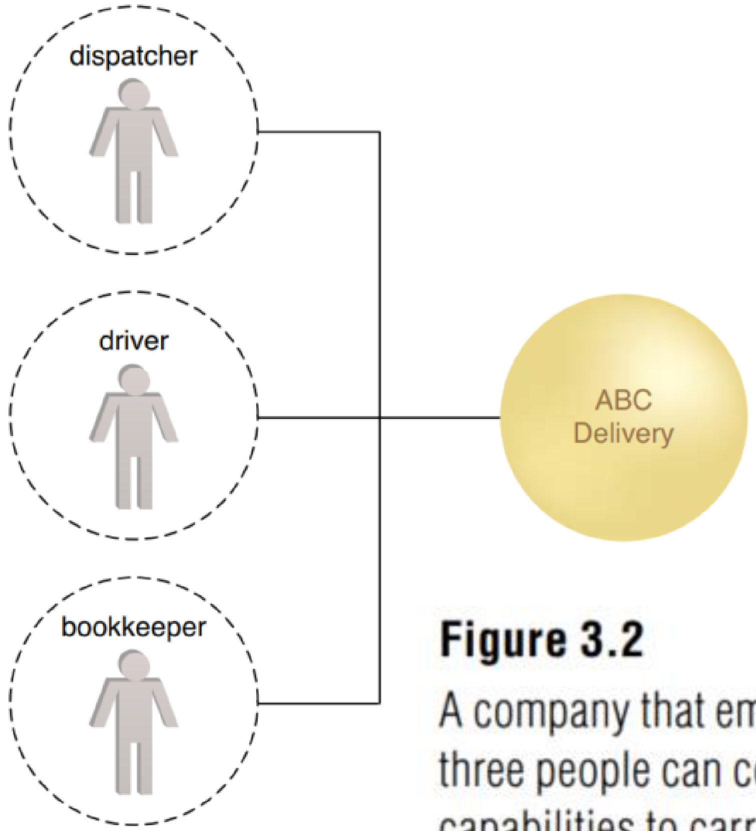


Figure 3.2
A company that employs these three people can compose their capabilities to carry out its business.

service-orientation

- Each individual contributes a distinct service

- Establishing these types of baseline requirements within and across **business automation solutions** is a key goal of *service-orientation*
  - Need to have fundamental, common characteristics, such as availability, reliability, and the ability to communicate using the same language

Services in Business Automation

- A *service* is a software program that makes its functionality available via a published API (that is part of a *service contract*)
- Different implementation technologies can be used to program and build services

1. SOAP-based Web Service
2. RESTful services (or just REST service)

Figure 3.3
The symbol used to represent an abstract service.

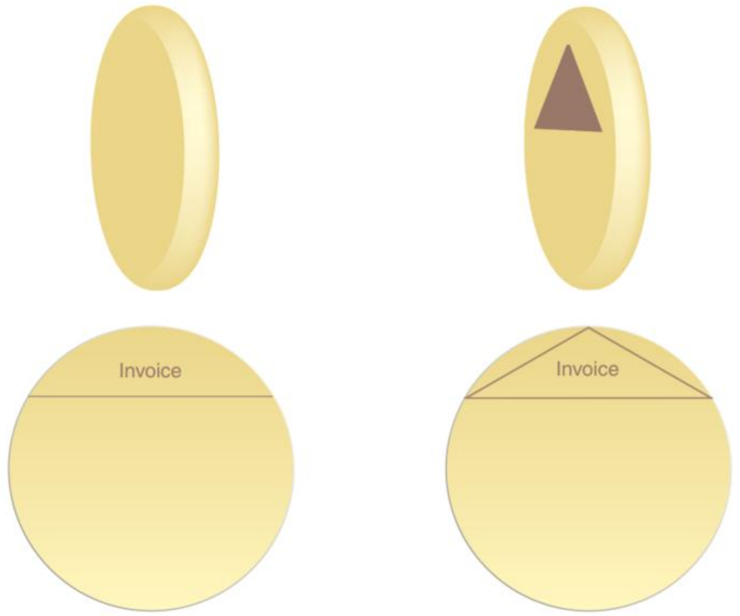
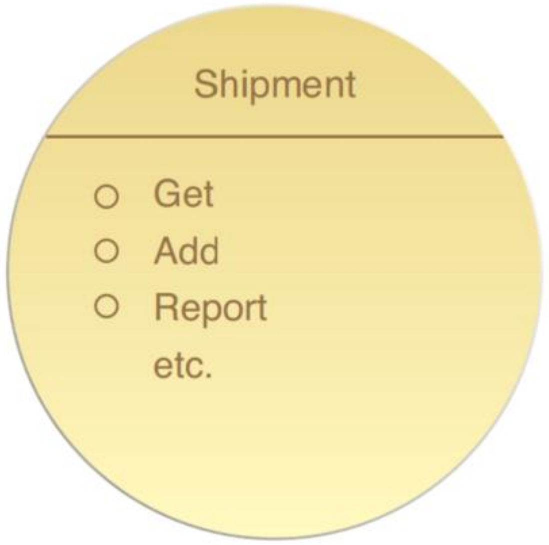


Figure 3.4
The chorded circle symbol used to display an Invoice service contract (left), and a variation of this symbol used specifically for REST service contracts (right).

Services Are Collections of Capabilities

- **Much like a human, an automated service can provide multiple capabilities**
  - They are grouped together because they relate to a **functional context** established by the service



- **A *service consumer* is a software program that accesses and invokes a service or sends a message to a service capability expressed in the service contract.**

"I can:
- drive
- fill out a waybill
- collect payment
etc."

Agnostic vs. Non-Agnostic Logic

- *Agnostic logic*
  - logic that is sufficiently generic so that it is not specific to (has no knowledge of) a particular parent task is classified
  - is considered multipurpose

- *Non-Agnostic Logic*
  - logic that is specific to (contains knowledge of) a single-purpose task

The term "agnostic" originated from Greek and means "without knowledge"

Service-Orientation as a Design Paradigm

- A design paradigm is an approach to designing solution logic.
  - When building distributed solution logic, design approaches revolve around a software engineering theory known as the "*separation of concerns*"

- Applying service-orientation to a meaningful extent results in solution logic that can be safely classified as "*service-oriented*" and units that qualify as "*services*."

Service-Orientation as a Design Paradigm (2)

In service-Orientation

- ***Services*****, as part of service-oriented solutions, exist as physically independent software programs with distinct design characteristics.**
  - Each service is assigned its own distinct functional context and is comprised of a set of capabilities related to this context.
- **Service-oriented solution logic is implemented as a composition of services and *service compositions* designed in accordance with service-orientation.**

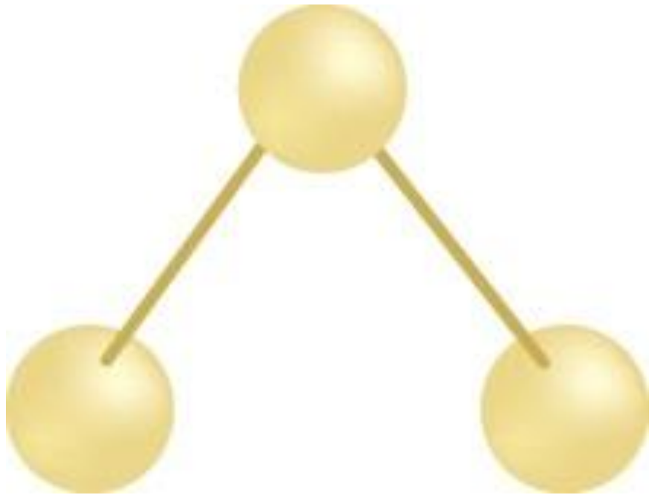


Figure 3.6 This symbol, comprised of three connected spheres, represents a service composition

Service-Orientation as a Design Paradigm (3)

- A *service inventory* is an independently standardized and governed collection of complementary services within a boundary that represents an enterprise or a meaningful segment of an enterprise.

- When an organization has multiple service inventories, this term is further qualified as *domain service inventory*.

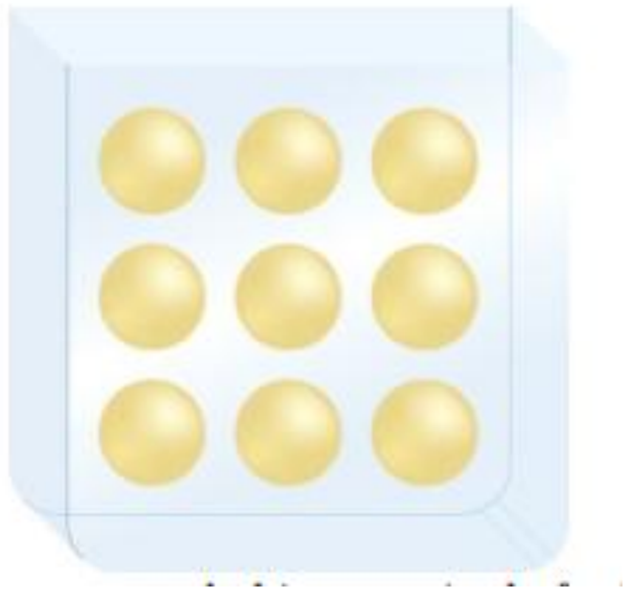


Figure 3.7 The service inventory symbol is comprised of spheres within a container.

Service-Orientation as a Design Paradigm (4)

- **Multiple business processes can be automated by the creation of *service compositions* that draw from a pool of existing agnostic services that reside within a *service inventory*.**

- **A *service composition* is comprised of *services* that have been assembled to provide the functionality required to automate a specific business task or process**

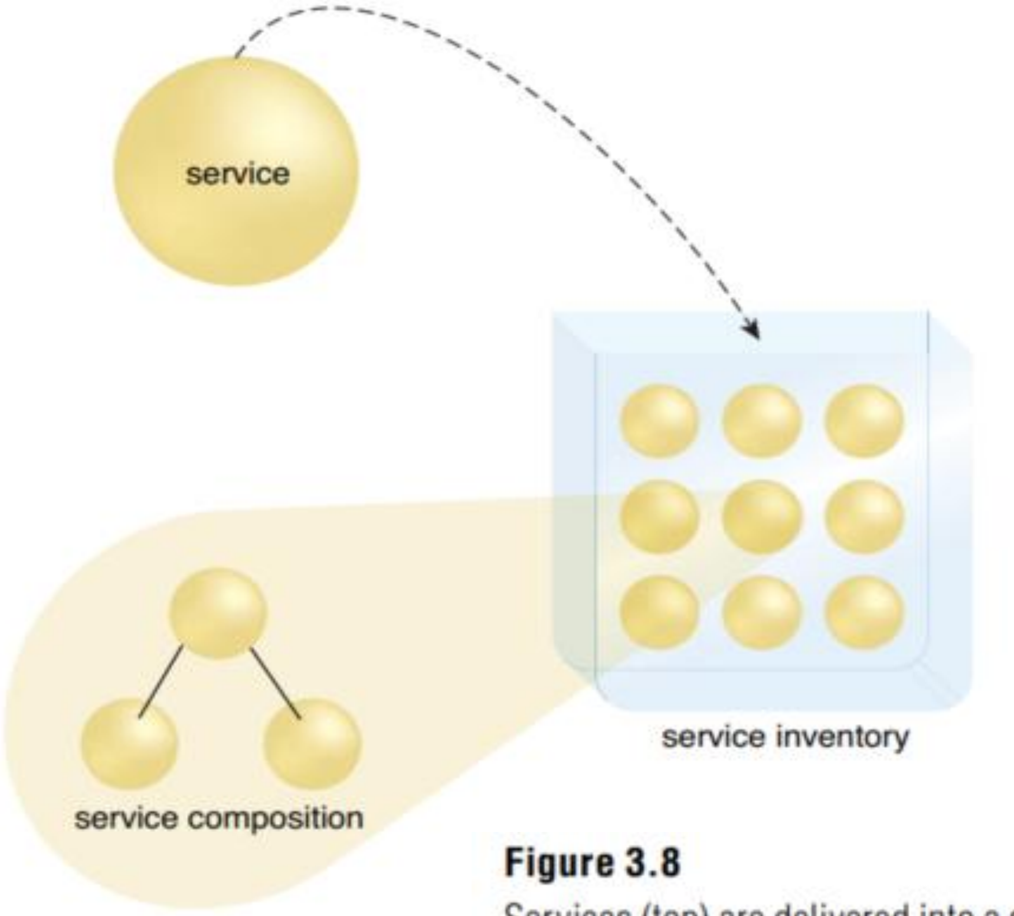
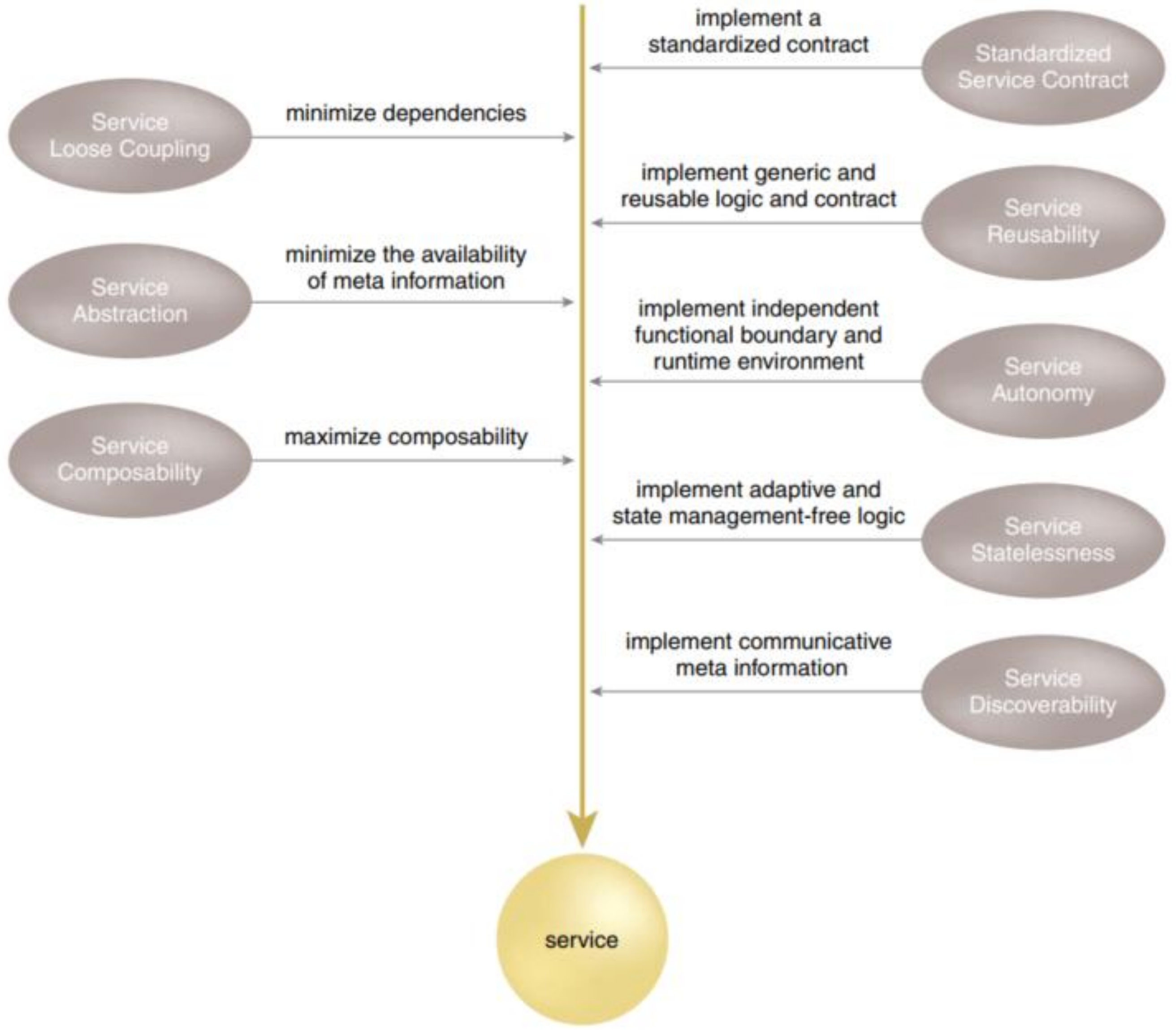


Figure 3.8
Services (top) are delivered into a service inventory (right) from which service compositions (bottom) are drawn.

Service-Orientation Design Principles

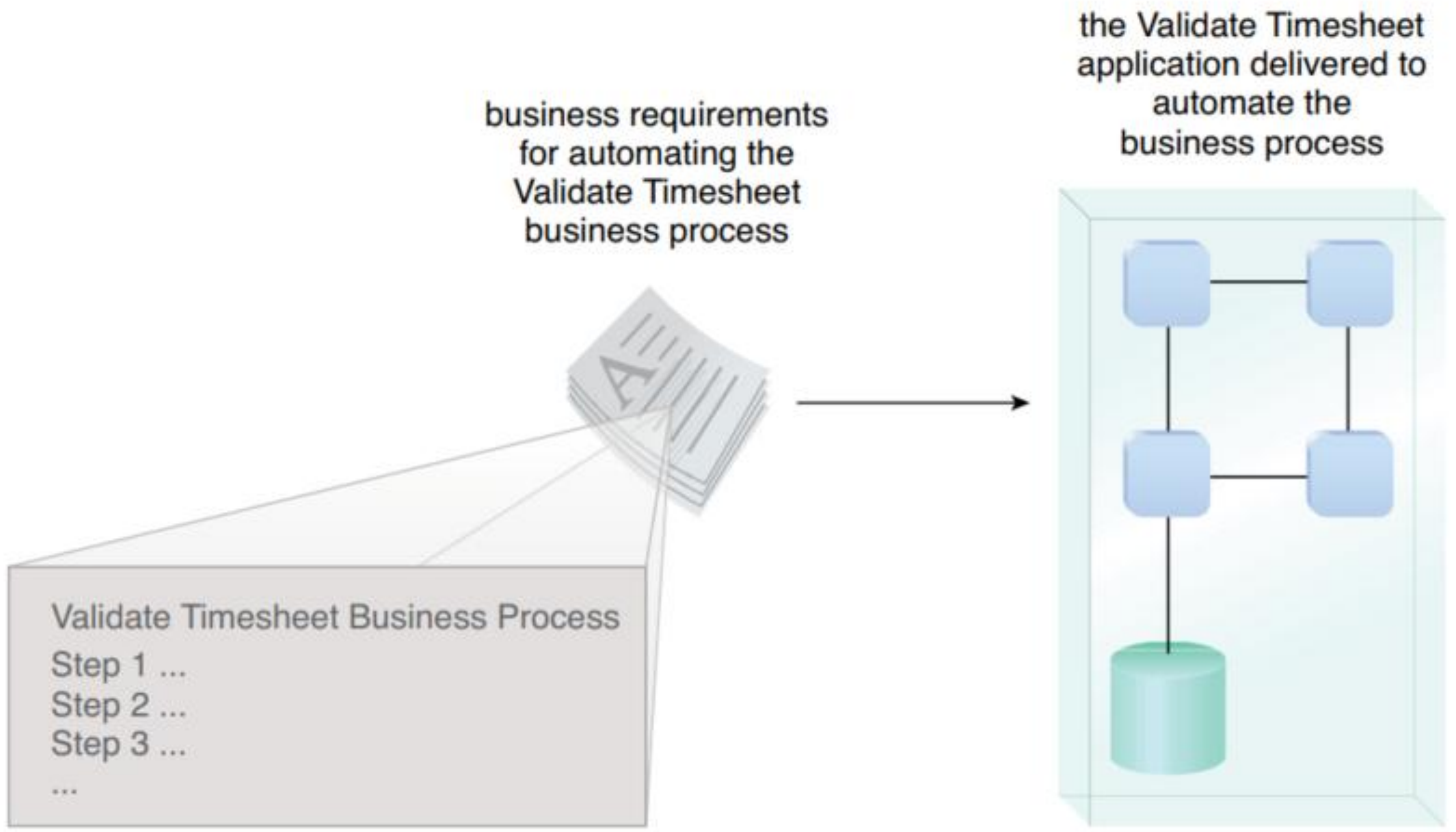
- How exactly is the service-orientation paradigm applied?
  - It is primarily applied at the service level (Figure 3.9) via the application of the following eight design principles



Standardized Service Contract
implement a standardized contract
Service Loose Coupling
minimize dependencies
Service Reusability
implement generic and reusable logic and contract
Service Abstraction
minimize the availability of meta information
Service Autonomy
implement independent functional boundary and runtime environment
Service Composability
maximize composability
Service Statelessness
implement adaptive and state management-free logic
Service Discoverability
implement communicative meta information
service

3.2 Problems Solved by Service-Orientation

Silo-based Application Architecture



business requirements for automating the Validate Timesheet business process
the Validate Timesheet application delivered to automate the business process
Validate Timesheet Business Process
Step 1 ...
Step 2 ...
Step 3 ...
...

Development cost = x

Yearly operational cost = y

Estimated yearly savings due to increased productivity = (x/2) - y

Figure 3.10

A ratio of one application for each new set of automation requirements has been common.

Silo-based Application Architecture (2)

The negative:

- The ability to gain any further value from these applications is usually inhibited
- When new requirements and processes come
  - Need make significant changes
  - or build a new application altogether

The positive:

- Solutions can be built efficiently
- The business analysis effort involved with defining the process to be automated is straightforward
- Solution designs are strategically focused.
- The project delivery lifecycle for each solution is streamlined and relatively predictable
- Building new systems from the ground up allows organizations to take advantage of the latest technology advancements

It Can Be Highly Wasteful

- **The creation of new solution logic in a given enterprise commonly results in a significant amount of redundant functionality**

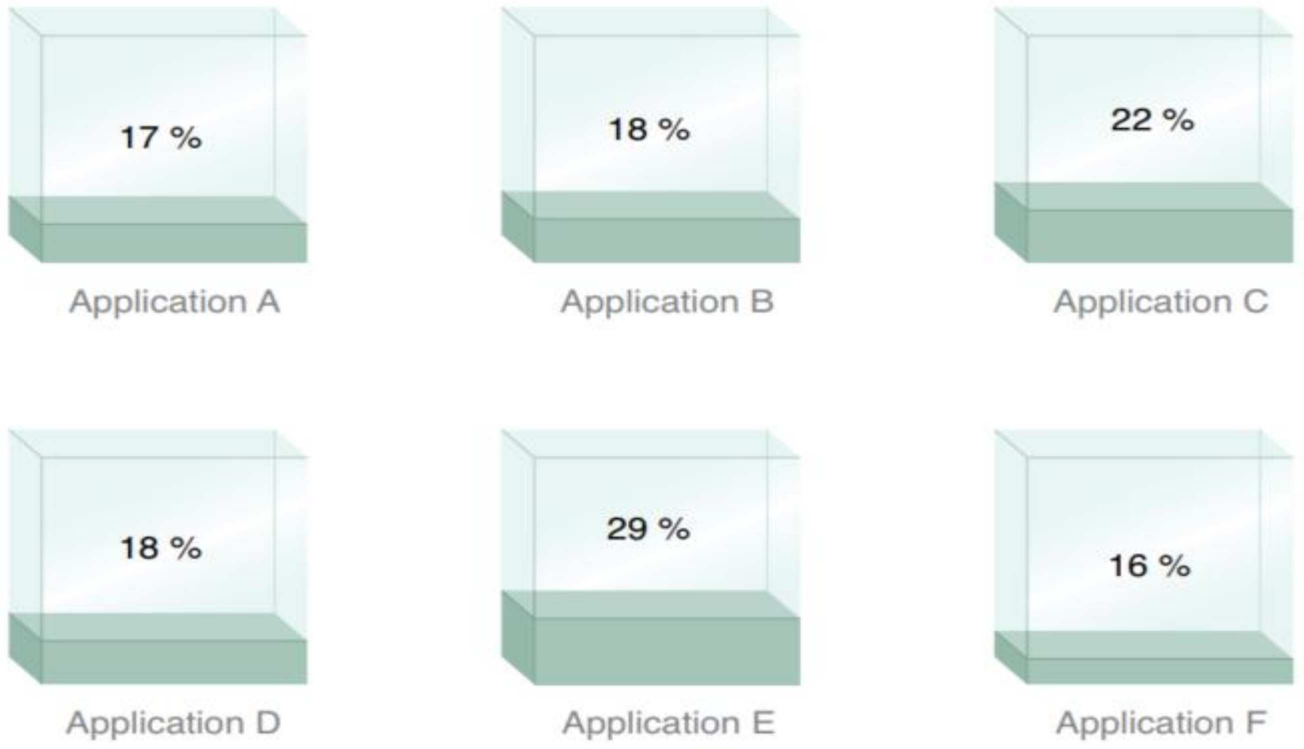


Figure 3.12 Different applications developed independently can result in significant mounts of redundant functionality. The applications displayed were delivered with various levels of solution logic that, in some form, already existed.

It’s Not as Efficient as It Appears

- However, by continually building and rebuilding logic that already exists elsewhere, the process is not as efficient as it could be if the creation of redundant logic could be avoided

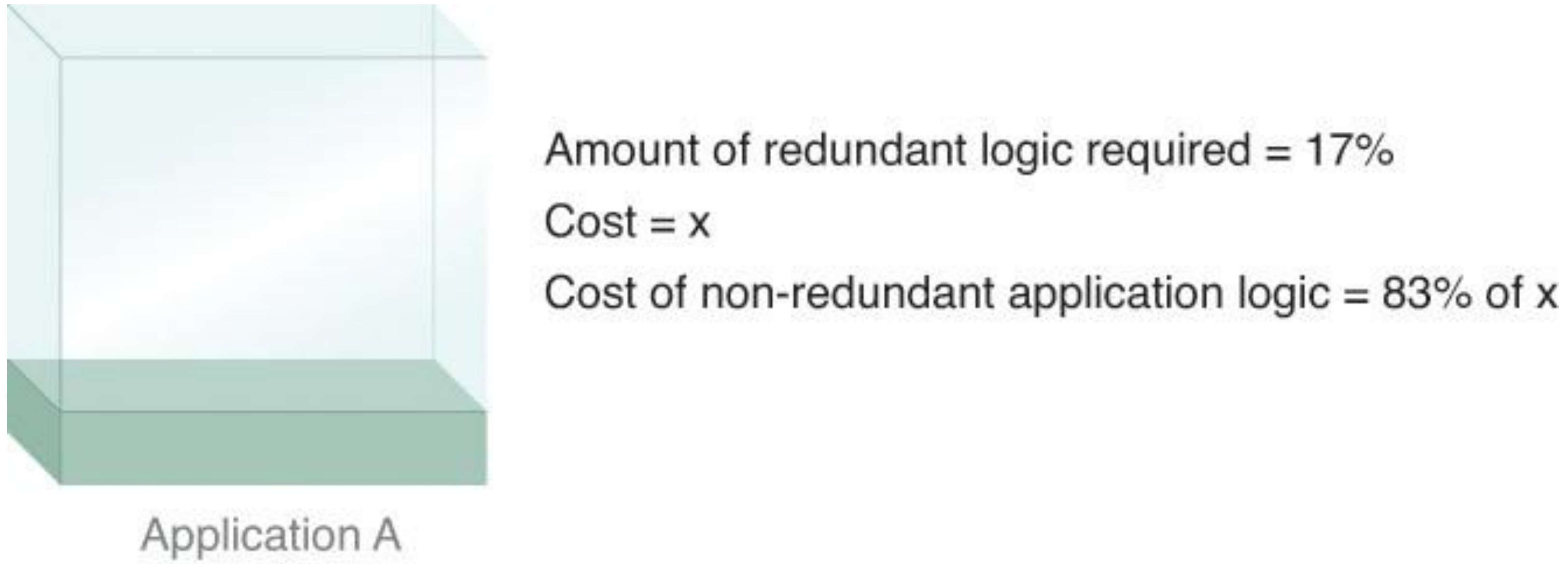

Amount of redundant logic required = 17%
Cost = x
Cost of non-redundant application logic = 83% of x
Application A

Figure 3.13 Application A was delivered for a specific set of business requirements. Because a subset of these business requirements had already been fulfilled elsewhere, Application A’s delivery scope is larger than it has to be

It Bloats an Enterprise

- **The ever-expanding hosting, maintenance, and administration demands can inflate an IT department in budget, resources, and size to the extent that IT becomes a significant drain on the overall organization**

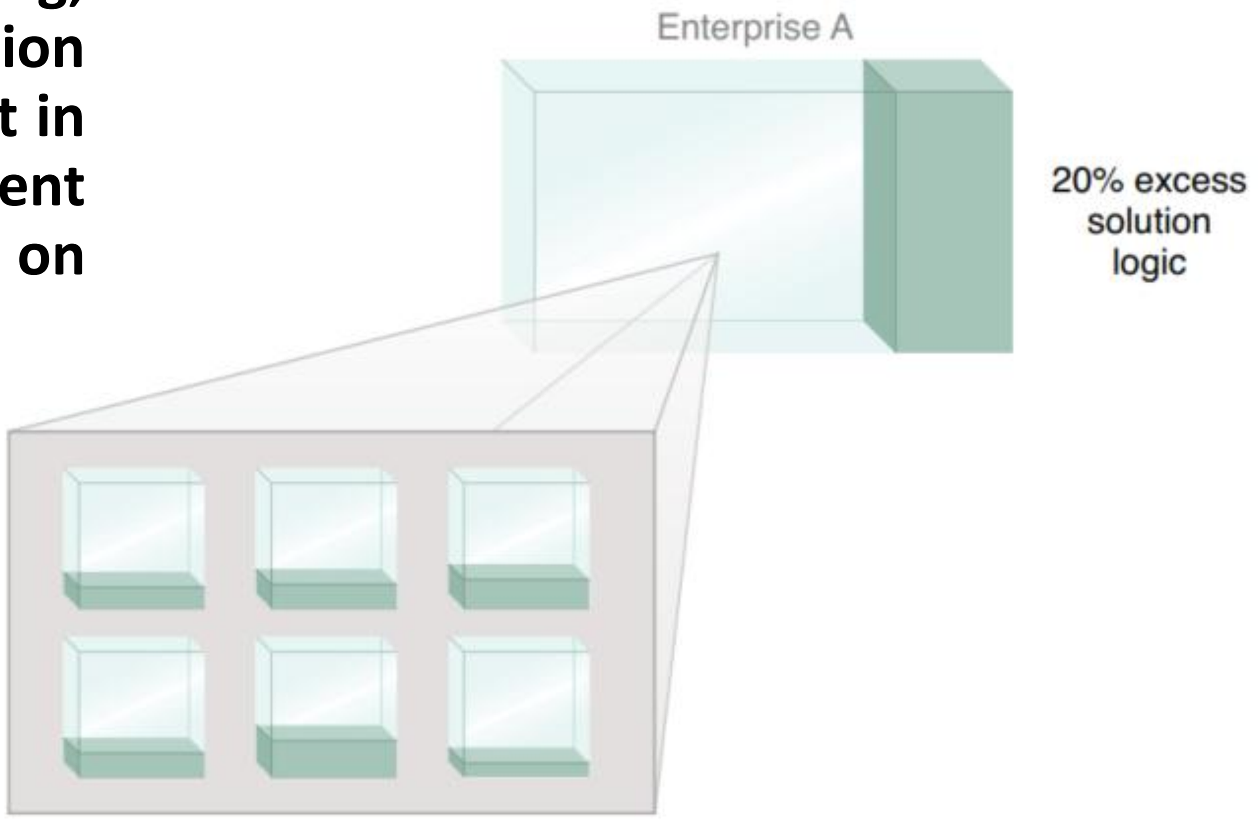


Figure 3.14 An enterprise environment containing applications with redundant functionality. The net effect is a larger enterprise

Complex Infrastructures and Convoluted Enterprise Architectures

- **Needing to host various applications developed using different technologies and platforms often means each has specific architectural requirements.**
- **The differences across these "siloed" applications can lead to a disjointed environment making it difficult to plan the enterprise's evolution and scale its infrastructure accordingly**

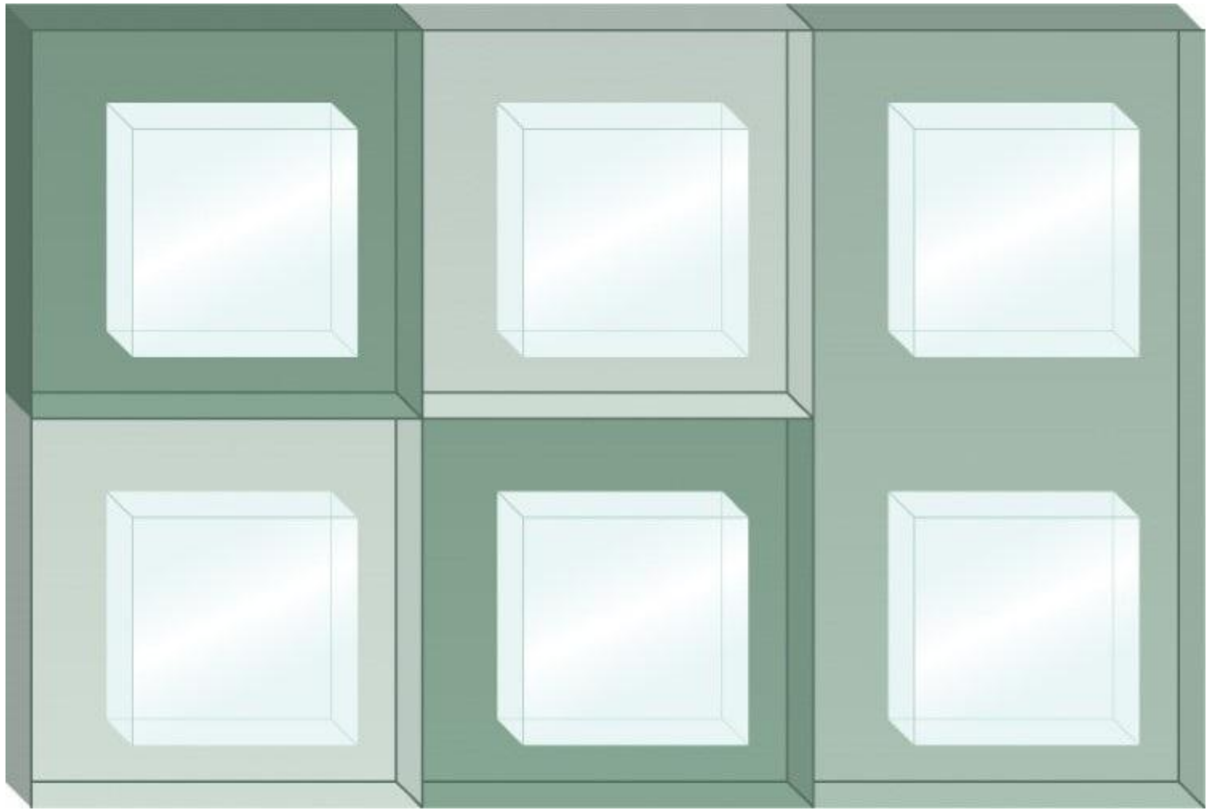


Figure 3.15 Different application environments within the same enterprise can introduce incompatible runtime platforms as indicated by the shaded zones.

Integration Becomes a Constant Challenge

- **Applications built only with the automation of specific business processes in mind are generally not designed to accommodate other interoperability requirements**

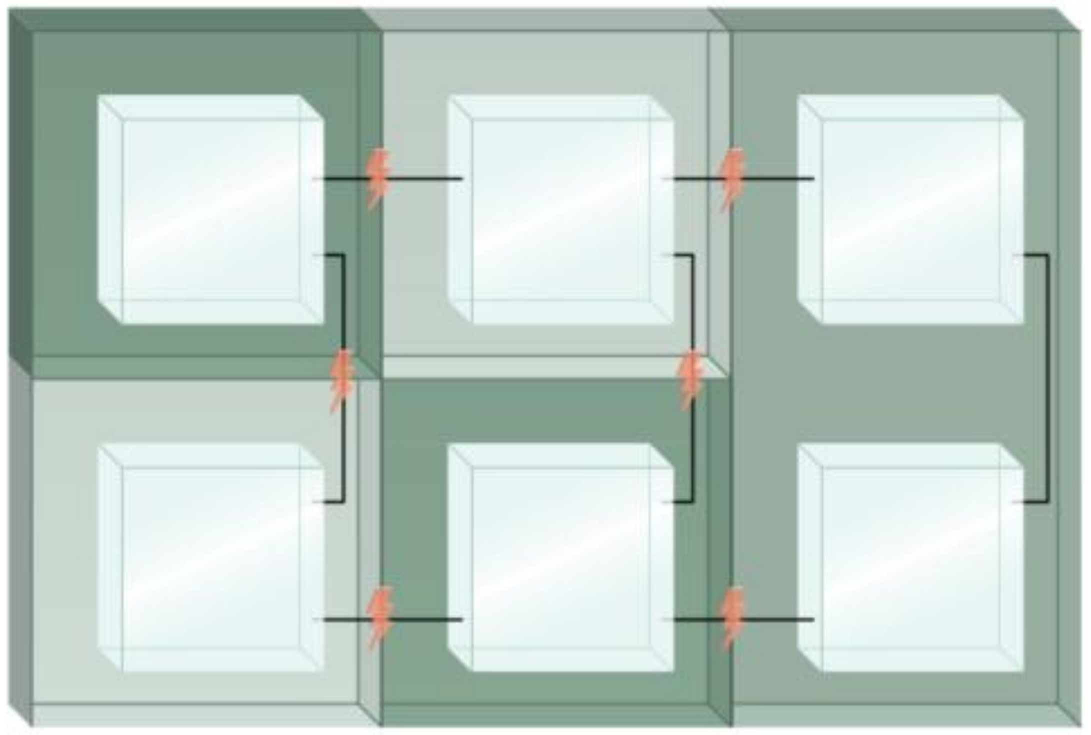


Figure 3.16 A vendor-diverse enterprise can introduce a variety of integration challenges, as expressed by the little lightning bolts that highlight points of concern when trying to bridge proprietary environments.

The Need for Service-Orientation (1)

1. **Increased Amounts of Reusable Solution Logic**
   - Within a service-oriented solution, units of logic (services) encapsulate functionality not specific to any one application or business process

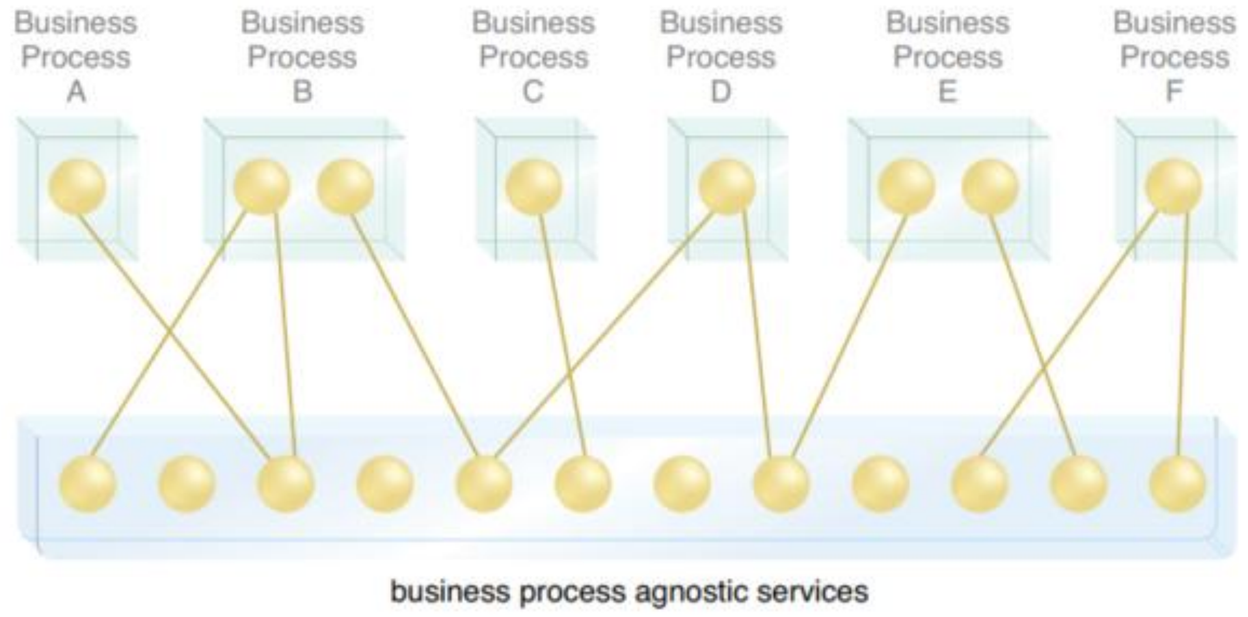


Figure 3.17 Business processes are automated by a series of business process–specific services (top layer) that share a pool of business process–agnostic services (bottom layer). These layers correspond to service models described in Chapter 5.

The Need for Service-Orientation (2)

2. **Reduced Amounts of Application-Specific Logic**
   - This blurs the lines between standalone application environments by reducing the overall quantity of standalone applications.

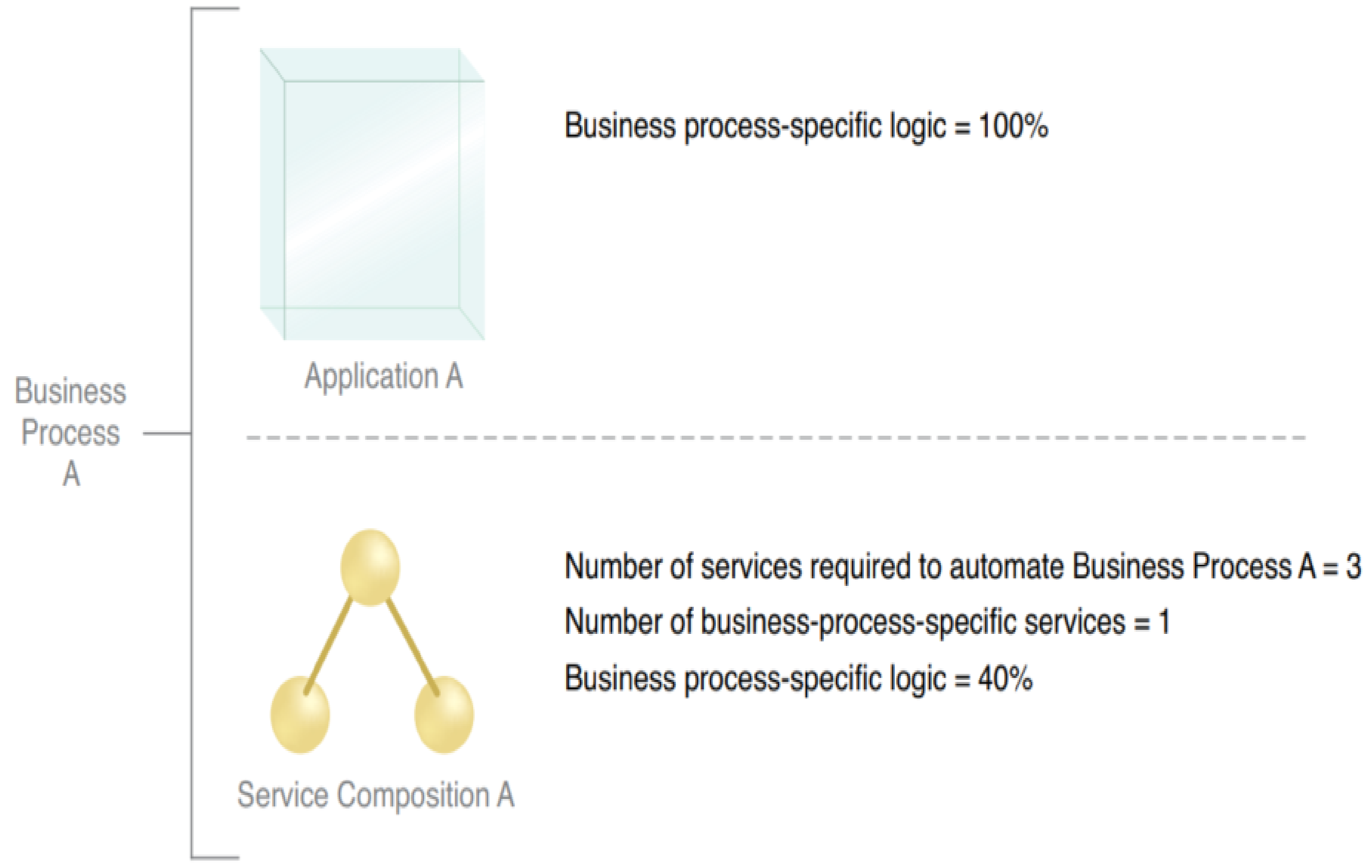

Business Process A
Application A
Business process-specific logic = 100%
Service Composition A
Number of services required to automate Business Process A = 3
Number of business-process-specific services = 1
Business process-specific logic = 40%

Figure 3.18 Business Process A can be automated by either Application A or Service Composition A. The delivery of Application A can result in a body of solution logic that is all specific to and tailored for the business process. Service Composition A would be designed to automate the process with a combination of reusable services and 40% of additional logic specific to the business process.

The Need for Service-Orientation (3)

3. **Reduced Volume of Logic Overall**
   - The overall quantity of solution logic is reduced because the same solution logic is shared and reused to automate multiple business processes

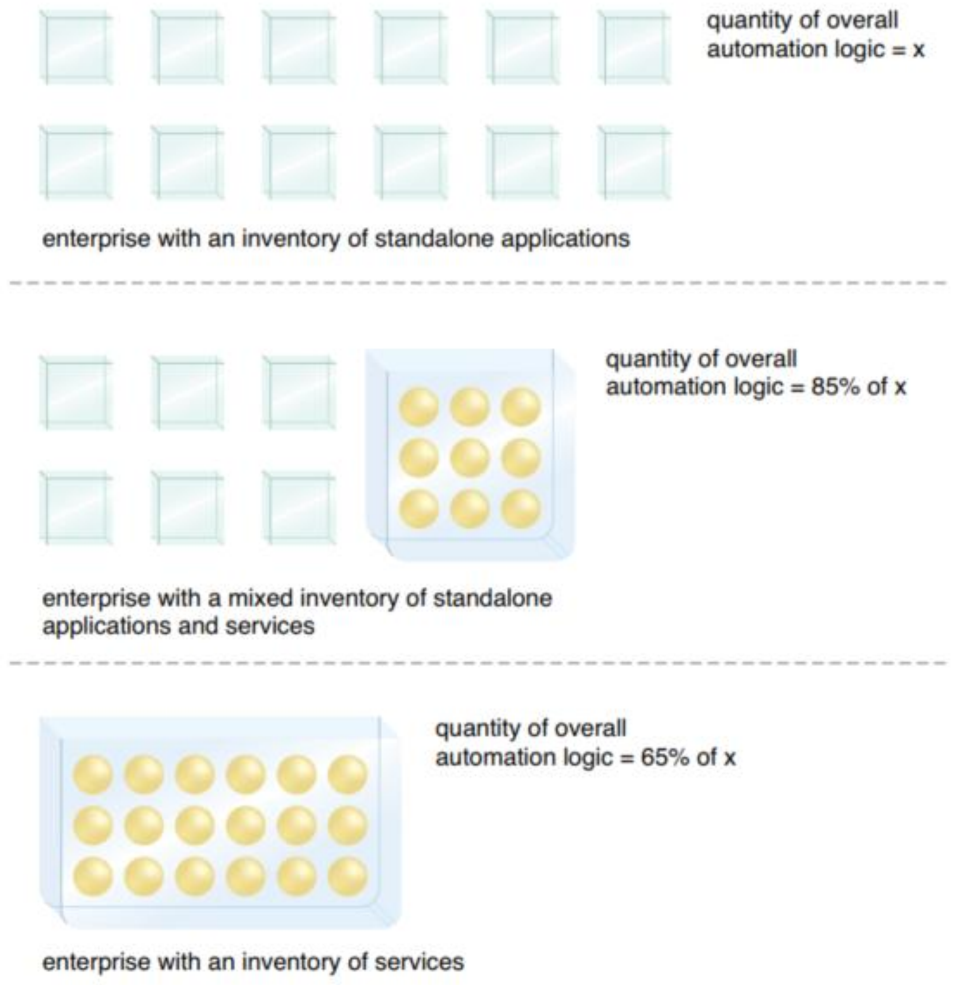

quantity of overall automation logic = x
enterprise with an inventory of standalone applications
quantity of overall automation logic = 85% of x
enterprise with a mixed inventory of standalone applications and services
quantity of overall automation logic = 65% of x
enterprise with an inventory of services

Figure 3.19 The quantity of solution logic shrinks as an enterprise transitions toward a standardized service inventory comprised of "normalized" services. (Service normalization is explained further at the end of Chapter 5.)

The Need for Service-Orientation (4)

4. Inherent Interoperability

- Common design characteristics consistently implemented result in solution logic that is naturally aligned
- When this carries over to the standardization of service contracts and their underlying data models, a base level of automatic interoperability is achieved across services

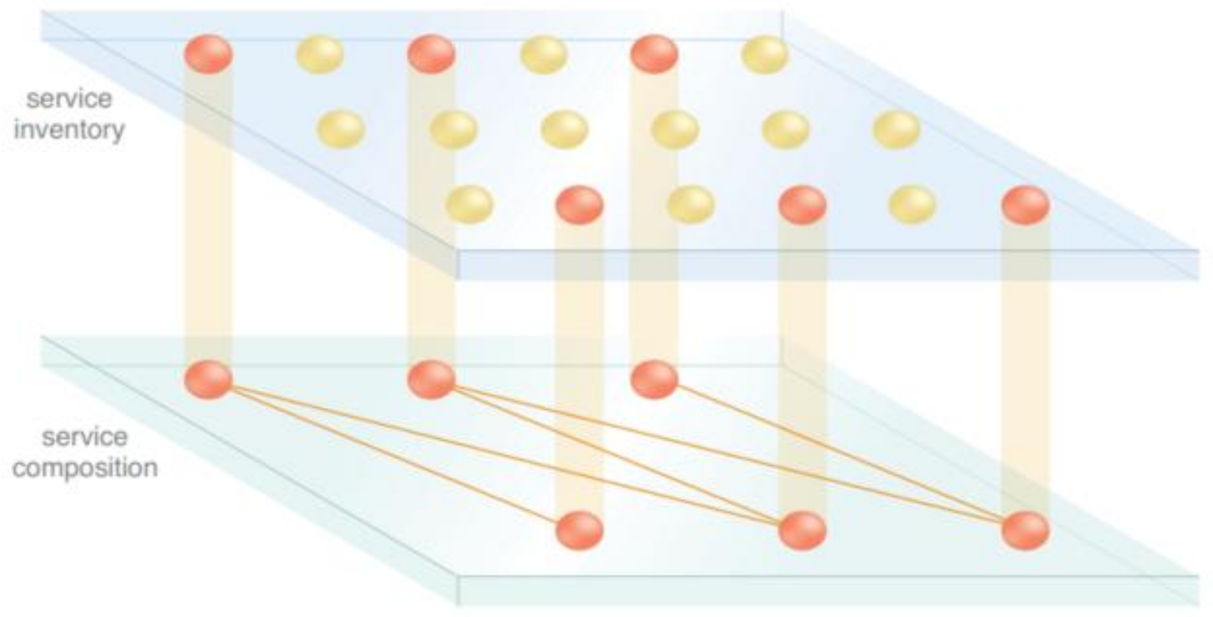
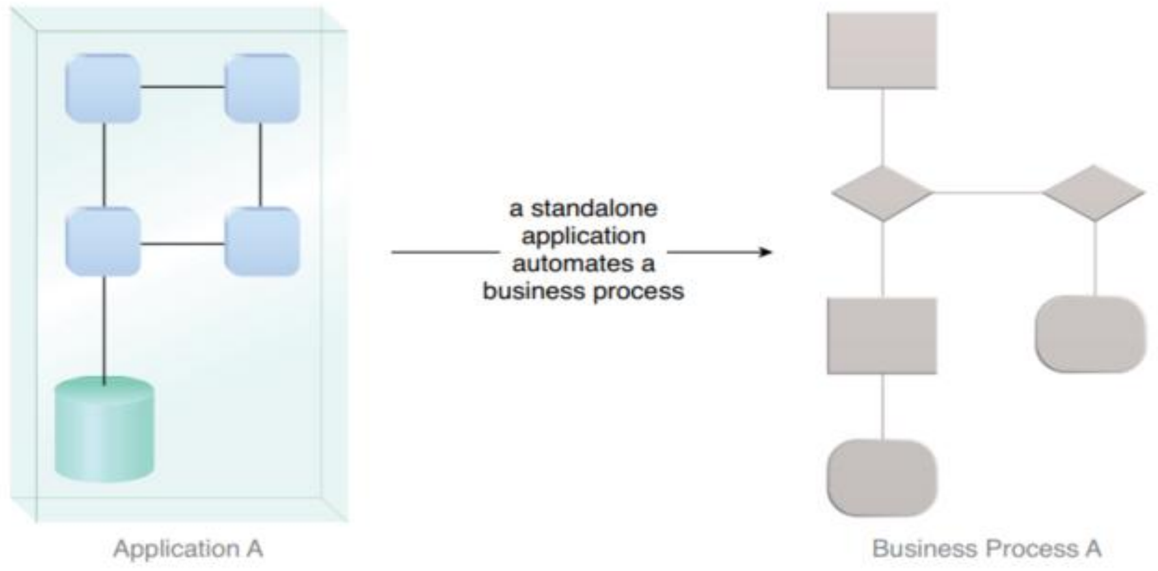


Figure 3.20 Services from different parts of a service inventory can be combined into new compositions. If these services are designed to be intrinsically interoperable, the effort to assemble them into new composition configurations is significantly reduced.

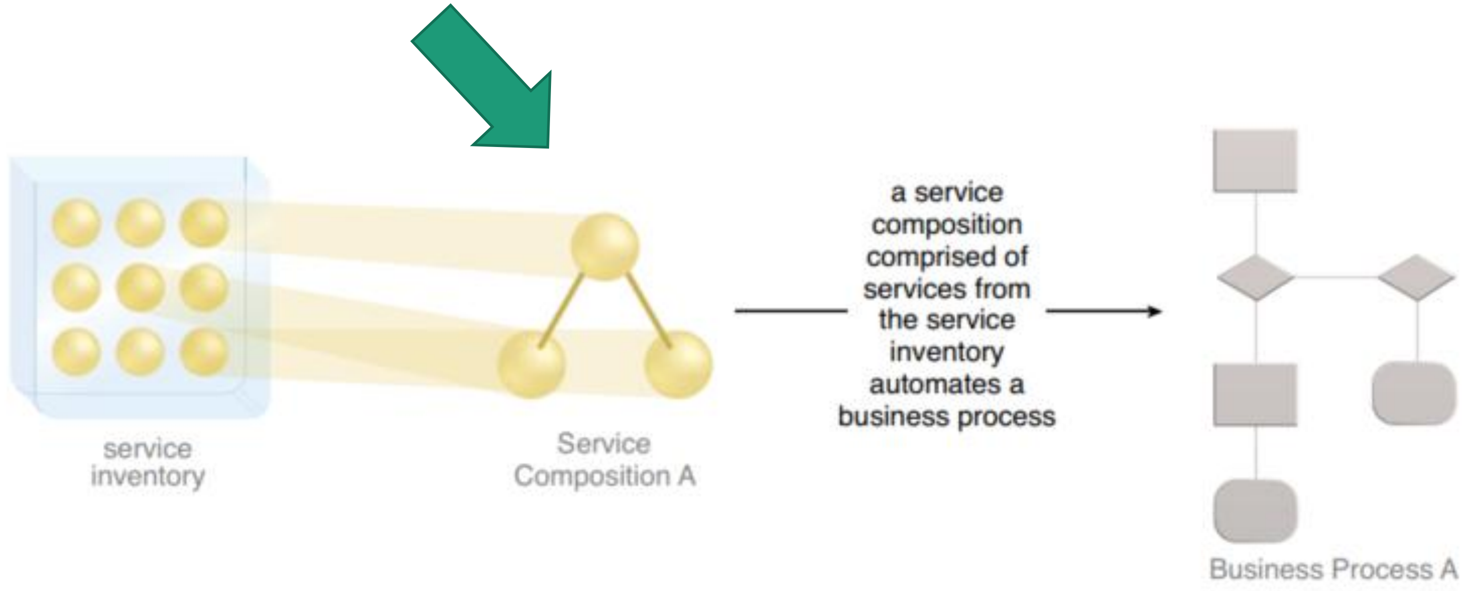
3.3 Effects of Service-Orientation on the Enterprise

Service-Orientation and the Concept of "Application"

- **Applications no longer consist of self-contained bodies of programming logic responsible for automating a specific set of tasks**



-> Service composition



service inventory
Service Composition A
a service composition comprised of services from the service inventory automates a business process
Business Process A

Service-Orientation and the Concept of “Integration”

- **In the past, integrating something implied connecting two or more applications or programs that may or may not have been compatible**

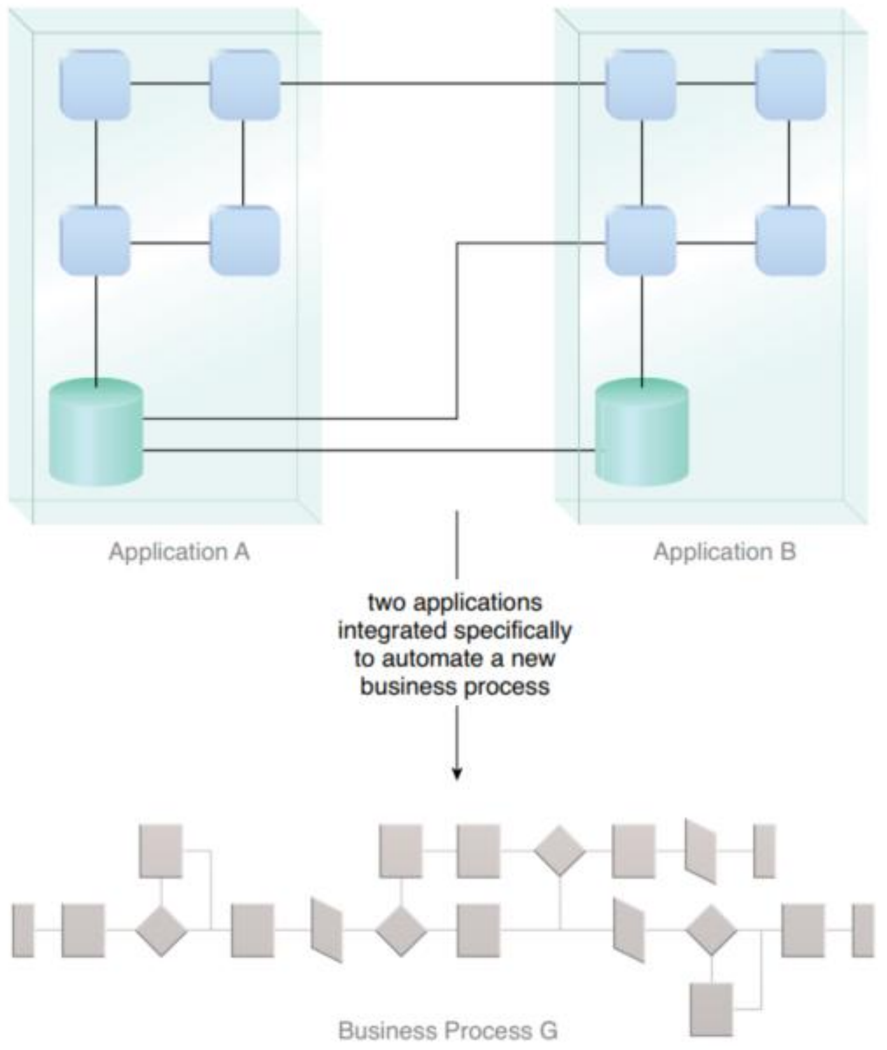

Application A
Application B
two applications integrated specifically to automate a new business process
Business Process G

Figure 3.23 The traditional integration architecture, comprised of two or more applications connected in different ways to fulfill a new set of automation requirements (as dictated by the new Business Process G)

Service-Orientation and the Concept of "Integration"

- **As a result, the concept of integration begins to fade. Exchanging data between different units of solution logic becomes a natural and secondary design characteristic**

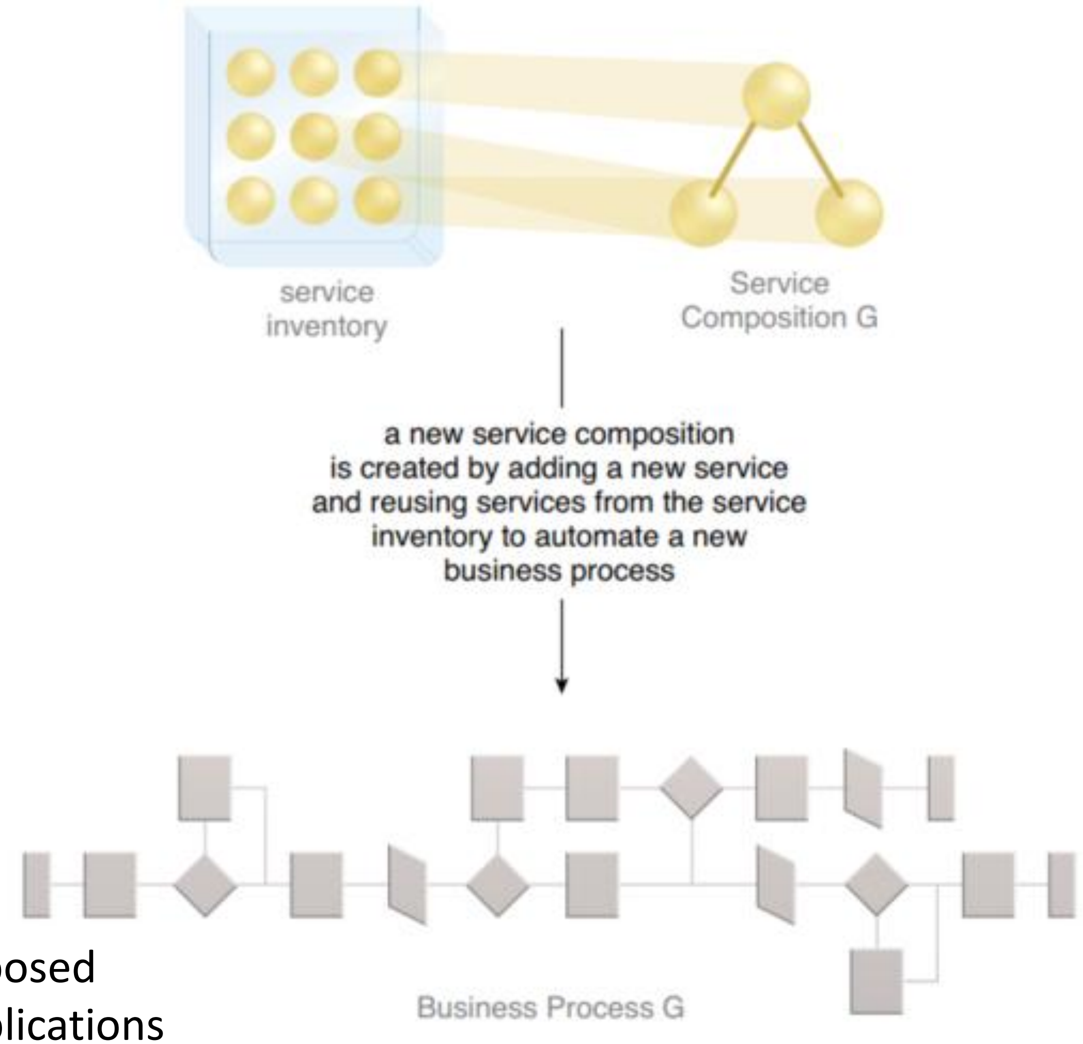
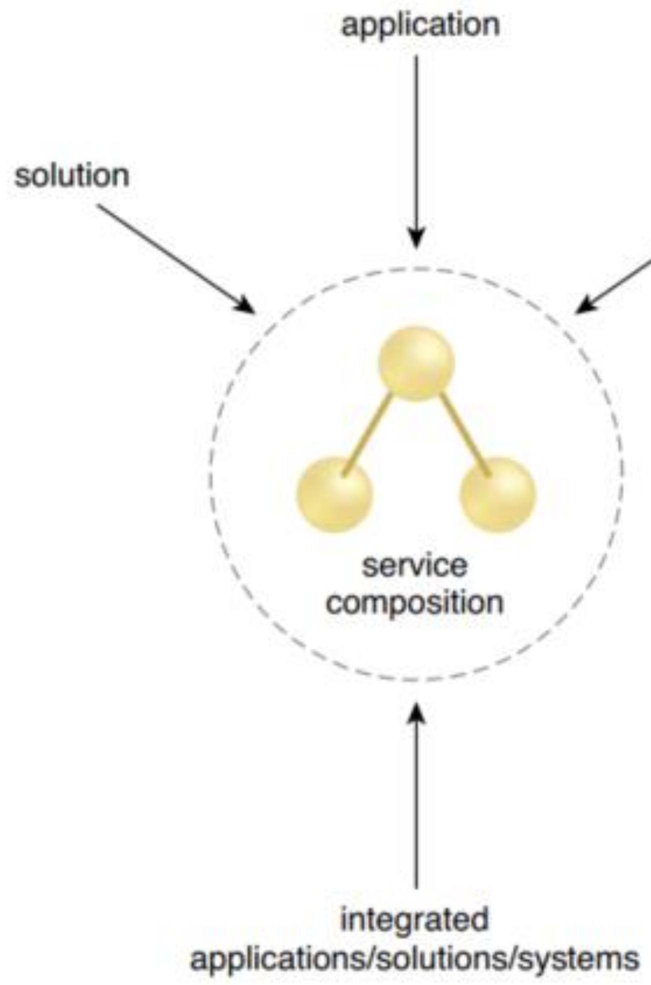

service inventory
Service Composition G
a new service composition is created by adding a new service and reusing services from the service inventory to automate a new business process
Business Process G

Figure 3.24 A new combination of services is composed together to fulfill the role of traditional integrated applications

The Service Composition

- **Applications, integrated applications, solutions, systems—all of these terms and what they have traditionally represented can be directly associated with the *service composition* in service-oriented**



3.4 Goals and Benefits of Service-Oriented

Strategic goals and benefits

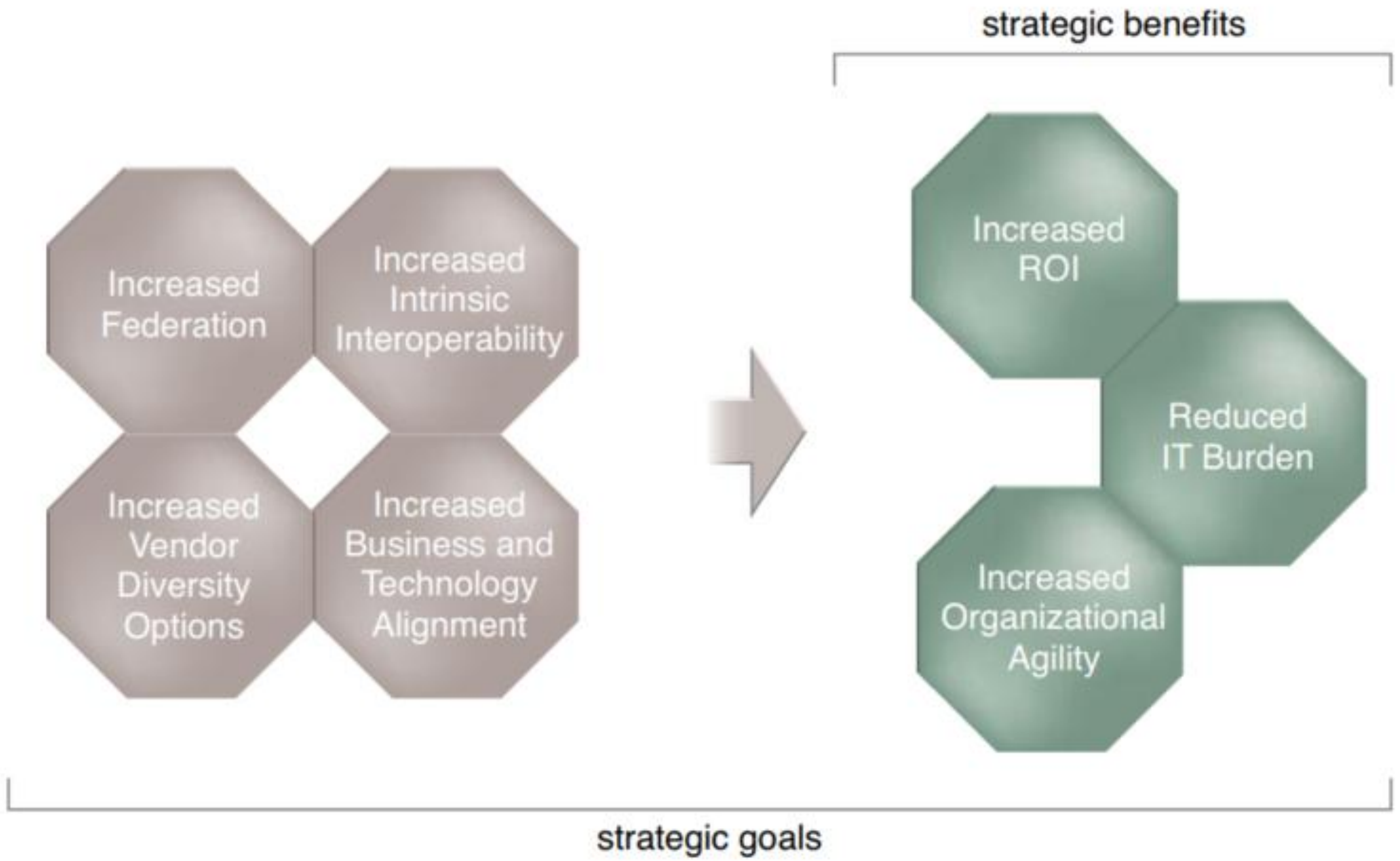

strategic benefits
Increased Federation
Increased Intrinsic Interoperability
Increased Vendor Diversity Options
Increased Business and Technology Alignment
Increased ROI
Reduced IT Burden
Increased Organizational Agility
strategic goals

Figure 3.26 The seven identified goals are interrelated and can be further categorized into two groups: strategic goals and resulting benefits. Increased organization agility, increased ROI, and reduced IT burden are concrete benefits resulting from the attainment of the remaining four goals.

G1: Increased Intrinsic Interoperability

- **Interoperability refers to the sharing of data.**
  - Integration can be seen as a process that enables interoperability
- **Intrinsic interoperability represents a fundamental goal of service-orientation that establishes a foundation for the realization of other strategic goals and benefits**
  - Services are designed to be intrinsically interoperable regardless of when and for which purpose they are delivered.

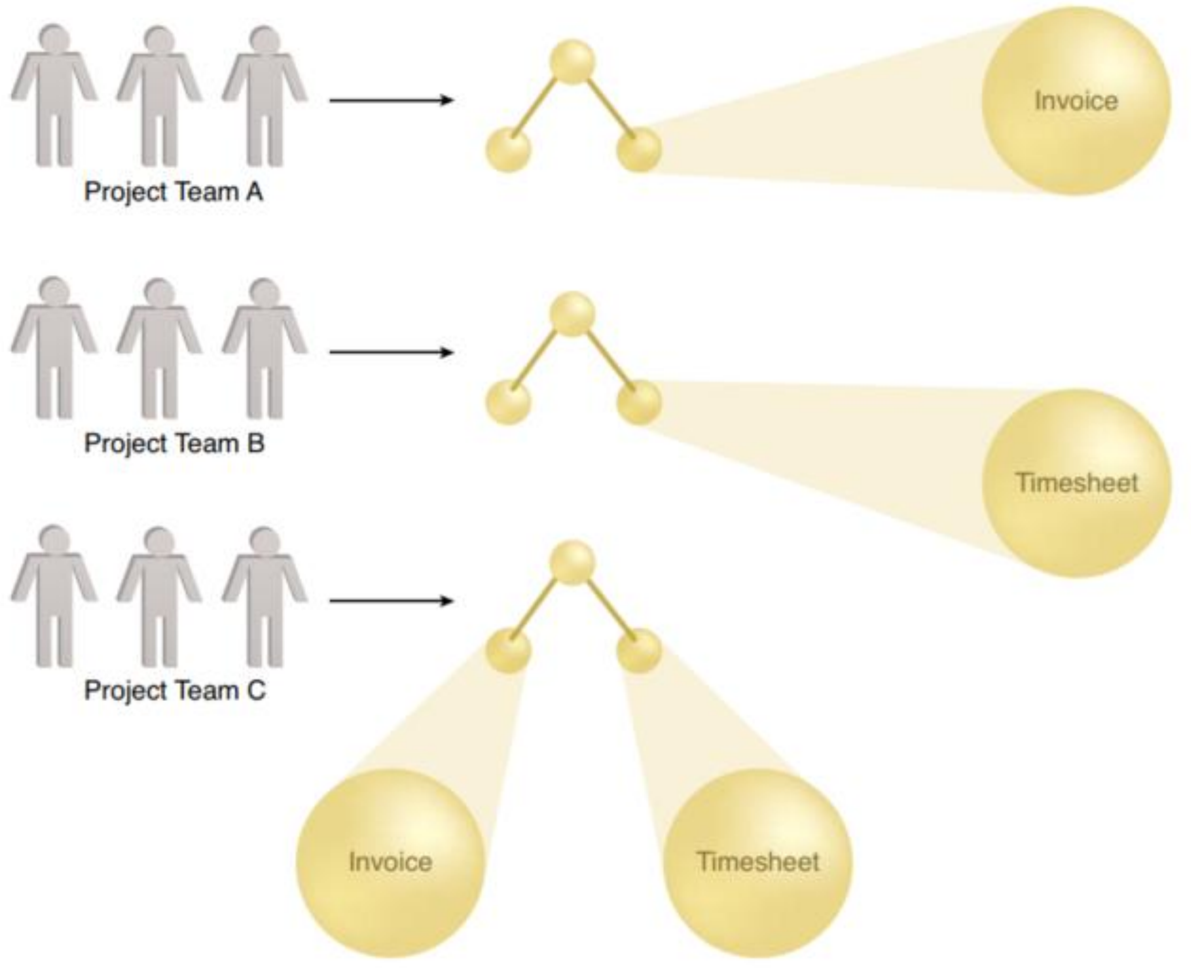


Figure 3.27 the intrinsic interoperability of the Invoice and Timesheet services delivered by Project Teams A and B allow them to be combined into a new service composition by Project Team C.

G2: Increased Federation

- **Service orientation aims to increase a federated perspective of an enterprise to whatever extent it is applied**

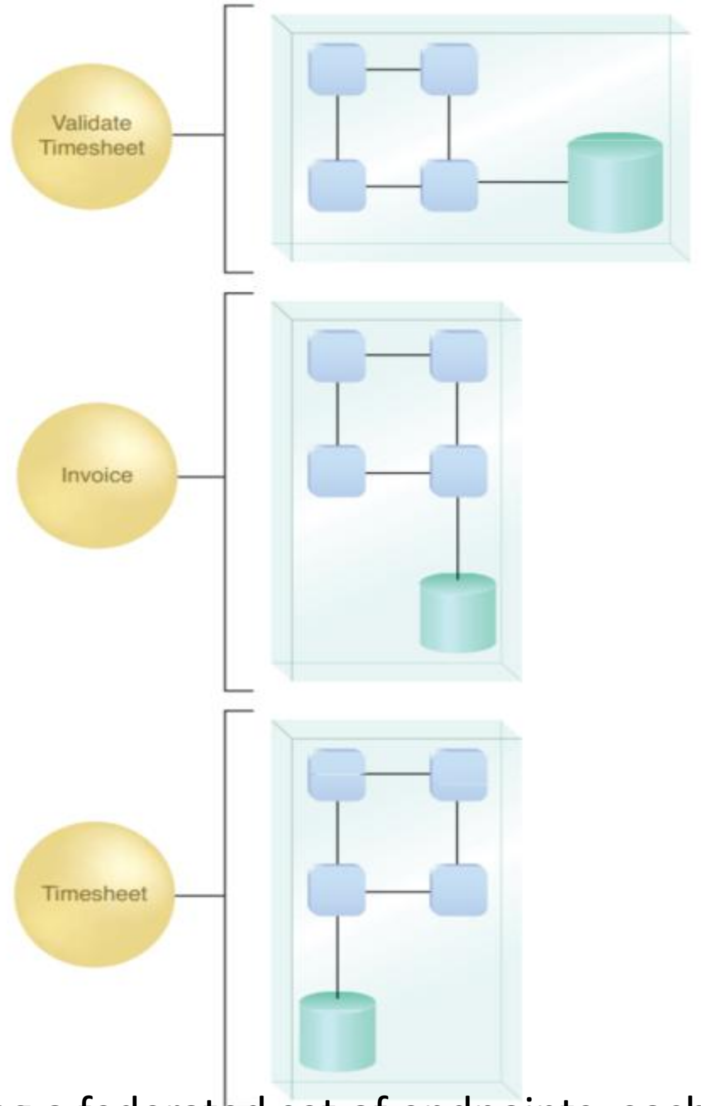


Figure 3.28 Three service contracts establishing a federated set of endpoints, each of which encapsulates a different implementation.

G3: Increased Vendor Diversification Options

- **The ability an organization has to pick and choose "best-ofbreed" vendor products and technology innovations and use them together within one enterprise**
  - To have and retain this option requires that **its technology architecture** not be tied or locked into any one specific vendor platform

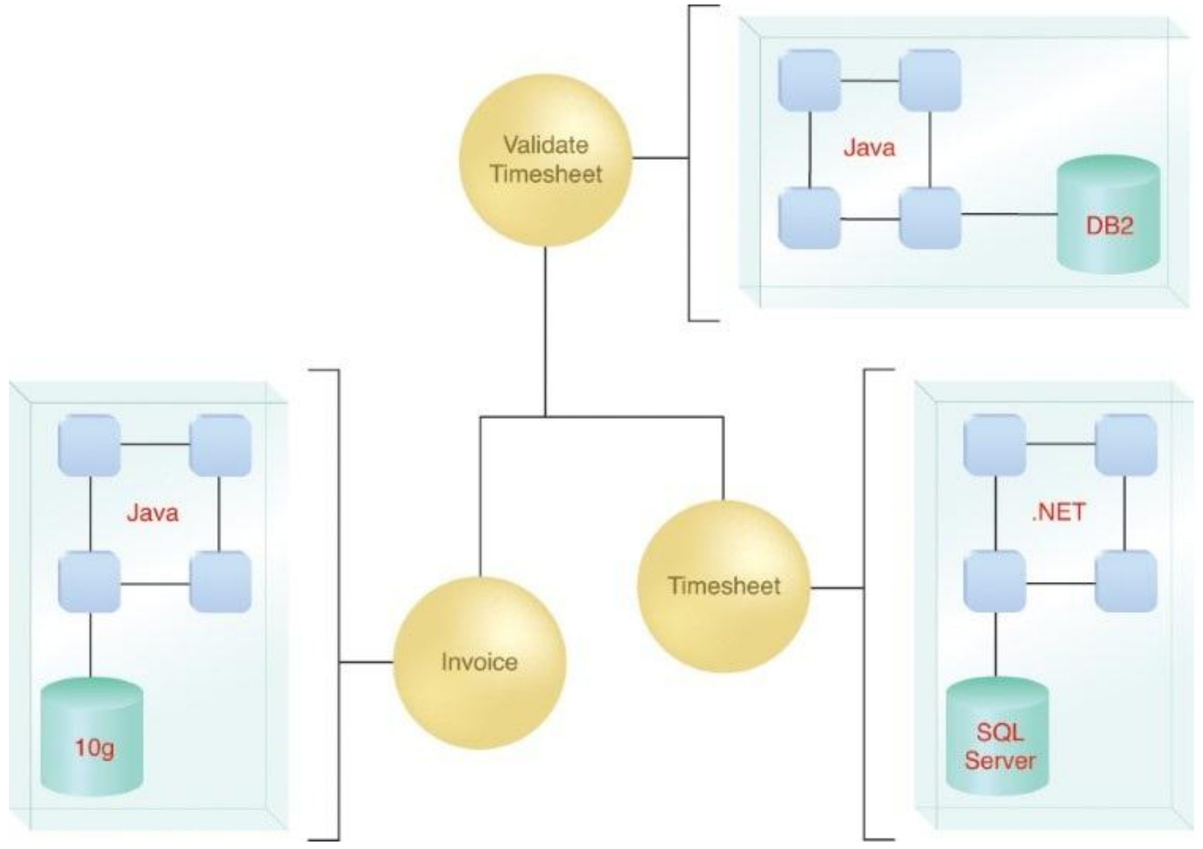
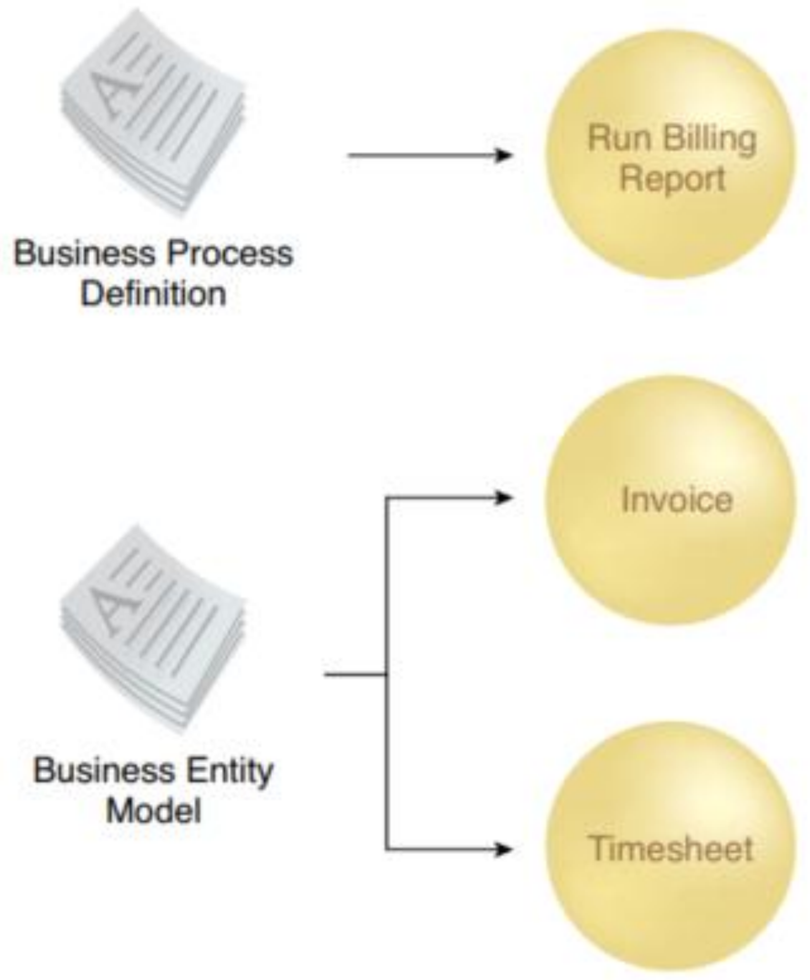


Figure 3.29 A service composition consisting of three services, each of which encapsulates a different vendor automation environment.

G4: Increased Business and Technology Domain Alignment

- **Service-orientation enhances abstraction by establishing service layers that encapsulate business models, involving business experts in their definition.**
  - This aligns automation technology with business at an unprecedented level in service designs.



B1: Increased ROI

- **Service-orientation advocates the creation of *agnostic solution logic* that is agnostic to any one purpose and therefore useful for multiple purposes.**

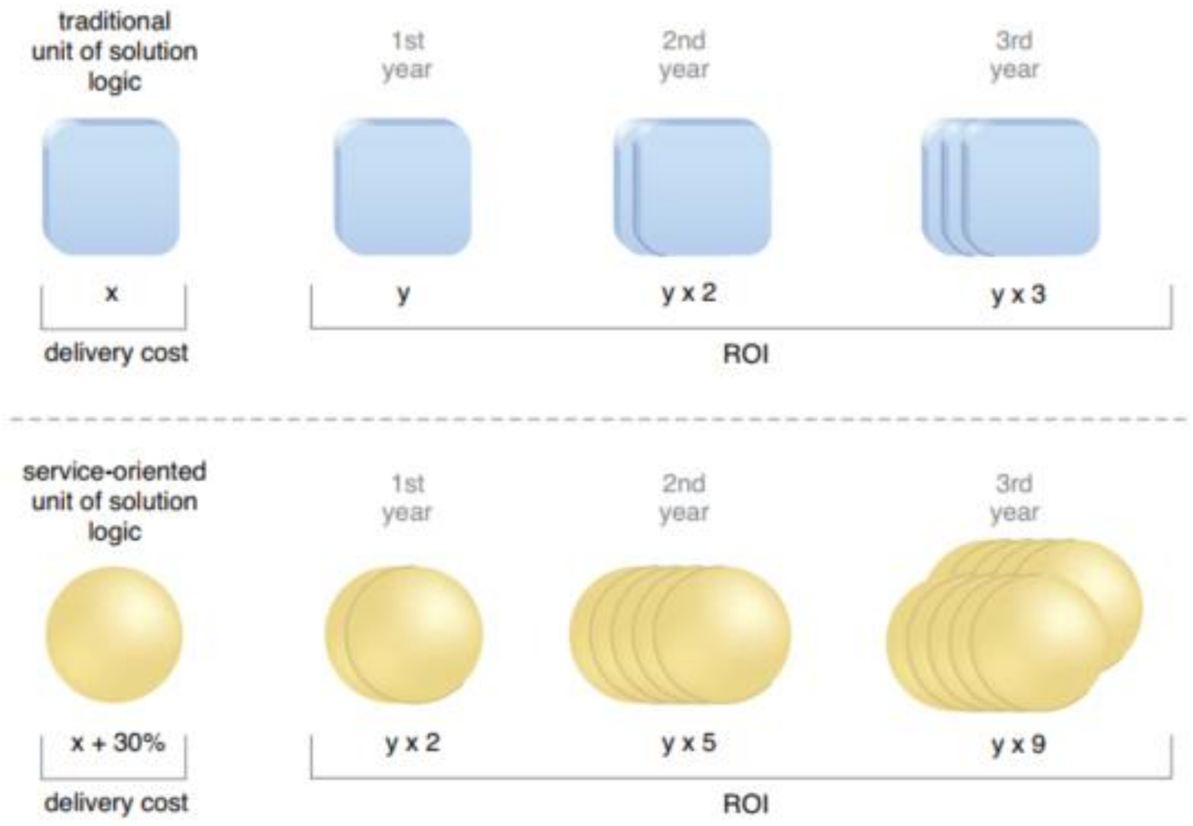


Figure 3.31

An example of the types of formulas being used to calculate ROI for SOA projects. More is invested in the initial delivery with the goal of benefiting from increased subsequent reuse.

B2: Increased Organizational Agility

- **Agility, on an organizational level, refers to efficiency with which an organization can respond to change**
- **Being able to more quickly adapt to industry changes and outmaneuver competitors has tremendous strategic significance**

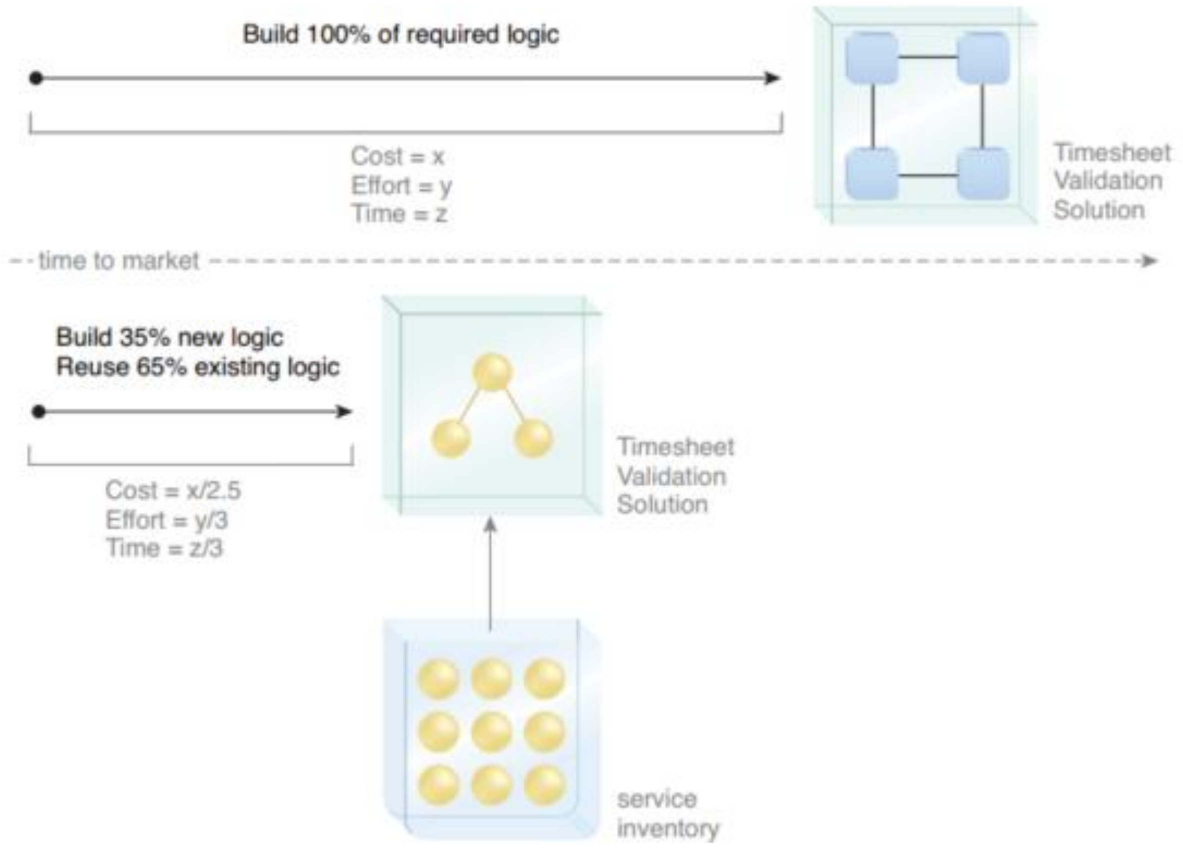
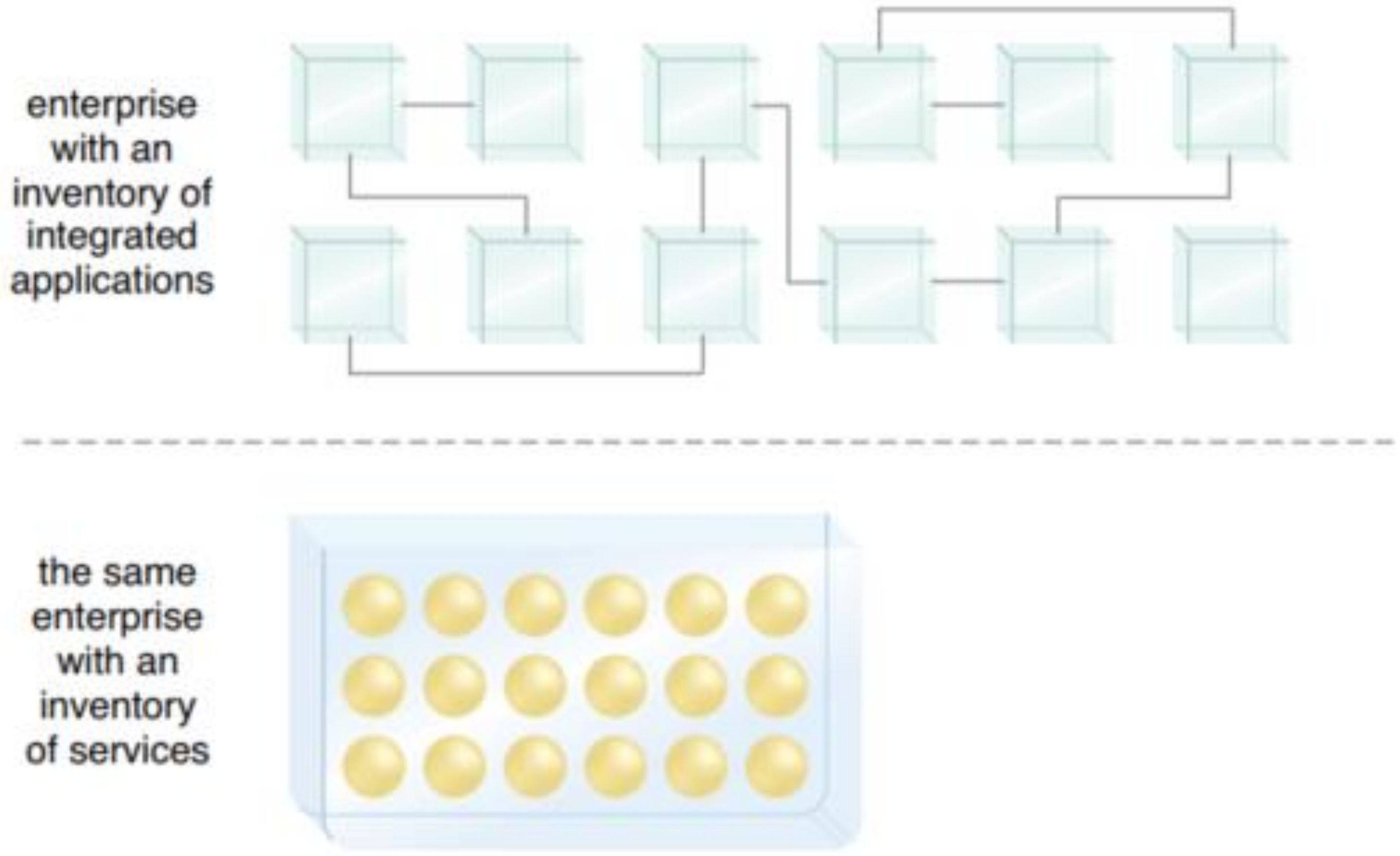


Figure 3.32

The delivery timeline is projected based on the percentage of "net new" solution logic that needs to be built. Though in this example only 35% of new logic is required, the timeline is reduced by around 50% because significant effort is still required to incorporate existing, reusable services from the inventory.

B3: Reduced IT Burden

- **Service-orientation reduces waste and operational costs in IT, creating a leaner, more agile department that actively contributes to the organization's strategic goals**


enterprise with an inventory of integrated applications
the same enterprise with an inventory of services

3.5 Four Pillars of Service-Orientation

The four pillars of service-orientation

1. **Teamwork**
2. **Education**
3. **Discipline**
4. **Balanced Scope**

The scope of SOA adoption can vary.
Keep efforts manageable and within meaningful boundaries

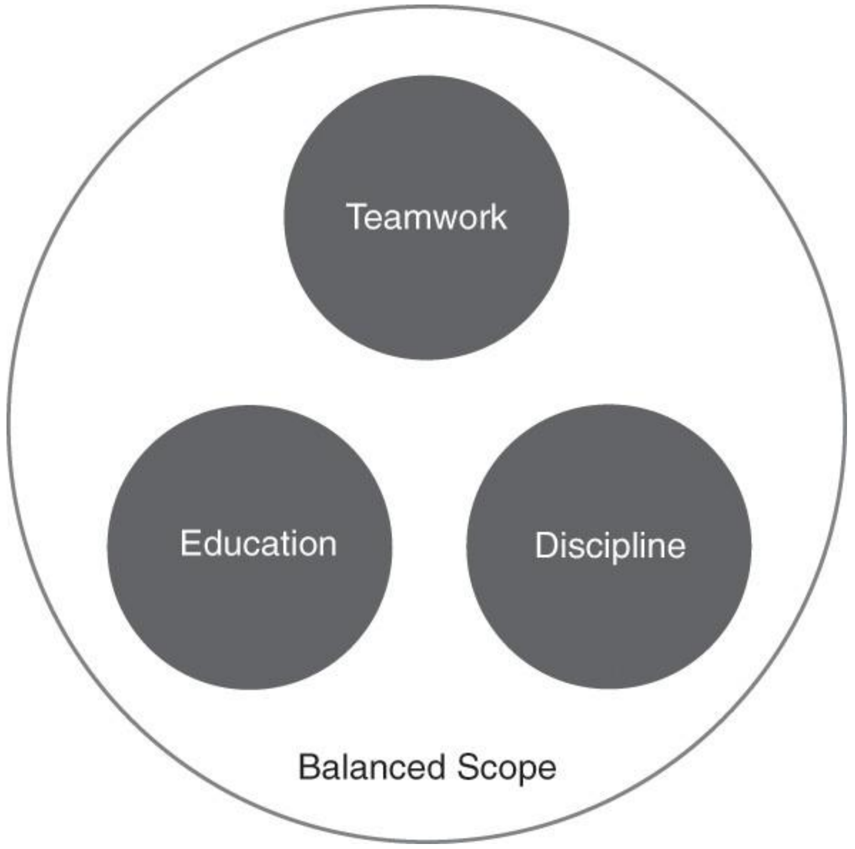


Figure 3.35 The Balanced Scope pillar encompasses and sets the scope at which the other three pillars are applied for a given adoption effort.

Determining a balanced scope

- **Common factors involved in determining a balanced scope include**
  - Cultural obstacles
  - Authority structures
  - Geography
  - Business domain alignment
  - Available stakeholder support and funding
  - Available IT resources
- **A single organization can choose one or more balanced adoption scopes**

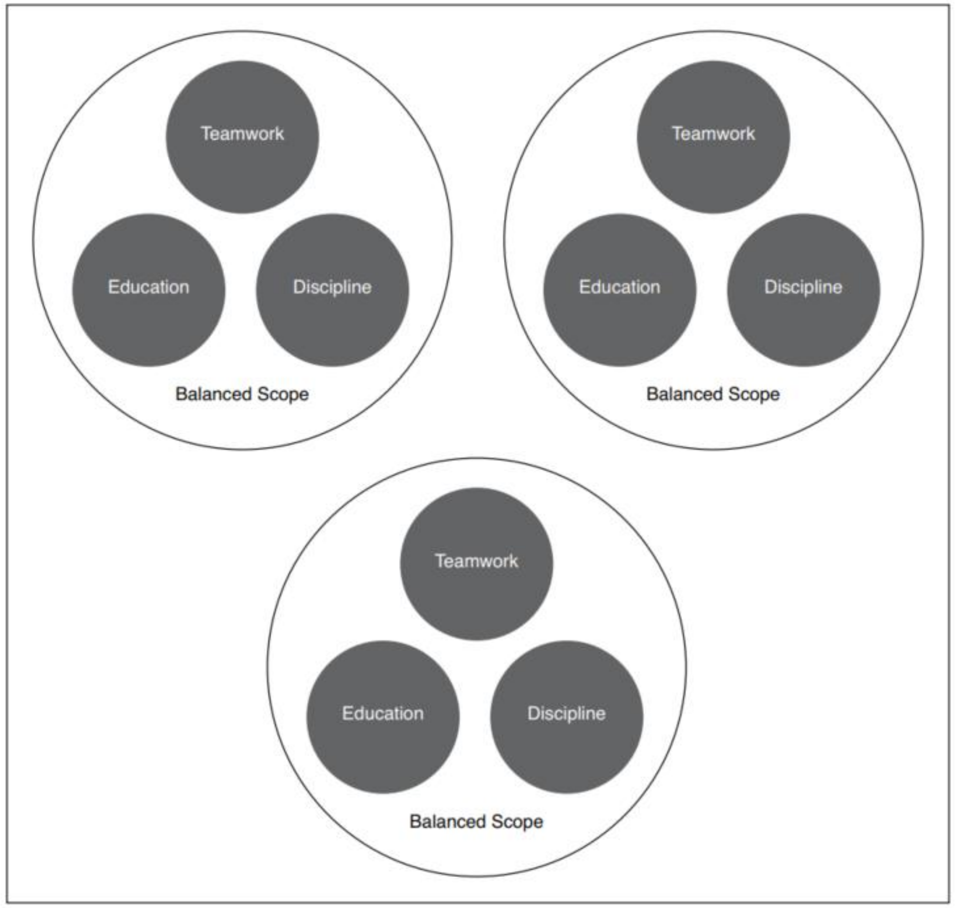


Figure 3.36 Multiple balanced scopes can exist within the same IT enterprise. Each represents a separate domain service inventory that is independently standardized, owned, and governed.

Q & A