Module Six: Service Engineering

Our materials for this module:

- **Thomas Erl, Service-Oriented Architecture, Concepts, Technology, and Design, ISBN： 9787030336422，**
  - Chapter 10
  - Chapter 11.1 and 11.2
  - Chapter 12.1
  - Chapter 13.1 and 13.5
  - Chapter 14.1
  - Chapter 15.1, 15.2, 15.3, and15.4
  - Chapter 16.3

Module 6 Learning Outcomes

- Understand the common phases of an SOA delivery lifecycle
- Understand the difference between different SOA delivery strategies
- Be able to conduct a service-oriented analysis
- Be able to use WSDL, SOAP and BPEL in service design

Guided questions for Module 6

- What is a series of steps that need to be completed to construct the services for a given service-oriented solution?
- What is a lifecycle of an SOA delivery project?
- What are SOA delivery lifecycle phases?
- What is the purpose of service-oriented analysis?
- What happens in service-oriented design phase of an SOA delivery lifecycle?
- What choices are made during service development?
- What key issue face service testers?
- What issues arise during service deployment?
- What application management issues come to the forefront during service administration?

Guided questions for Module 6

- What are three common SOA delivery strategies?
- What is the process of top-down strategy?
- What are the advantages and disadvantages of top-down strategy?
- What is the process of bottom-up strategy?
- What are the advantages and disadvantages of bottom-up strategy?
- What is the process of agile strategy?
- What are the advantages and disadvantages of agile strategy?
- How do we use WSDL, SOAP and BPEL in service design?

SOA Lifecycle

Service Engineering

- also called service-oriented software engineering
- a software engineering process that attempts to decompose the system into self-running units that either perform services or expose services

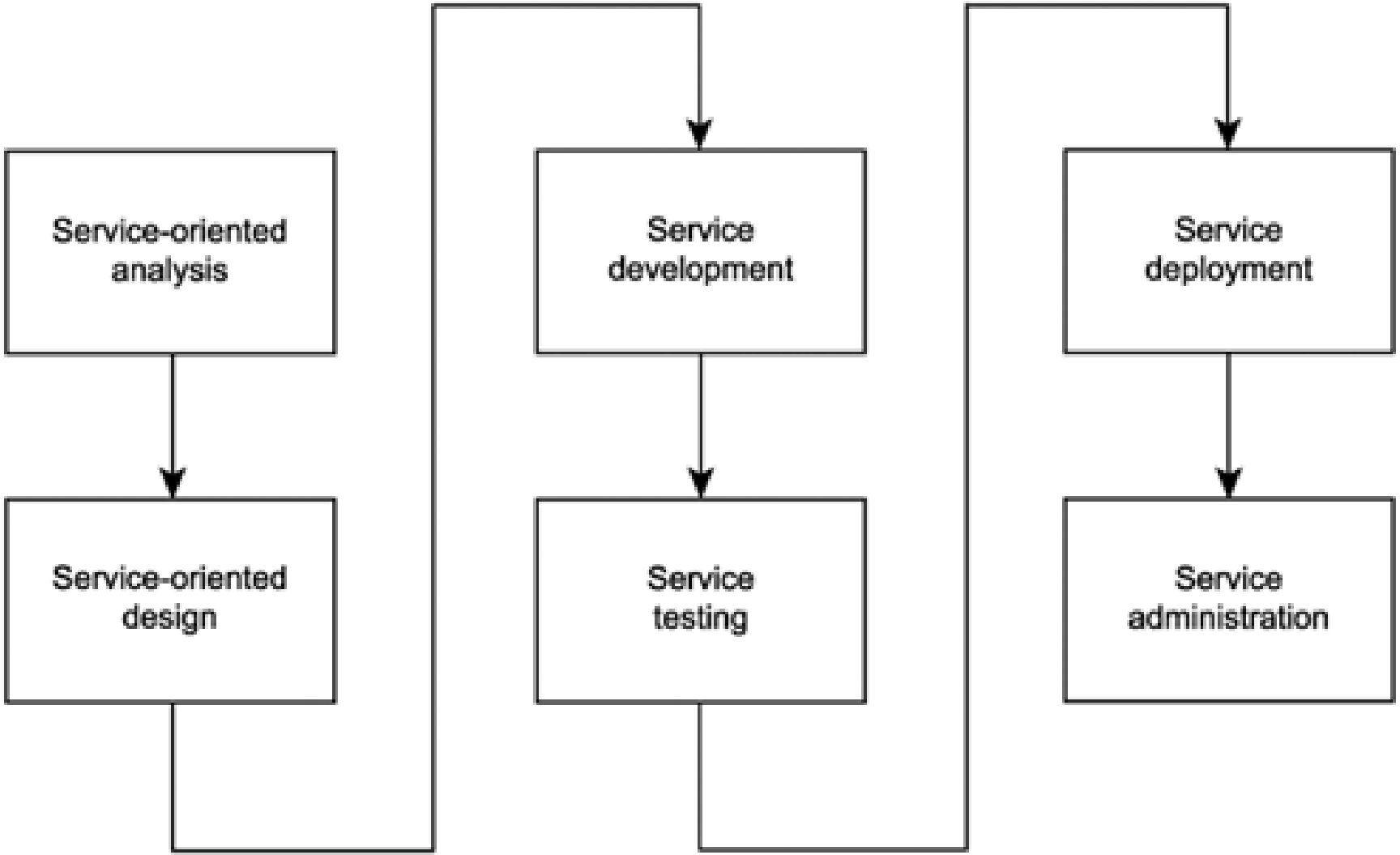
SOA lifecycle

- A series of steps that need to be completed to construct the services for a given service-oriented solution.
- The basic SOA lifecycle consists of a series of phases similar to those used for regular development projects

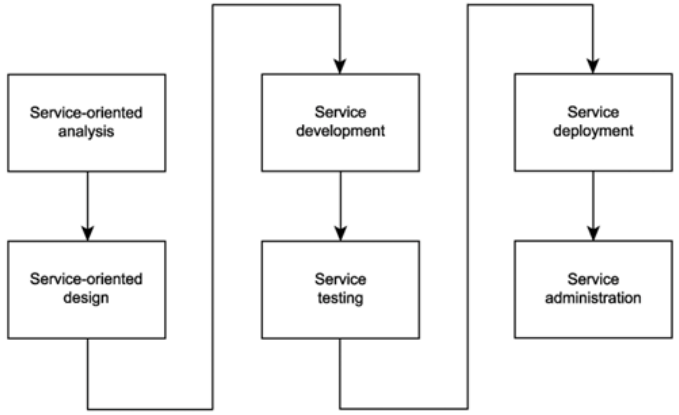
SOA lifecycle

- SOA introduces unique considerations in every phase of service construction and delivery

Common phases of an SOA delivery lifecycle

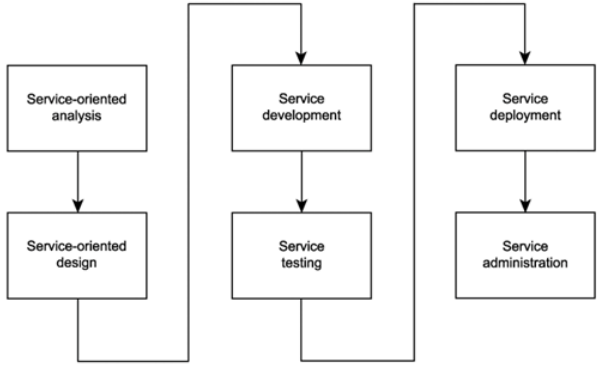


Service-oriented analysis



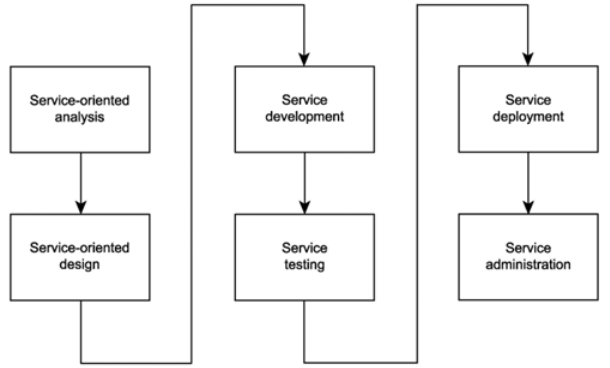
- In this initial stage that we determine the potential scope of our SOA
- Service layers are mapped out
- Individual services are modeled as service candidates that comprise a preliminary SOA

Service-oriented design



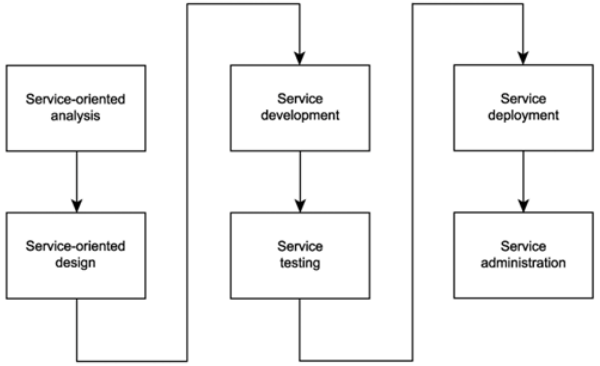
- When we know what it is we want to build, we need to determine how it should be constructed.
- Service-oriented design is a heavily standards-driven phase that incorporates industry conventions and service-orientation principles into the service design process.
- This phase, therefore, confronts service designers with key decisions that establish the hard logic boundaries encapsulated by services. The service layers designed during this stage can include the orchestration layer, which results in a formal business process definition.

Service development



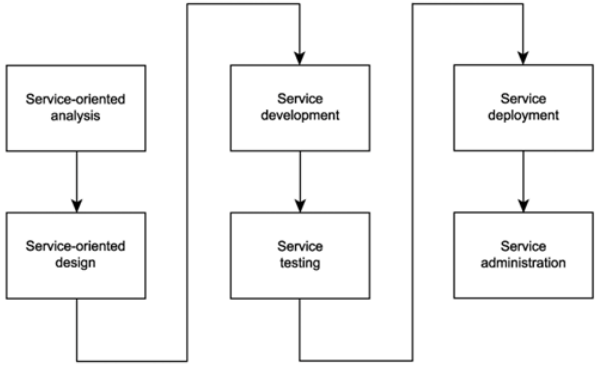
- The actual construction phase.
- Development platform-specific issues come into play, regardless of service type.
  - The choice of programming language and development environment will determine the physical form services and orchestrated business processes take, in accordance with their designs.

Service testing (1)



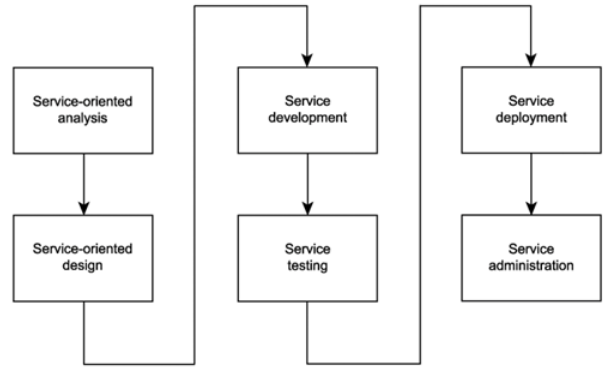
- Given their generic nature and potential to be reused and composed in unforeseeable situations, services are required to undergo rigorous testing prior to deployment into a production environment.

Service testing (2)



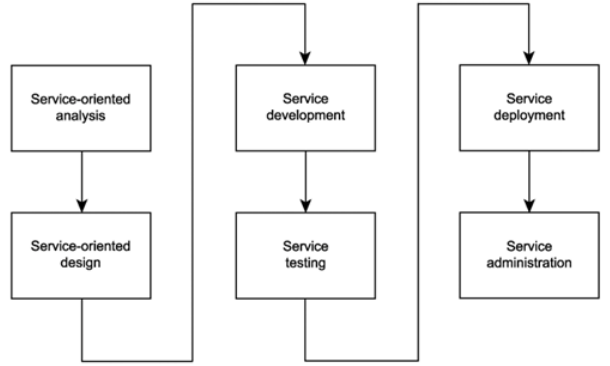
- Some of the key issues facing service testers:
  - What types of service requestors could potentially access a service?
  - Can all service policy assertions be successfully met?
  - What types of exception conditions could a service be potentially subjected to?
  - How well do service descriptions communicate service semantics?
  - Do revised service descriptions alter or extend previous versions?

Service testing (3)



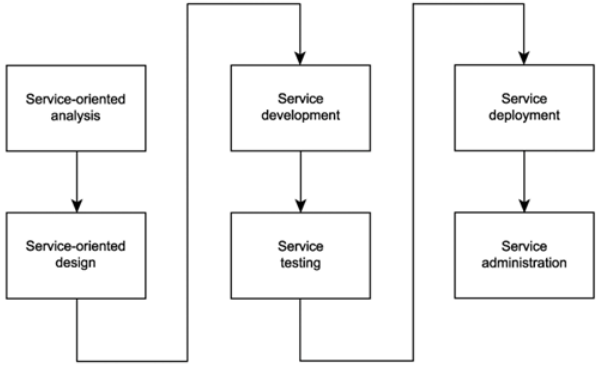
- Some of the key issues facing service testers:
  - How easily can the services be composed?
  - How easily can the service descriptions be discovered?
  - Is compliance to WS-I profiles required?
  - What data typing-related issues might arise?
  - Have all possible service activities and service compositions been mapped out?
  - Have all compensation processes been fully tested?

Service testing (4)



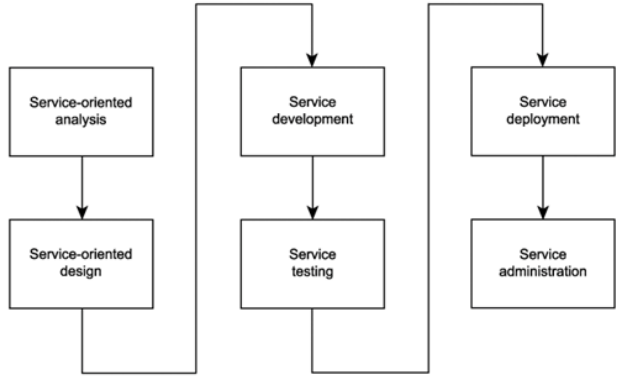
- Some of the key issues facing service testers:
  - What happens if exceptions occur within compensation processes?
  - Do all new services comply with existing design standards?
  - Do new services introduce custom SOAP headers? And, if yes, are all potential requestors (including intermediaries) required to do so, capable of understanding and processing them?
  - Do new services introduce functional or QoS requirements that the current architecture does not support?

Service deployment (1)



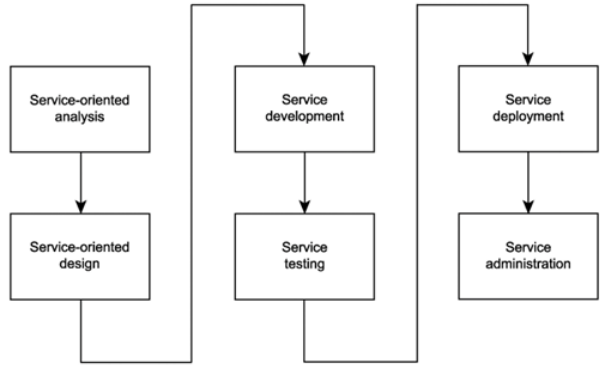
- Installing and configuring distributed components, service interfaces, and any associated middleware products onto production servers.

Service deployment (2)



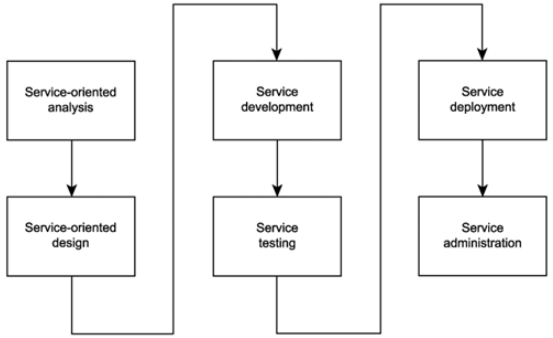
- Typical issues that arise during this phase include:
  - How will services be distributed?
  - Is the infrastructure adequate to fulfill the processing requirements of all services?
  - How will the introduction of new services affect existing services and applications?
  - How should services used by multiple solutions be positioned and deployed?
  - How will the introduction of any required middleware affect the existing environment?

Service deployment (3)



- Typical issues (cont):
  - Do these services introduce new versions of service descriptions that will need to be deployed alongside existing versions?
  - What security settings and accounts are required?
  - How will service pools be maintained to accommodate planned or unforeseen scalability requirements?
  - How will encapsulated legacy systems with performance or reliability limitations be maintained and monitored?

Service administration



- After services are deployed, standard application management issues come to the forefront.
- Issues frequently include:
  - How will service usage be monitored?
  - What form of version control will be used to manage service description documents?
  - How will messages be traced and managed?
  - How will performance bottlenecks be detected?

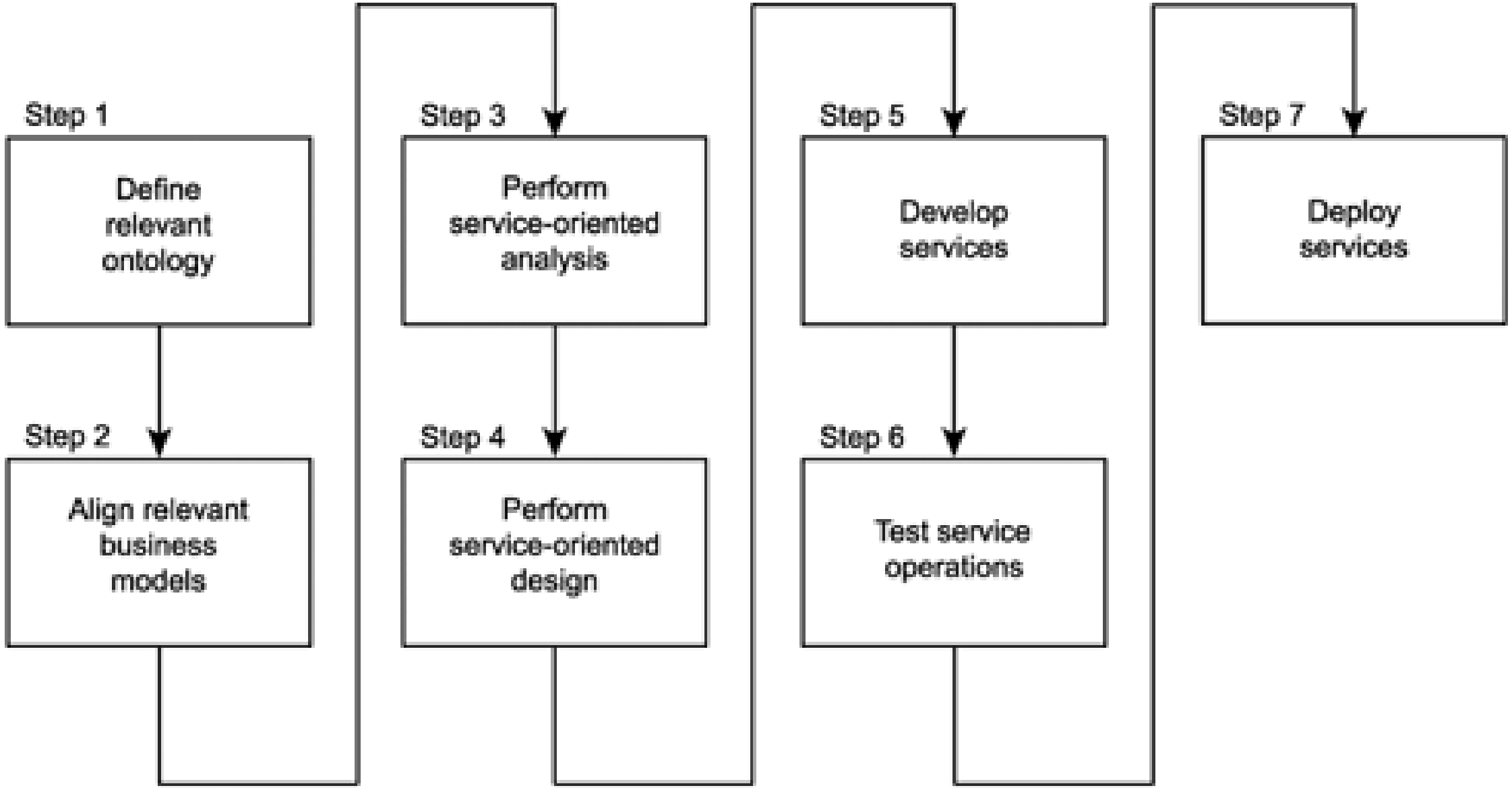
SOA Delivery Strategies

SOA delivery strategies

- Different strategies exist for how to organize lifecycle stages to enable delivery of specialized service layers
  - Top-down
  - Bottom-up
  - Agile (meet-in-the-middle)

Top-down strategy

- "analysis first" approach
- Promotes the formal definition of corporate business models prior to modeling service boundaries

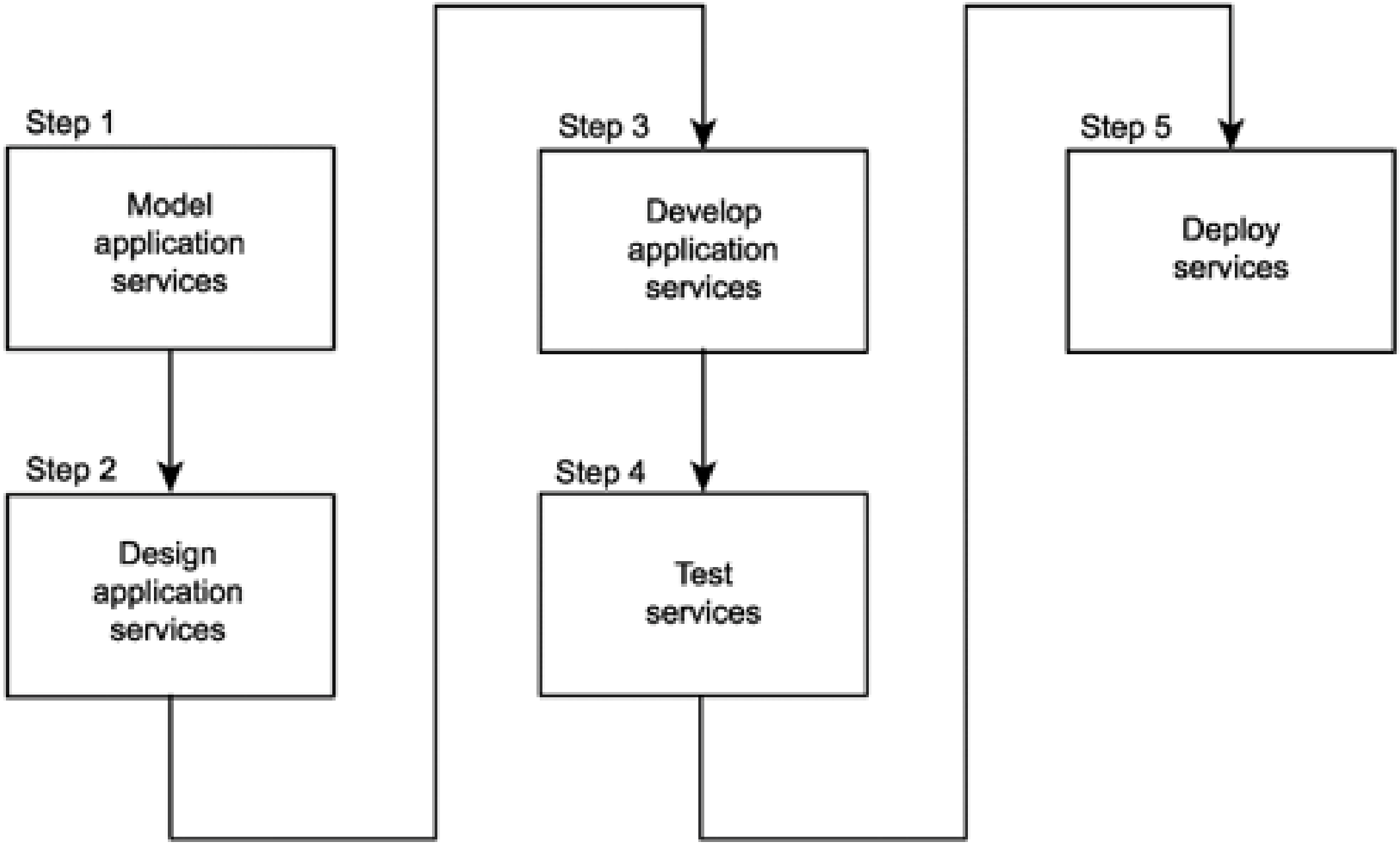

Step 1
Define relevant ontology
Step 2
Align relevant business models
Step 3
Perform service-oriented analysis
Step 4
Perform service-oriented design
Step 5
Develop services
Step 6
Test service operations
Step 7
Deploy services

Top-down strategy benefits and weaknesses

- It can result in highest quality level of SOA
- Significant volume of up-front analysis work

Bottom-up strategy

- This approach encourages the creation of services as a means of fulfilling application-centric requirements.
- Based on the delivery of application services on an “as needed” basis
- Integration is the primary motivator for bottom-up designs, where the need to take advantage of the open SOAP communications framework can be met by simply appending services as wrappers to legacy systems.

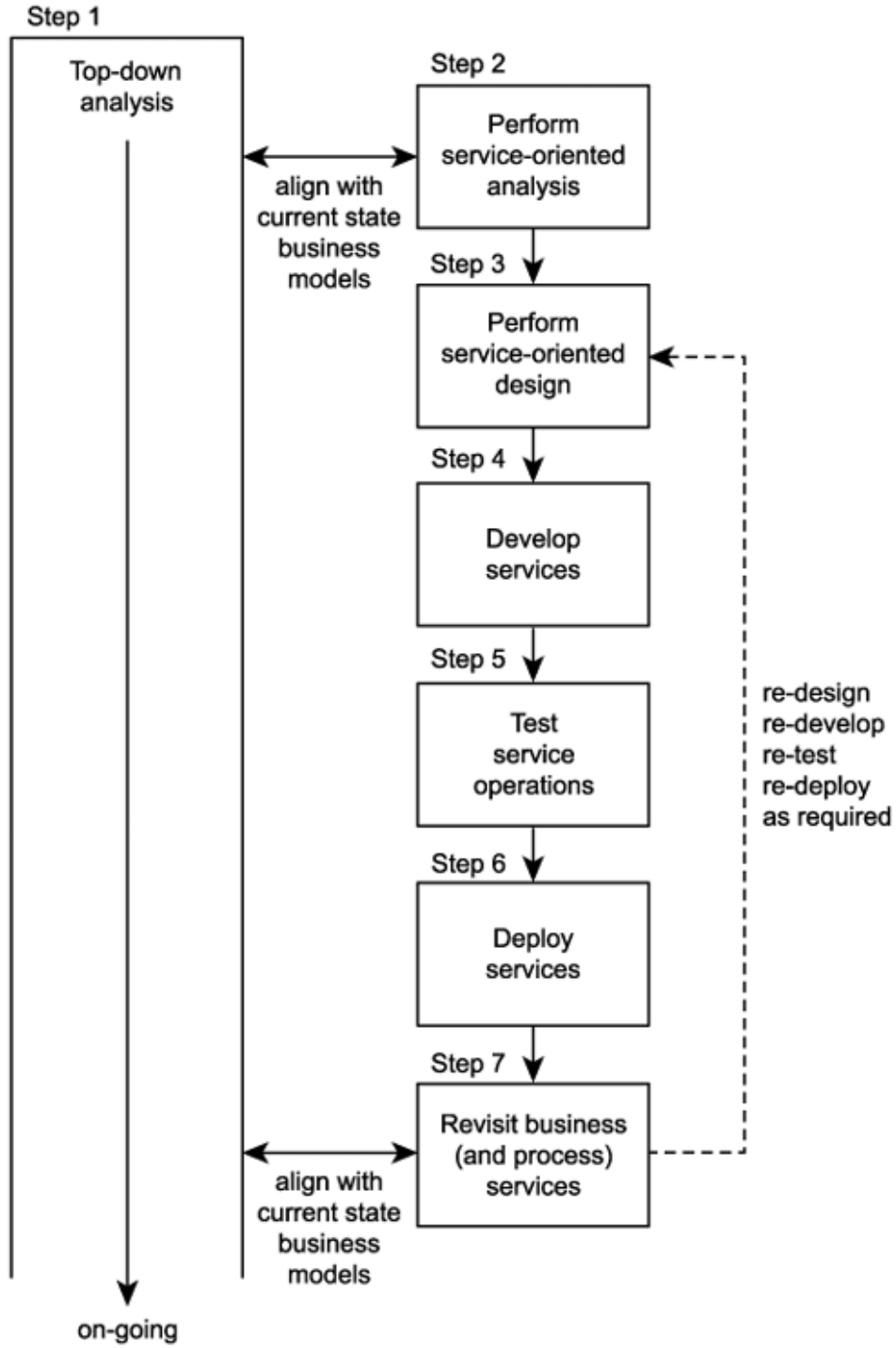

Step 1
Model application services
Step 2
Design application services
Step 3
Develop application services
Step 4
Test services
Step 5
Deploy services

Bottom-up strategy benefits and weaknesses

- Easy to follow
- Does not result in the advancement of service-orientation

Agile strategy

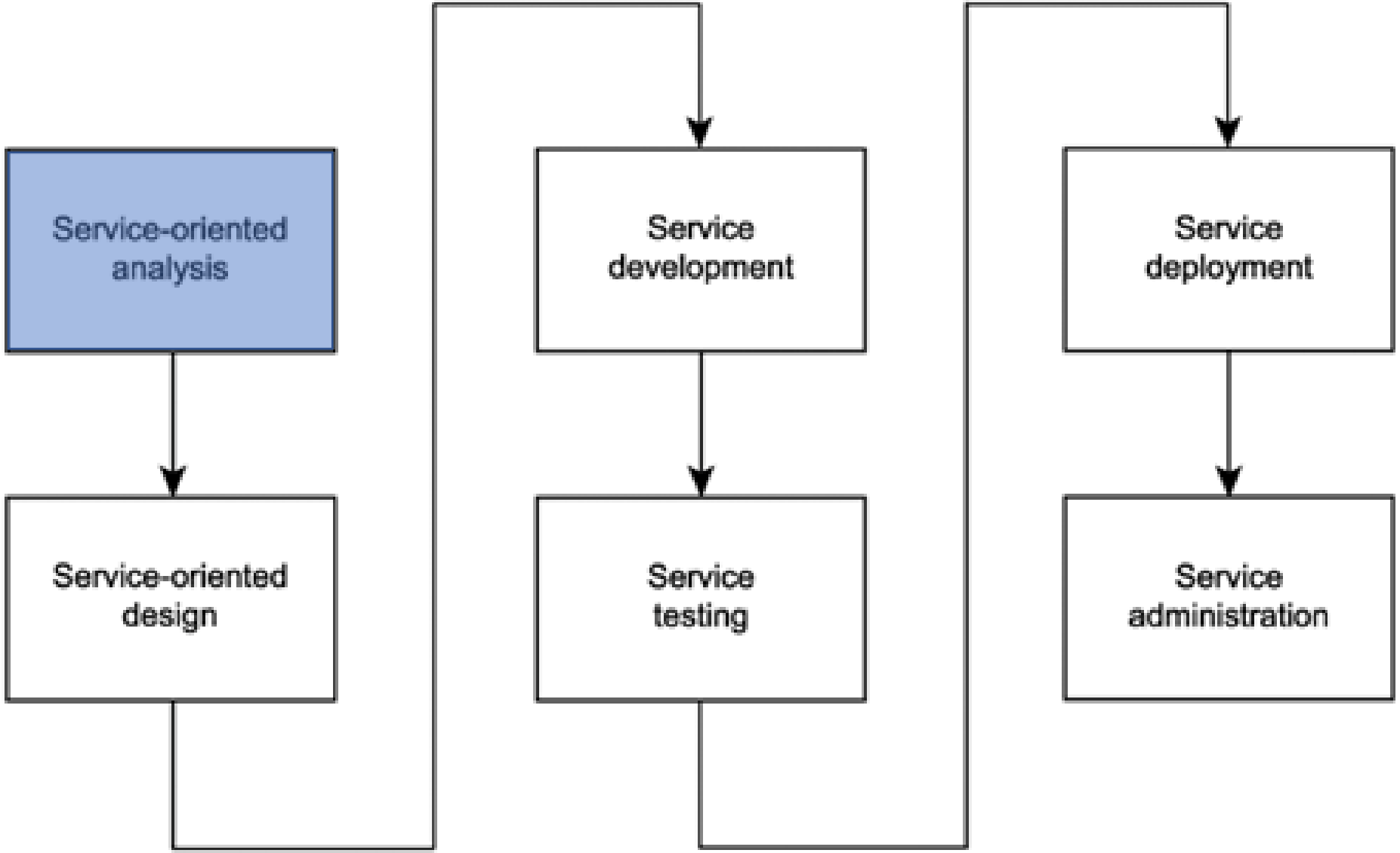
- A combination of top-down and bottom-up approaches
- Business-level analysis occurs concurrently with service design and development


Step 1
Top-down analysis
Step 2
Perform service-oriented analysis
align with current state business models
Step 3
Perform service-oriented design
Step 4
Develop services
Step 5
Test service operations
Step 6
Deploy services
Step 7
Revisit business (and process) services
re-design
re-develop
re-test
re-deploy
as required
align with current state business models
on-going

Agile strategy benefits and weaknesses

- On-going analysis is supported, while allowing immediate delivery of service
- As analysis progresses, existing services are revisited and revised as required
- More complex - it needs to fulfill two opposing sets of requirements
- Increased effort associated with the delivery of every service
  - services may need to be revisited, redesigned, redeveloped, and redeployed

SOA delivery lifecycle



Service-oriented analysis

- The service-oriented analysis phase is probably the most important part of our SOA delivery lifecycle.

Service-oriented analysis results - benefits

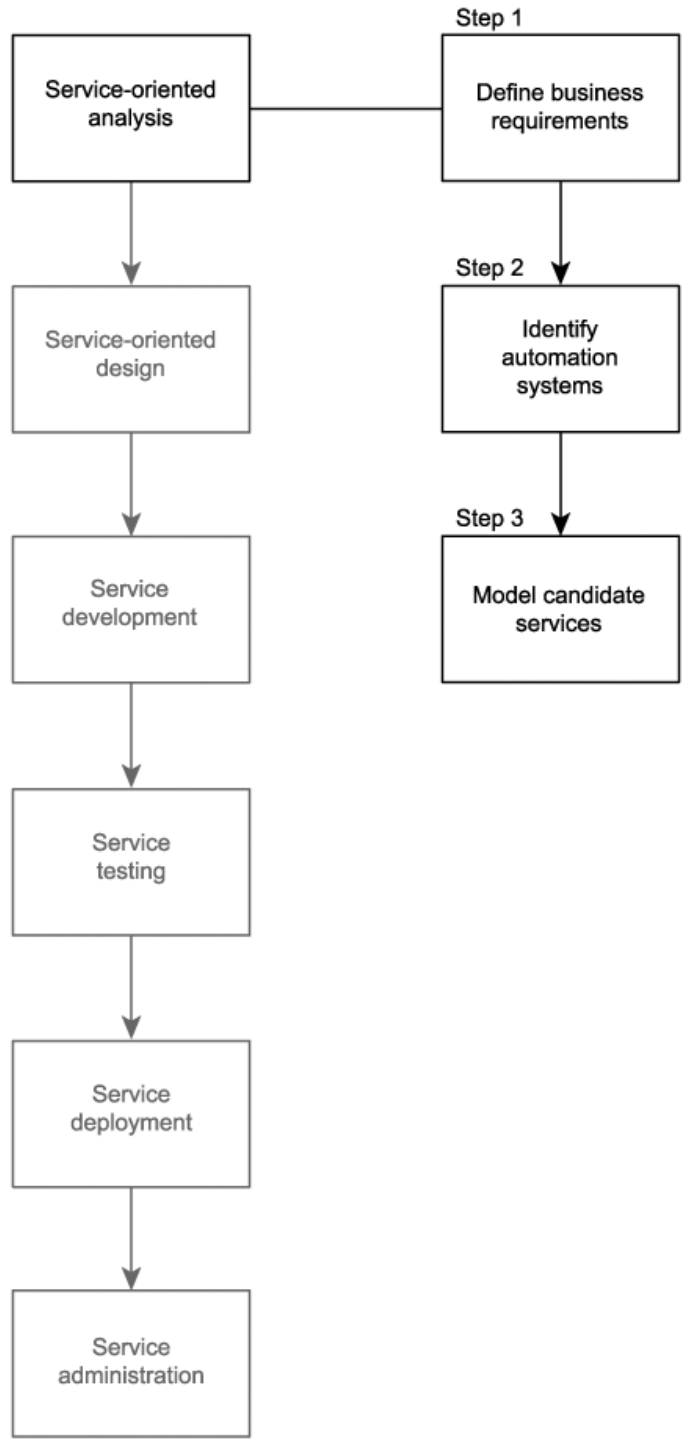
- The benefits that SOA can realize appear to provide the potential to fulfill key business goals of the organization
  - Allow a single set of services to accommodate interaction with different SOA partners, allowing RailCo to pursue new customers without having to build a new set of services each time
    - Services can be modelled in such a manner that they perform common functions for non-specific service requestors
  - Create a legacy systems integration channels that do not need to be removed if underlying legacy technology is replaced
    - Establishing standardized application endpoints
    - The abstraction provided by service layers
    - A separation of business and application logic will allow RailCo to wrap its legacy accounting system in one or more services

Service-oriented analysis objectives

- The primary questions addressed during this phase are:
  - What services need to be built?
  - What logic should be encapsulated by each service?

Service-oriented analysis goals

- Define a preliminary set of service operation candidates.
- Group service operation candidates into logical contexts. These contexts represent service candidates.
- Define preliminary service boundaries so that they do not overlap with any existing or planned services.
- Identify encapsulated logic with reuse potential.
- Ensure that the context of encapsulated logic is appropriate for its intended use.
- Define any known preliminary composition models.


Step 1
Service-oriented analysis
Define business requirements
Step 2
Service-oriented design
Identify automation systems
Step 3
Service development
Model candidate services
Service testing
Service deployment
Service administration

Step 1: Define business automation requirements

- Business requirements documentation is required for this analysis process to begin
- Only requirements related to the scope of a service-oriented solution should be considered
- This business process documentation will be used as the starting point of the service modeling process described in Step 3.

Step 2: Identify existing automation systems

- Gain an understanding of affected legacy environments
- Existing application logic that is already automating any of the requirements identified in Step 1 needs to be identified
- This information will be used to help identify application service candidates during the service modeling process described in Step 3

Step 3: Model candidate services

- Service operation candidates are identified and then grouped into a logical context
- These groups eventually take shape as service candidates
  - they are then further assembled into a tentative composite model representing the combined logic of the planned service-oriented application

Deriving business services

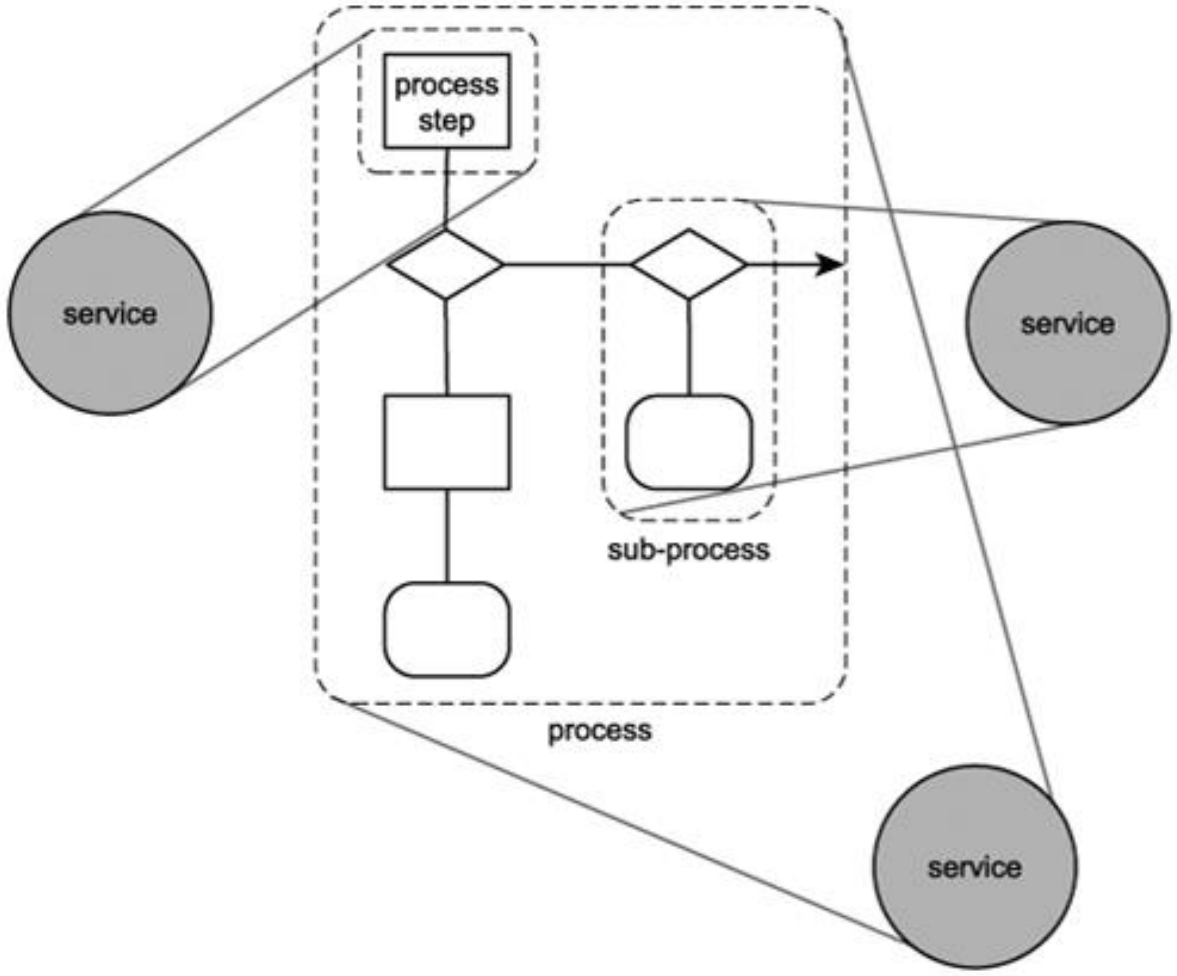
- No standardized means of modeling business services
- Every business model is unique
- Up-front analysis cannot be avoided to properly derive business services that best represent an organization as a cohesive business entity

Common business analysis approaches

- Business services can be derived from
  - Business Process Management (BPM) models
  - Entity models
  - …

Business Process Management (BPM) models

- Business services can be derived from process workflow logic
- Requires a thorough knowledge of the underlying workflow logic



RailCo business processes example (1)

- Invoice Submission Process
  - Accounting clerk creates and issues an electronic invoice using the legacy accounting system.
  - The save event triggers a custom script that exports an electronic copy of the invoice to a network folder.
  - A custom developed component, which polls this folder at ten-minute intervals, picks up the document and transforms it into an XML document.
  - The invoice XML document is then validated. If it is deemed valid, it is forwarded to the Invoice Submission Service. If validation fails, the document is rejected, and the process ends.
  - Depending on when the last metadata check was performed, the service may issue a Get Metadata request to the TLS B2B solution.
  - If the Get Metadata request is issued and if it determines that no changes were made to the relevant TLS service descriptions, the Invoice Submission Service transmits the invoice document to the TLS B2B solution using the ExactlyOnce delivery assurance. If the Get Metadata request identifies a change to the TLS service descriptions, the invoice is not submitted, and the process ends.

RailCo business processes example (2)

- Order Fulfilment Process
  - The RailCo Order Fulfilment Service receives a SOAP message from TLS, containing a payload consisting of a TLS purchase order document.
  - The service validates the incoming document. If valid, the document is passed to a custom component. If the TLS PO fails validation, a rejection notification message is sent to TLS, and the process ends.
  - The component has the XML document transformed into a purchase order that conforms to the accounting system's native document format.
  - The PO then is submitted to the accounting system using its import extension.
  - The PO ends up in the work queue of an accounting clerk who then processes the document.

Entity models

- Primary entities represent the main business documents and transaction areas of an enterprise.
  - For example, Invoice, Purchase Order, Customer, and Claim are all milestone entities within different types of businesses.
- Further, organizations model entities according to proprietary rules and business policies. This results in entities having relationships with each other.

RailCo Entities example (1)

- RailCo has decided to build services centered around specific tasks and is therefore not using entity models as a source for deriving business services. However, to demonstrate the concept of entities, let's briefly look at what entities exist within the RailCo business areas we've been exploring so far.

RailCo Entities example (2)

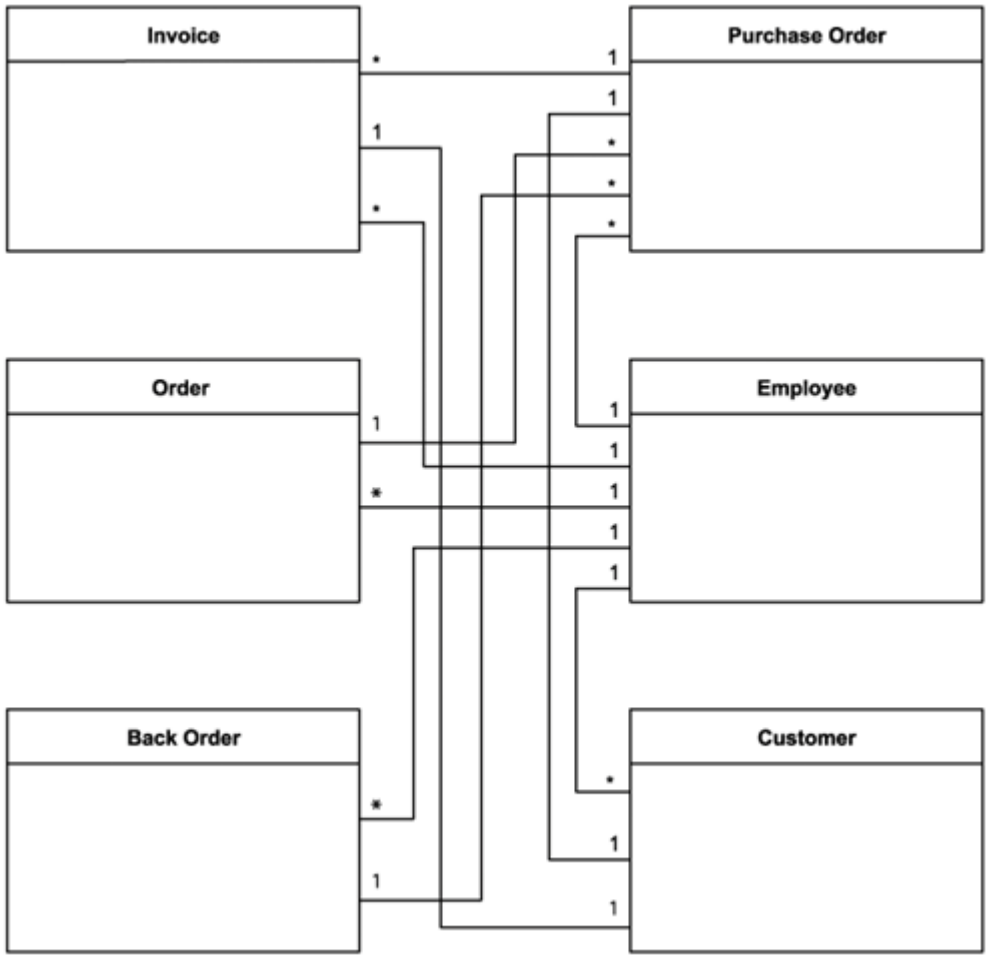
- Based on the business processes we described earlier, RailCo entities of relevance are:
  - Invoice
  - Purchase Order
- Additional entities involved with these processes could include:
  - Employee
  - Order
  - Back Order
  - Customer

RailCo Entities example (3)

- Examples of how the entities described above could relate to each other:
  - A Purchase Order can be related to zero or more Invoices.
  - A Purchase Order can only be related to one Customer.
  - An Order can be related to one or more Purchase Orders.
  - A Back Order can be related to one or more Purchase Orders.
  - An Invoice can only be related to one Customer.
  - An Employee can be related to zero or more Invoices, Purchase Orders, Orders, Back Orders, or Customers.

RailCo Entities example (4)

- Relationships between RailCo entities



Types of derived business services

- Task-centric business services
- Entity-centric business services

Task-centric business services

- These are Web services that have been modeled to accommodate a specific business process. Operations are grouped according to their relevance to the execution of a task in support of a process.
- Typical examples of task-centric business services are:
  - VerifyInvoice
  - GetHistoryReport

Task-centric business services pros and cons

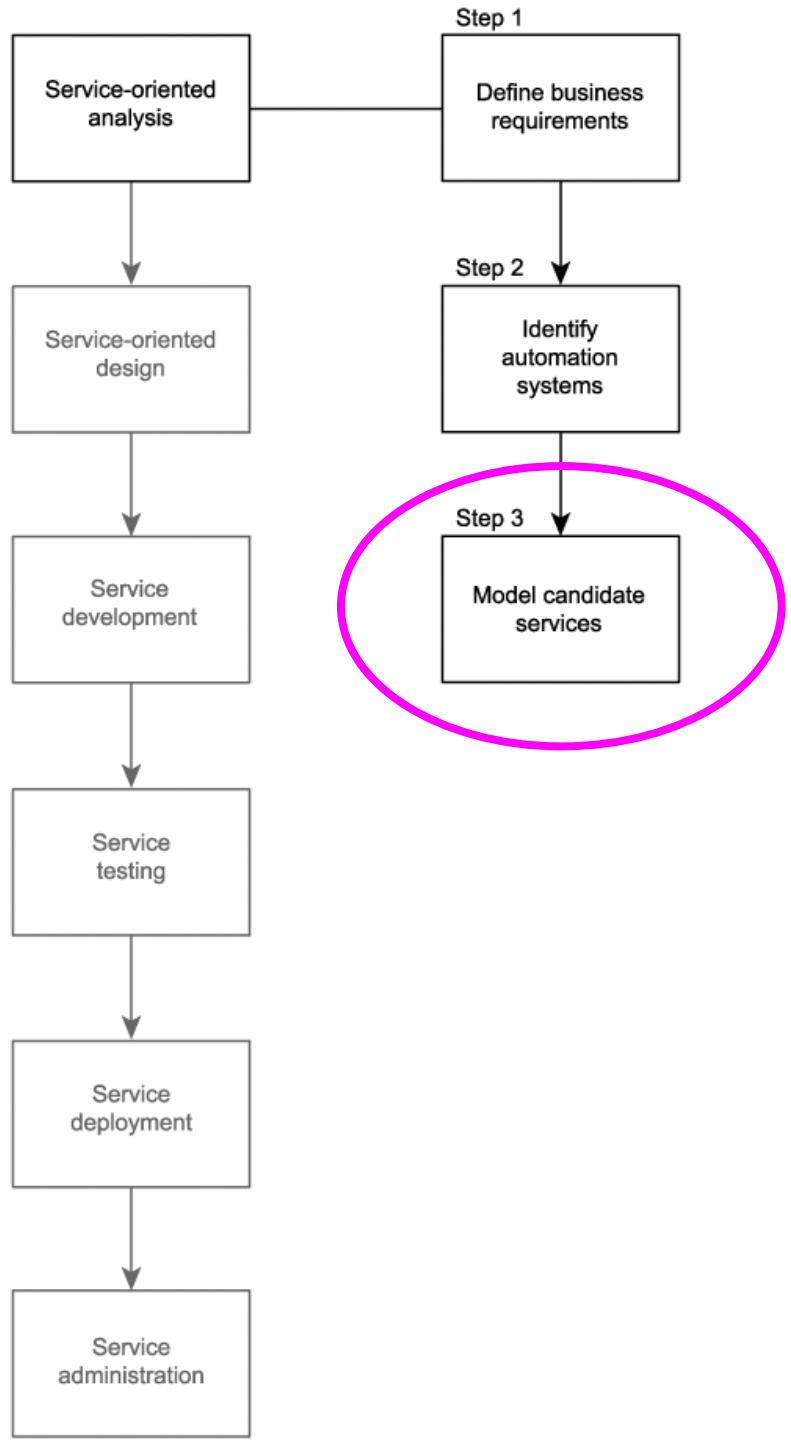
- Require less analysis effort to produce
- Have limited reusability potential
  - Modeling reusable task-centric business services often requires that multiple use-cases and business process models first be analyzed to identify commonality

Entity-centric business services

- Built as part of application development projects centered around a particular business process
- Do not provide an interface specific to that process
- The source of inspiration for these types of services is entity models

Entity-centric business services pros and cons

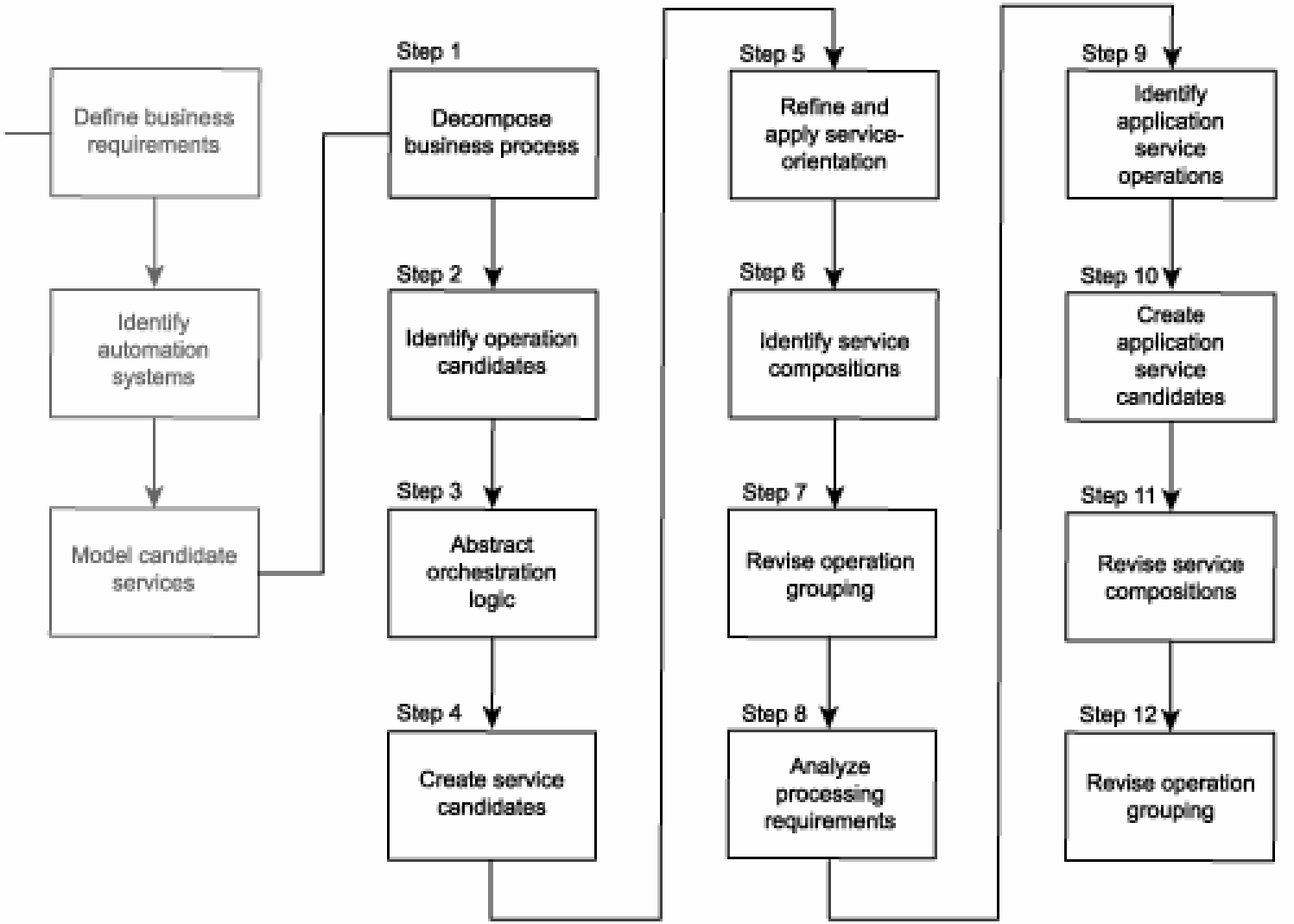
- Highly reusable by numerous business processes
- Require more up-front analysis
- Increased cost of each service and the time required to produce it


Step 1
Service-oriented analysis
Define business requirements
Step 2
Service-oriented design
Identify automation systems
Step 3
Service development
Model candidate services
Service testing
Service deployment
Service administration

Service Modelling

- A service modeling process - organizing the information we gathered in Steps 1 and 2 of the service-oriented analysis process
- This process can be structured in many different ways
- We will provide some guidelines that can be used to customize the process to fit within organization's existing business analysis platforms and procedures

Common steps of modeling process



Step 1: Decompose the business process

- Take the documented business process and break it down into a series of granular process steps

Step 2: Identify business service operation candidates

- Some steps within a business process can be easily identified as not belonging to the potential logic that should be encapsulated by a service candidate
  - Manual process steps that cannot or should not be automated
  - Process steps performed by existing legacy logic for which service candidate encapsulation is not an option
- By filtering out these parts we are left with the processing steps most relevant to our service modeling process.

Step 3: Abstract orchestration logic

- If the decision was made to not incorporate an orchestration service layer - then skip this step
- If it was decided to build an orchestration layer as part of the SOA, then we should identify the parts of the processing logic that this layer would abstract
- Potential types of logic suitable for this layer include:
  - business rules
  - conditional logic
  - exception logic
  - sequence logic

Step 4: Create business service candidates

- Review the processing steps that remain after we completed step 2 and determine one or more logical contexts with which these steps can be grouped.
- Each context represents a service candidate.
- In this step, we can also add additional service operation candidates not required by the current business process, but added to round out entity services with a complete set of reusable operations.

Step 5: Refine and apply principles of service-orientation

- So far we have just grouped processing steps derived from an existing business process
- To make our service candidates truly worthy of an SOA, we must take a closer look at the underlying logic of each proposed service operation candidate
- In this step we will make adjustments by applying key service-orientation principles
  - reusability
  - autonomy

Step 6: Identify candidate service compositions

- Identify a set of the most common scenarios that can take place within the boundaries of the business process.
- For each scenario, follow the required processing steps as they exist now.
- This exercise accomplishes the following:
  - It gives you a good idea as to how appropriate the grouping of your process steps is.
  - It demonstrates the potential relationship between orchestration and business service layers.
  - It identifies potential service compositions.
  - It highlights any missing workflow logic or processing steps.
- Ensure that as part of your chosen scenarios you include failure conditions that involve exception handling logic. Note also that any service layers you establish at this point are still preliminary and still subject to revisions during the design process

Step 7: Revise business service operation grouping

- Based on the results of the composition exercise in Step 6, revisit the grouping of your business process steps and revise the organization of service operation candidates as necessary.
- It is not unusual to consolidate or create new groups (service candidates) at this point.

Step 8: Analyze application processing requirements

- Optional and more suited for complex business processes and larger service-oriented environments
- Requires to more closely study the underlying processing requirements of all service candidates to abstract any further technology-centric service candidates from this view that will complete a preliminary application services layer.
- To accomplish this, each processing step identified so far is required to undergo a mini-analysis.
- Specifically, what needs to determined is:
  - What underlying application logic needs to be executed to process the action described by the operation candidate.
  - Whether the required application logic already exists or whether it needs to be newly developed.
  - Whether the required application logic spans application boundaries. In other words, is more than one system required to complete this action?

Step 9: Identify application service operation candidates

- Break down each application logic processing requirement into a series of steps.
- Be explicit about how you label these steps so that they reference the function they are performing.

Step 10: Create application service candidates

- Group these processing steps according to a predefined context.
- With application service candidates, the primary context is a logical relationship between operation candidates.
- This relationship can be based on any number of factors, including:
  - association with a specific legacy system
  - association with one or more solution components
  - logical grouping according to type of function

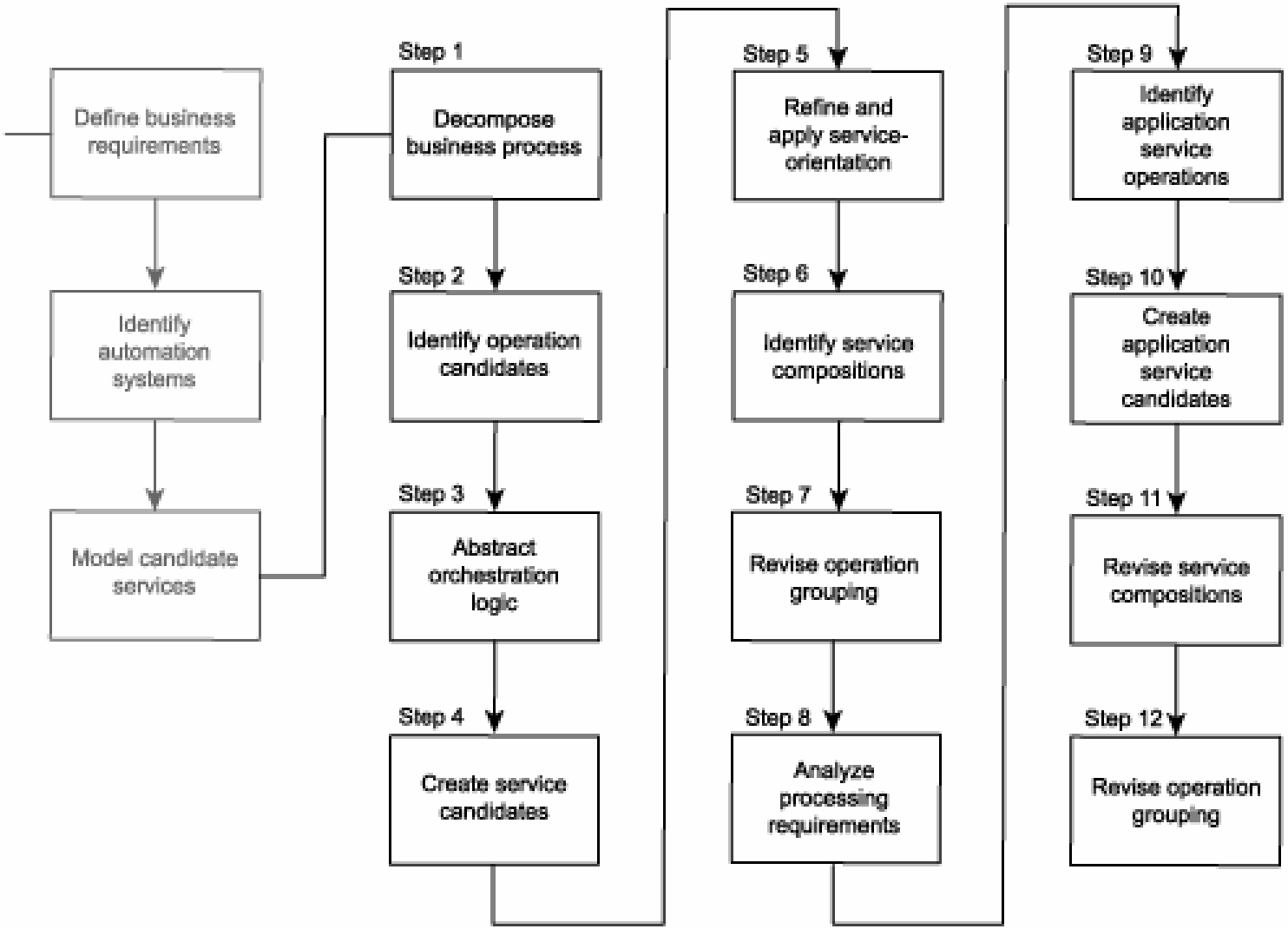
Step 11: Revise candidate service compositions

- Revisit the original scenarios identified in Step 5 and run through them again.
- Incorporate the new application service candidates as well.
- This will result in the mapping of elaborate activities that bring to life expanded service compositions.

Step 12: Revise application service operation grouping

- This step is optional

Common steps of modeling process



SOA delivery lifecycle