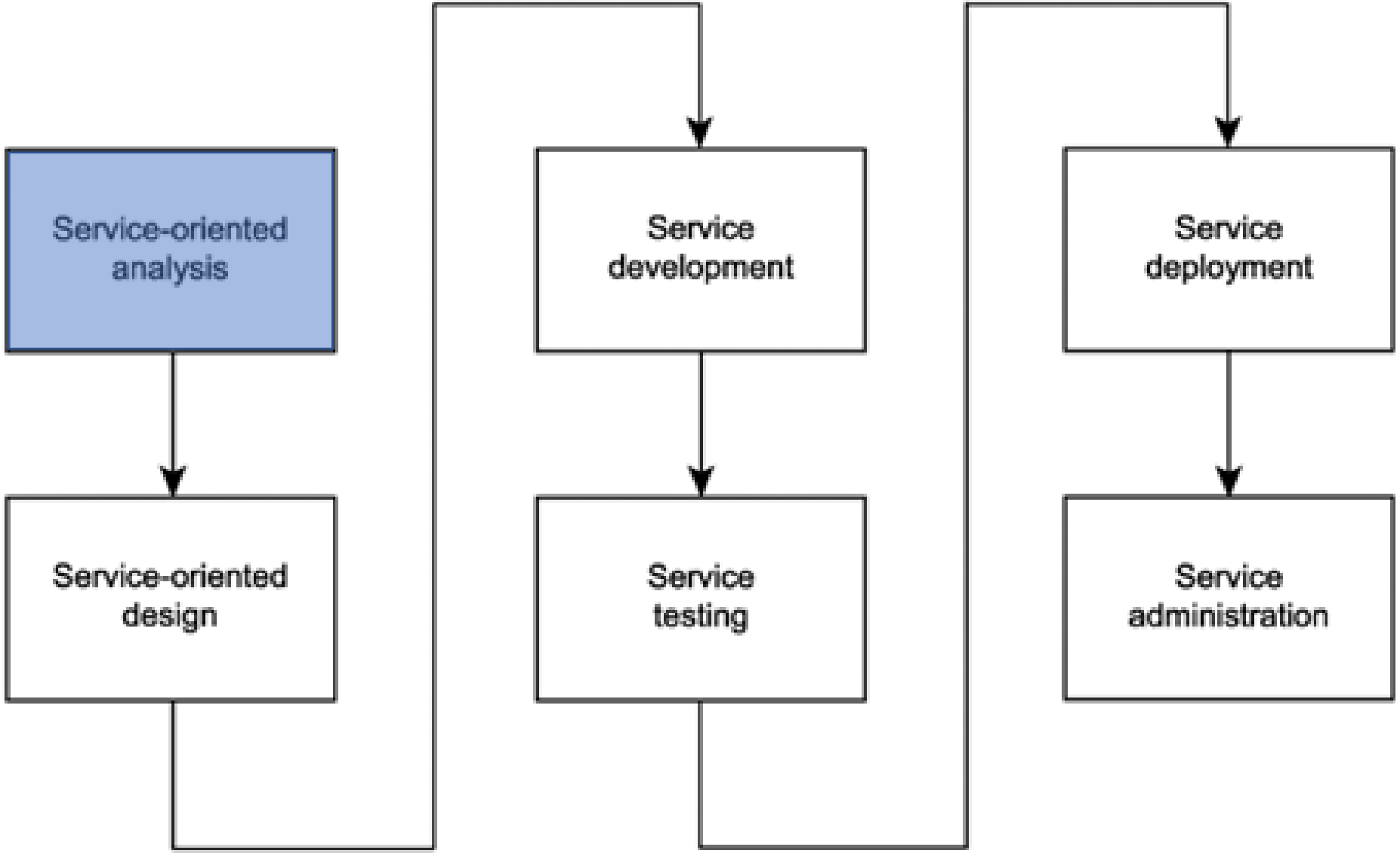
Reminder
Hot topic study – task 3 – individual/group

- Create a home page in your groups wiki area devoted to your selected topic – give an overview of your topic
- Deadline: 28th April (Thursday)

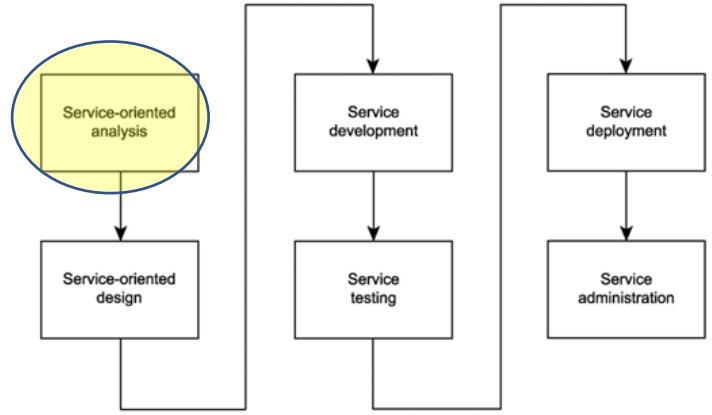
Module Six: Service Engineering

- Last time we studied the SOA delivery lifecycle and we performed some of the steps in the SOA delivery lifecycle for RailCo company

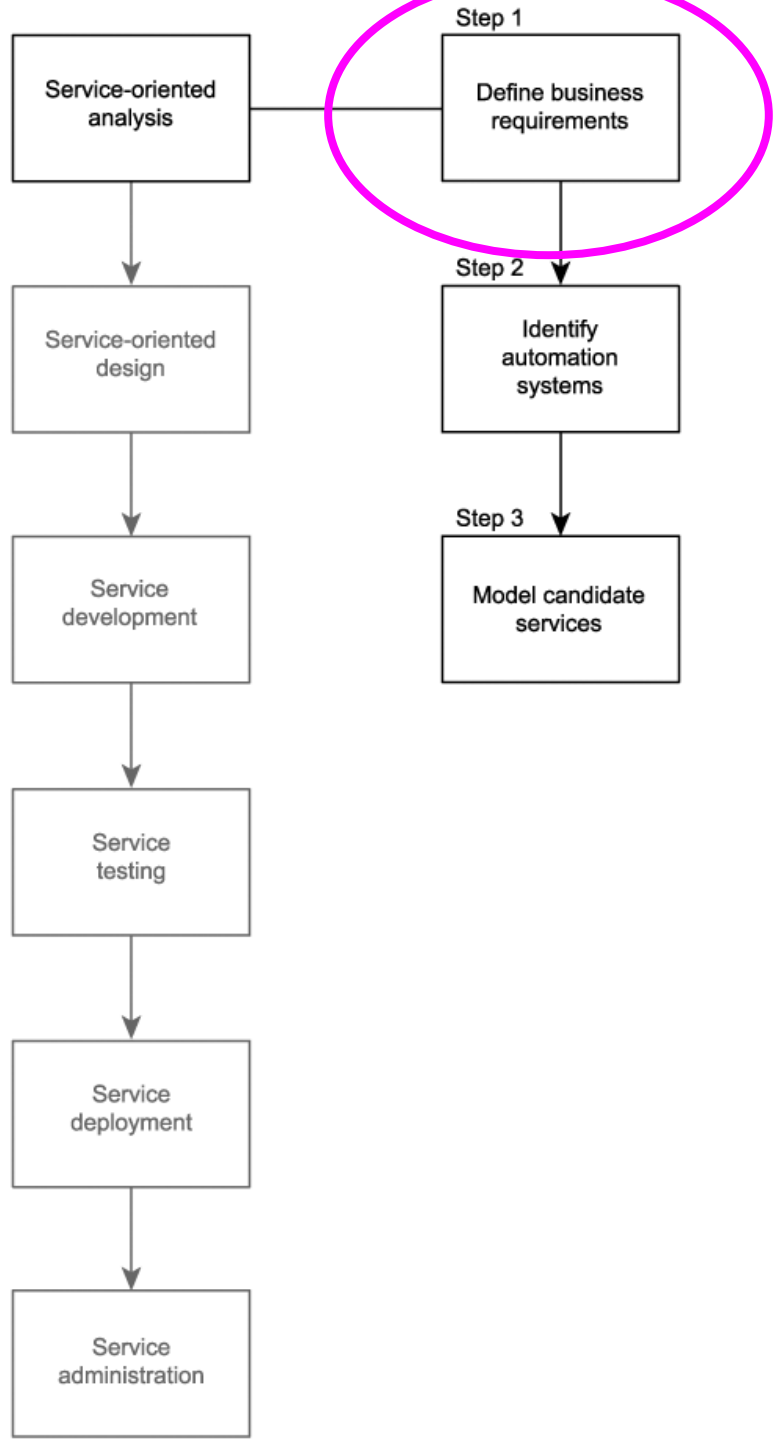
SOA delivery lifecycle



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

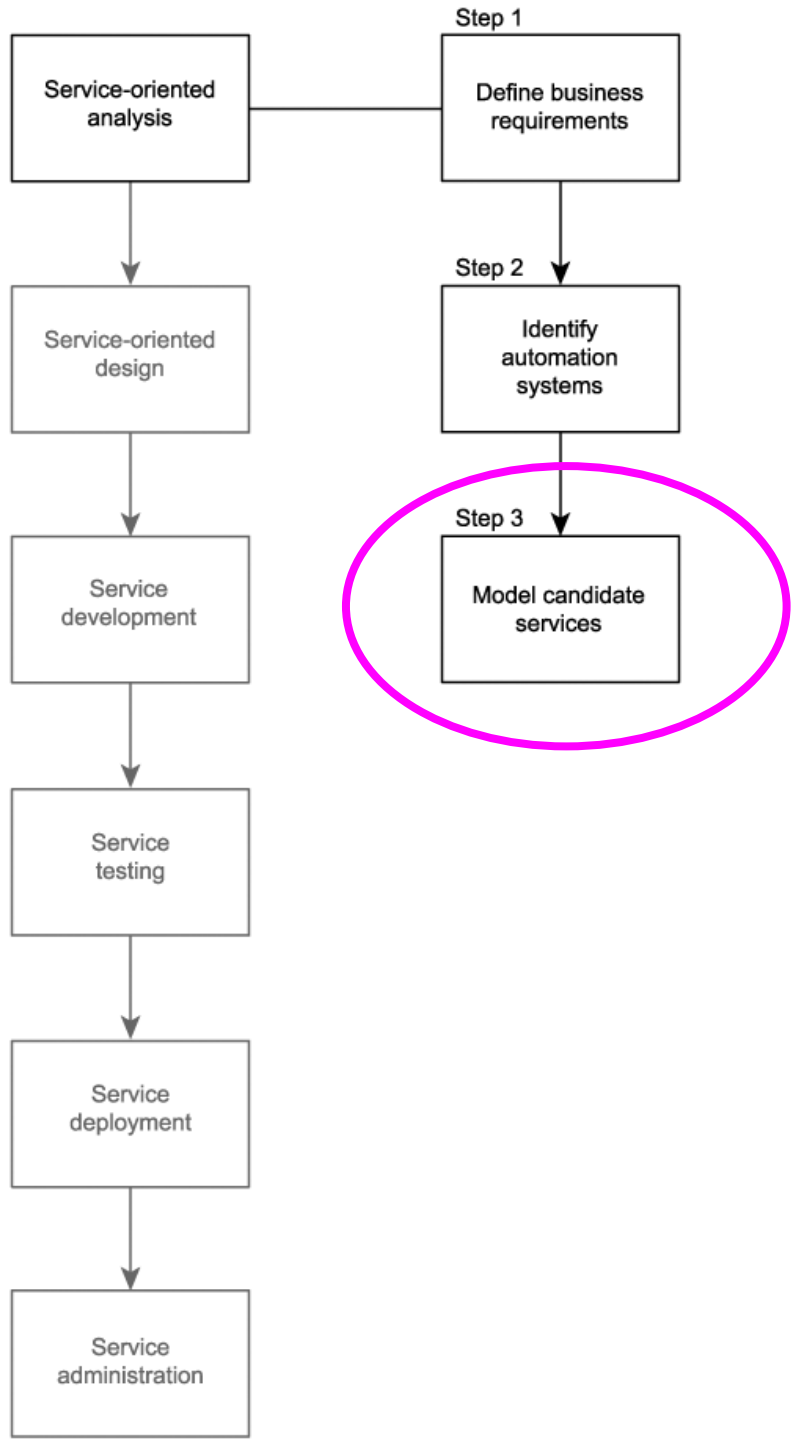

Step 1
Service-oriented analysis
Define business requirements
Step 2
Service-oriented design
Identify automation systems
Step 3
Service development
Model candidate services
Service testing
Service deployment
Service administration

Invoice Submission business process

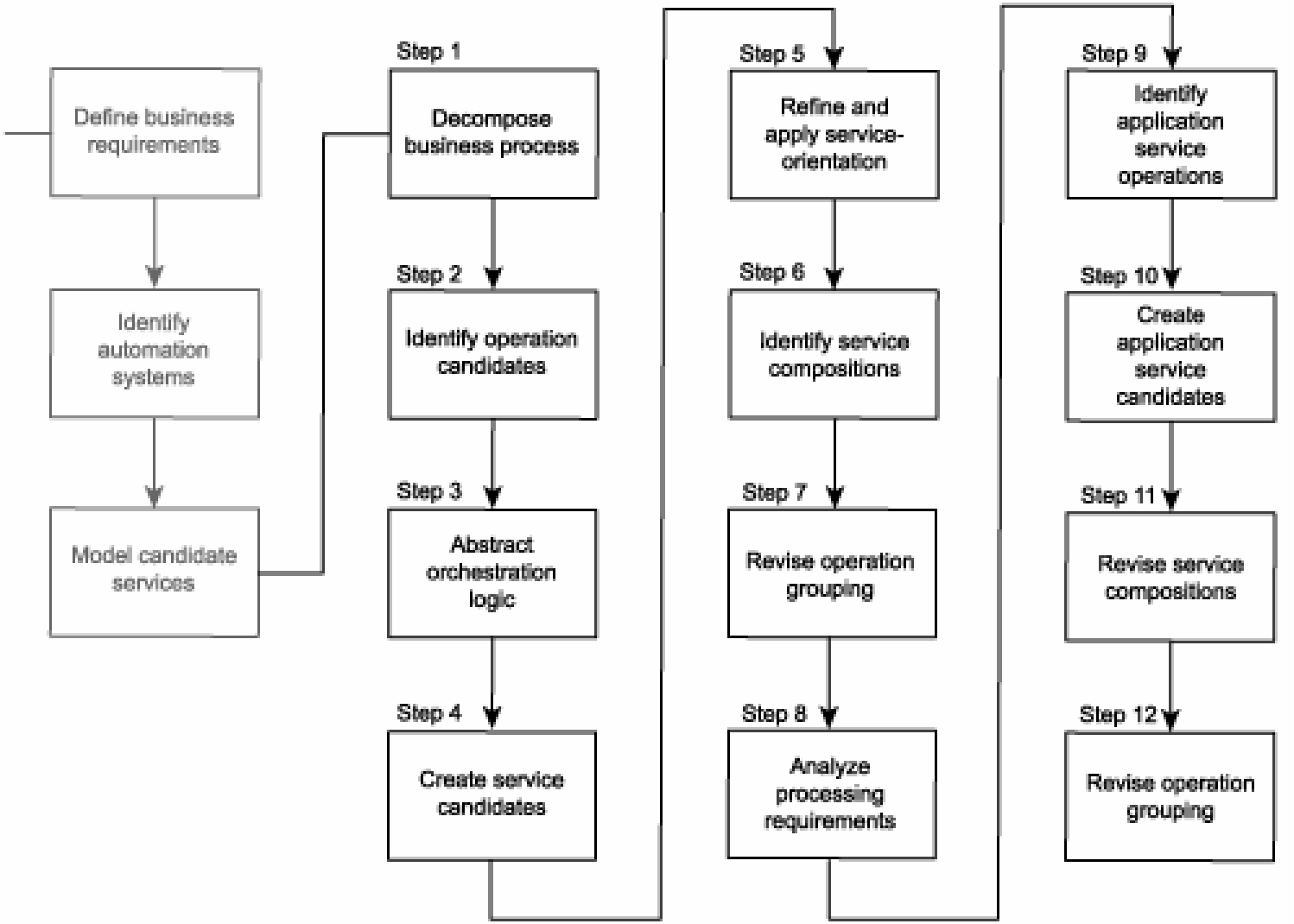
- Accounting clerk creates and issues an electronic invoice using the legacy accounting system.
- The save event triggers a custom script that exports an electronic copy of the invoice to a network folder.
- A custom developed component, which polls this folder at ten-minute intervals, picks up the document 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.

Order Fulfilment Business process

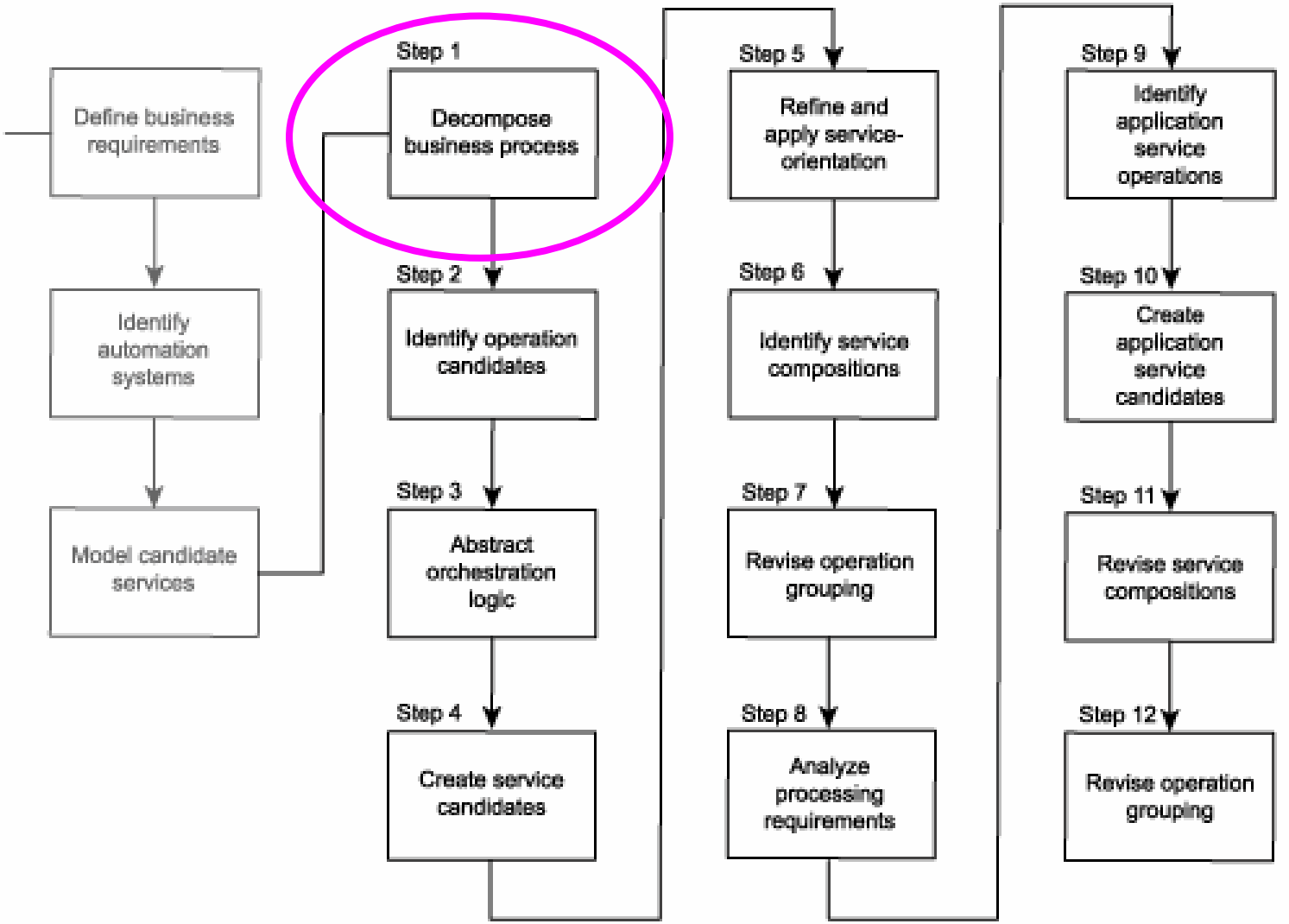
- The RailCo Order Fulfilment Service receives a SOAP message from TLS, containing a payload consisting of a TLS purchase order document.
- The service validates the incoming document. If valid, the document is passed to a custom component. If the TLS PO fails validation, a rejection notification message 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.


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

Common steps of modeling process



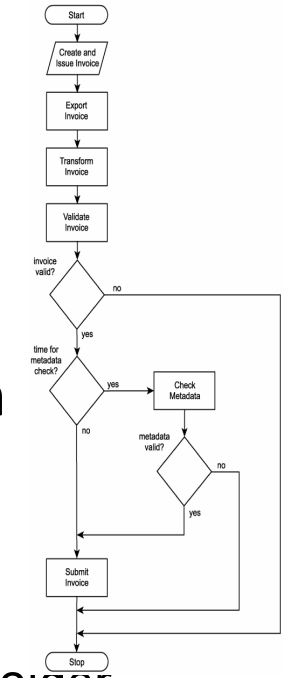
Common steps of modeling process



From business process to decomposed business process

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.



Decomposed Invoice Submission Process

- Create electronic invoice.
- Issue electronic invoice.
- Export electronic invoice to network folder.
- Poll network folder.
- Retrieve electronic invoice.
- Transform electronic invoice to XML document.
- Check validity of invoice document. If invalid, end process.
- Check if it is time to verify TLS metadata.
- If required, perform metadata check. If metadata check fails, end process.

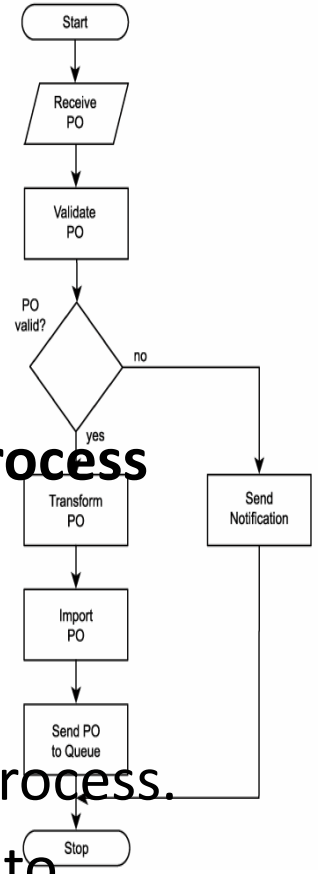
From business process to decomposed business process

Order Fulfilment Process

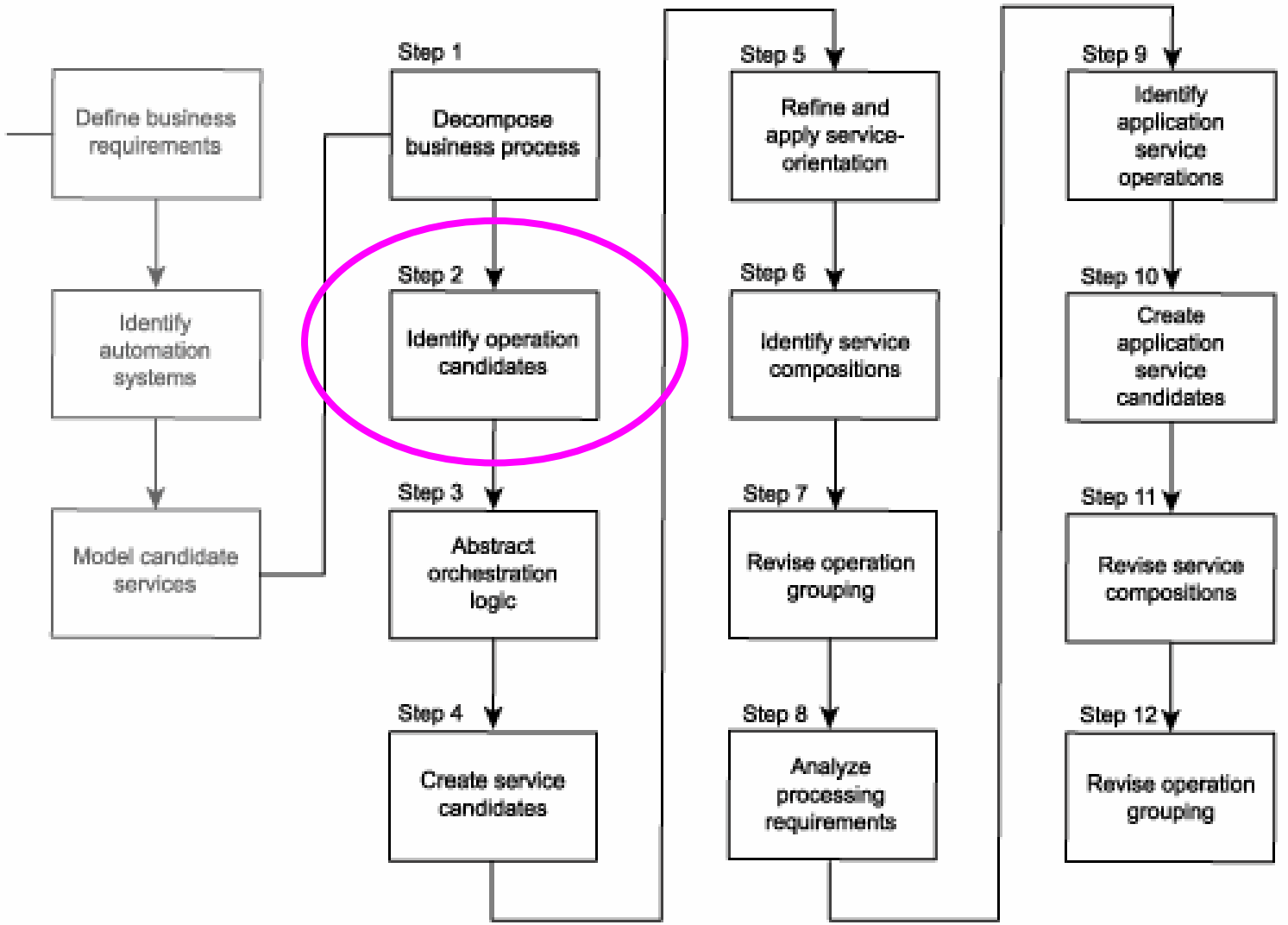
- The RailCo Order Fulfilment Service receives a SOAP message from TLS, containing a payload consisting of a TLS purchase order document.
- The service validates the incoming document. If valid, the document is passed to a custom component. If the TLS PO fails validation, a rejection notification message 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.

Decomposed Order Fulfilment Process

- Receive PO document.
- Validate PO document.
- If PO document is invalid, send rejection notification and end process.
- Transform PO XML document into native electronic PO format.
- Import electronic PO into accounting system.
- Send PO to accounting clerk's work queue.



Common steps of modeling process



From decomposed business process to service operation candidates

Decomposed Invoice Submission Process

- Create electronic invoice.
- Issue electronic invoice.
- Export electronic invoice to network folder.
- Poll network folder.
- Retrieve electronic invoice.
- Transform electronic invoice to XML document.
- Check validity of invoice document. If invalid, end process.
- Check if it is time to verify TLS metadata.
- If required, perform metadata check. If metadata check fails, end process.

Invoice Submission operation candidates:

- ~~Create electronic invoice.~~ (A manual step performed by the accounting clerk.)
- ~~Issue electronic invoice.~~ (A manual step performed by the accounting clerk.)
- Export electronic invoice to network folder. (Currently a custom developed extension of the legacy system. Could be made part of a generic service candidate.)
- Poll network folder. (Currently performed by a custom developed component. Could be made part of a service candidate.)
- Retrieve electronic invoice. (Same as previous.)
- Transform electronic invoice to XML document. (Same as previous.)
- Check validity of invoice document. If invalid, end process. (Is currently being performed as part of the Invoice Submission Service's parsing routine. No foreseeable need to change this.)
- Check if it is time to verify TLS metadata. (Is currently being performed as part of the Invoice Submission Service's parsing routine. Looks like a potentially reusable operation candidate. Could be moved to a separate service candidate.)
- If required, perform metadata check. If metadata check fails, end process. (Same as previous.)

From decomposed business process to service operation candidates

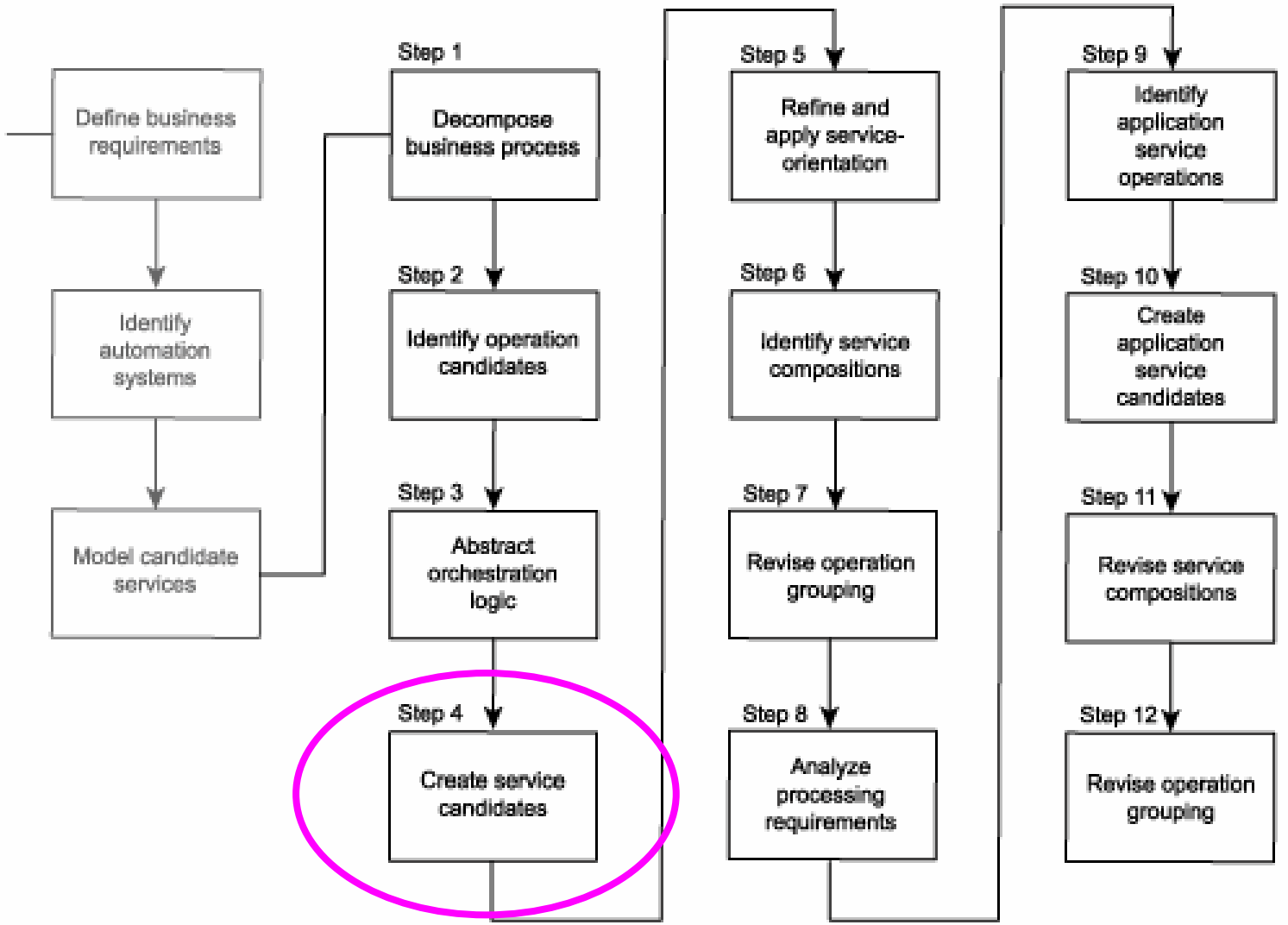
Decomposed Order Fulfilment Process

- Receive PO document.
- Validate PO document.
- If PO document is invalid, send rejection notification and end process.
- Transform PO XML document into native electronic PO format.
- Import electronic PO into accounting system.
- Send PO to accounting clerk's work queue.

Order Fulfillment operation candidates

- Receive PO document. (Is currently being performed by the Order Fulfillment Service. No foreseeable need to change this.)
- Validate PO document. (Same as previous.)
- If PO document is invalid, send rejection notification and end process. (Same as previous.)
- Transform PO XML document into native electronic PO format. (Currently performed by a custom developed component. Could be made part of a service candidate.)
- Import electronic PO into accounting system. (Currently a custom developed extension of the legacy system. Could be made part of a generic service candidate.)
- Send PO to accounting clerk's work queue. (Same as previous.)

Common steps of modeling process



From service operation candidates to service candidates

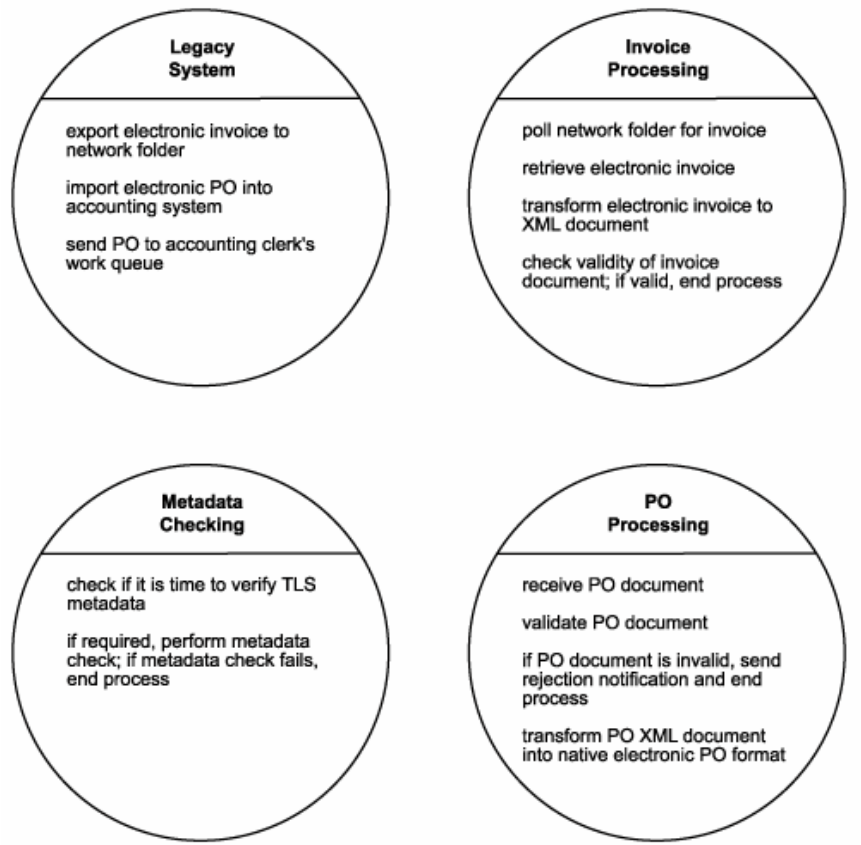
Invoice Submission operation candidates:

- ~~Create electronic invoice.~~ (A manual step performed by the accounting clerk.)
- ~~Issue electronic invoice.~~ (A manual step performed by the accounting clerk.)
- Export electronic invoice to network folder. (Currently a custom developed extension of the legacy system. Could be made part of a generic service candidate.)
- Poll network folder. (Currently performed by a custom developed component. Could be made part of a service candidate.)
- Retrieve electronic invoice. (Same as previous.)
- Transform electronic invoice to XML document. (Same as previous.)
- Check validity of invoice document. If invalid, end process. (Is currently being performed as part of the Invoice Submission Service's parsing routine. No foreseeable need to change this.)
- Check if it is time to verify TLS metadata. (Is currently being performed as part of the Invoice Submission Service's parsing routine. Looks like a potentially reusable operation candidate. Could be moved to a separate service candidate.)
- If required, perform metadata check. If metadata check fails, end process. (Same as previous.)

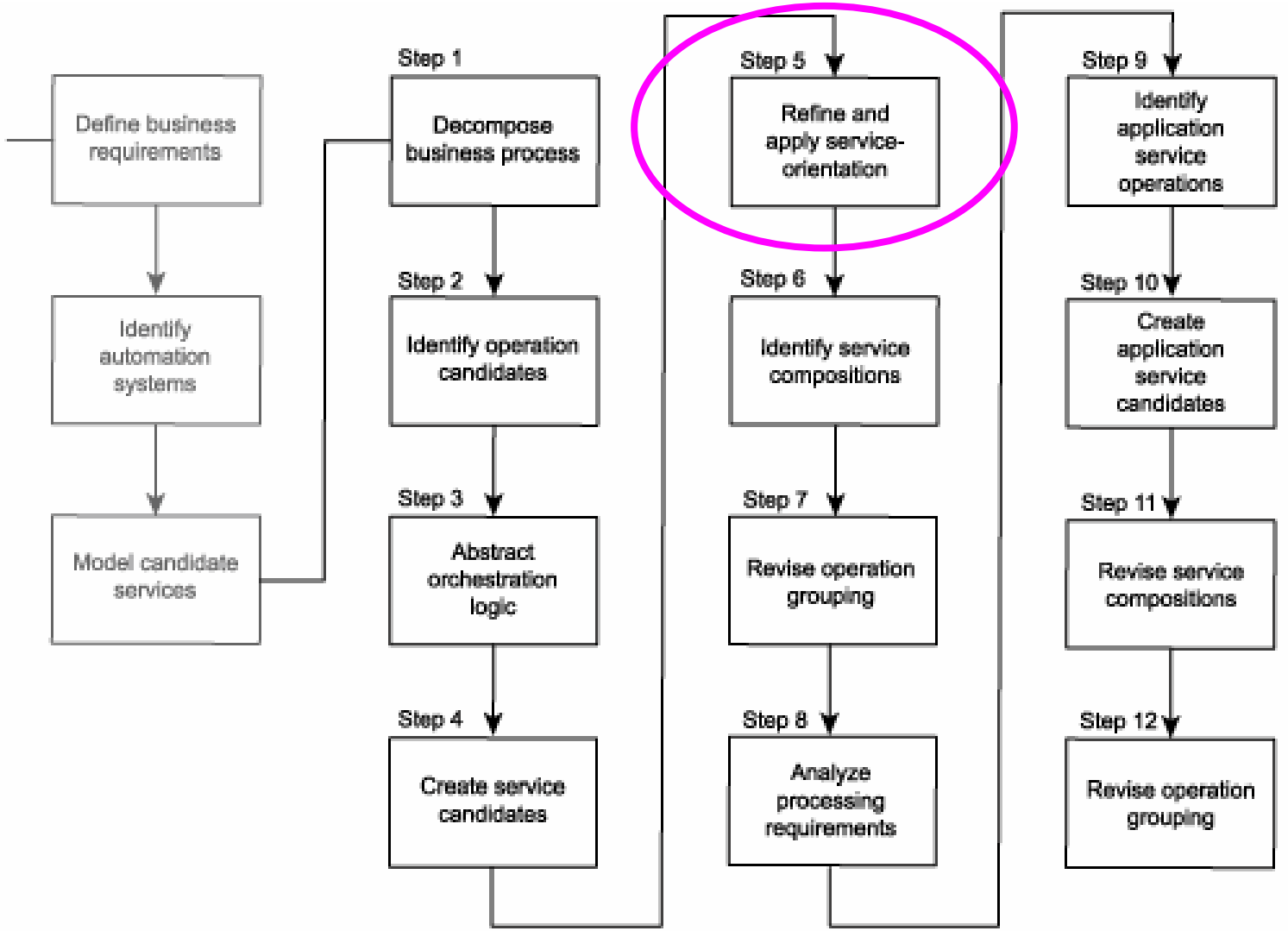
Order Fulfillment operation candidates:

- Receive PO document. (Is currently being performed by the Order Fulfillment Service. No foreseeable need to change this.)
- Validate PO document. (Same as previous.)
- If PO document is invalid, send rejection notification and end process. (Same as previous.)
- Transform PO XML document into native electronic PO format. (Currently performed by a custom developed component. Could be made part of a service candidate.)
- Import electronic PO into accounting system. (Currently a custom developed extension of the legacy system. Could be made part of a generic service candidate.)
- Send PO to accounting clerk's work queue. (Same as previous.)

service candidates



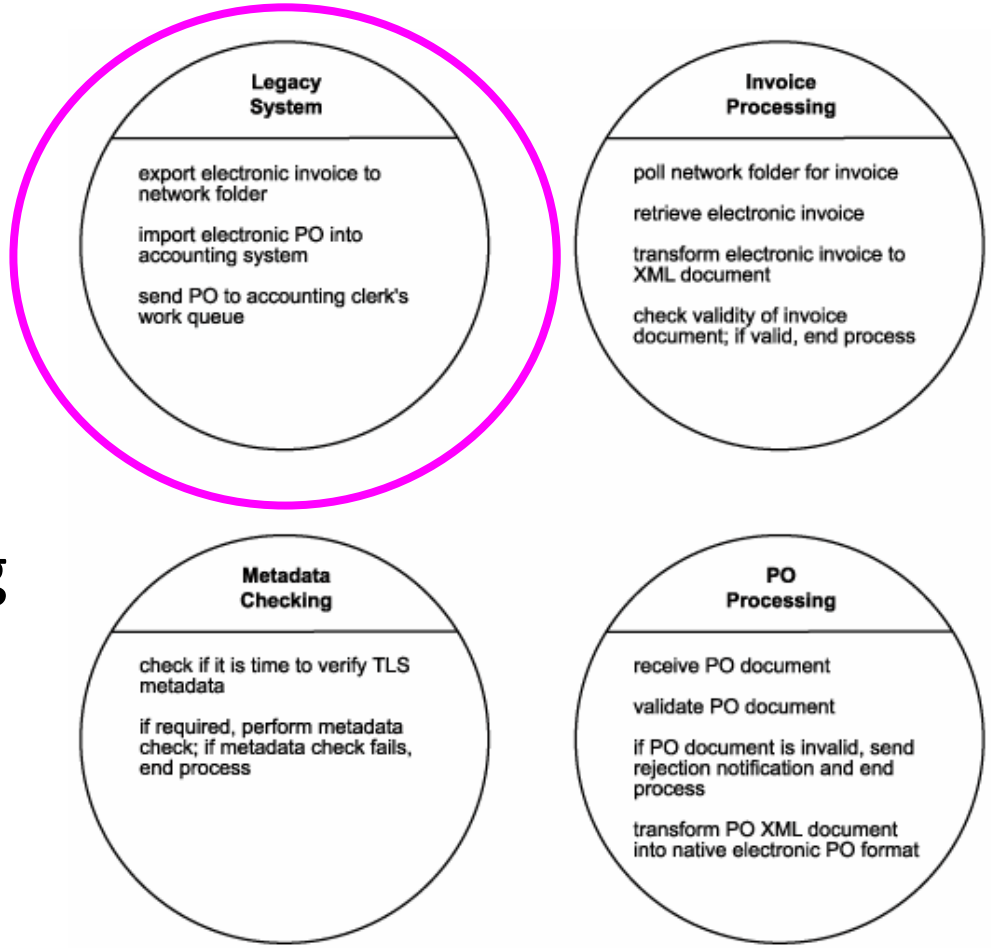
Common steps of modeling process



- In step 5, we studied our initially proposed design and refined it
- We will now review the adjustments we made in 2 of our proposed service candidates
  - Legacy System Service
  - Invoice Processing
- While reviewing, please think about these questions:
  - What kind of adjustments were made?
  - How do we refine the designed services?

Adjustments to the Legacy System Service (1)

- Legacy System Service has at the moment 3 service operation candidates
  - Export electronic invoice to network folder.
  - Import electronic PO into accounting system.
  - Send PO to accounting clerk's work queue.

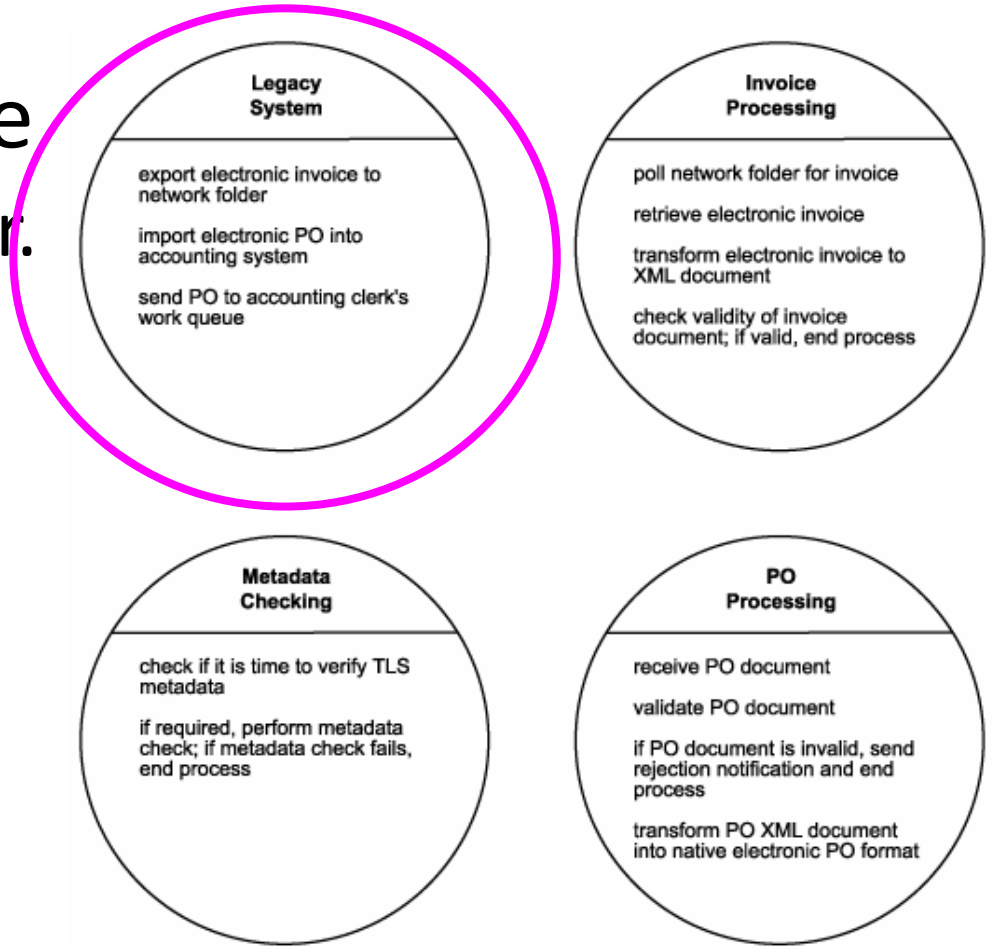


Adjustments to the Legacy System Service (2)

- We make this operation more reusable by designing it in more generic way
  - Export document to network folder.
  - ~~Export electronic invoice to network folder.~~
- We decide to combine these two steps into one (the second step is directly dependent on the first step)
  - Import and forward document to work queue
  - ~~Import electronic PO into accounting system.~~
  - ~~Send PO to accounting clerk's work queue.~~

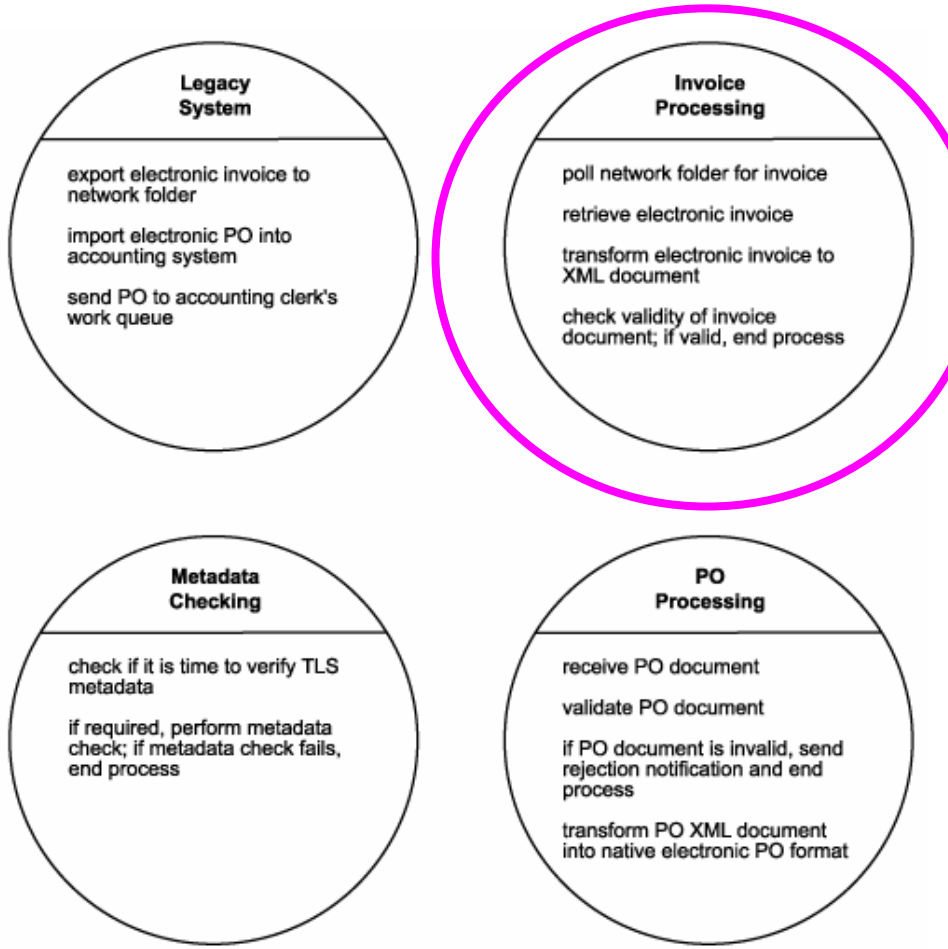
Adjustments to the Legacy System Service (3)

- The revised Legacy System Service
  - Export document to network folder.
  - Import and forward document to work queue



Adjustments to the Invoice Processing Service (1)

- Invoice Processing Service has at the moment 4 service operation candidates
  - Poll network folder for invoice.
  - Retrieve electronic invoice.
  - Transform electronic invoice to XML document.
  - Check validity of invoice document. If invalid, end process.

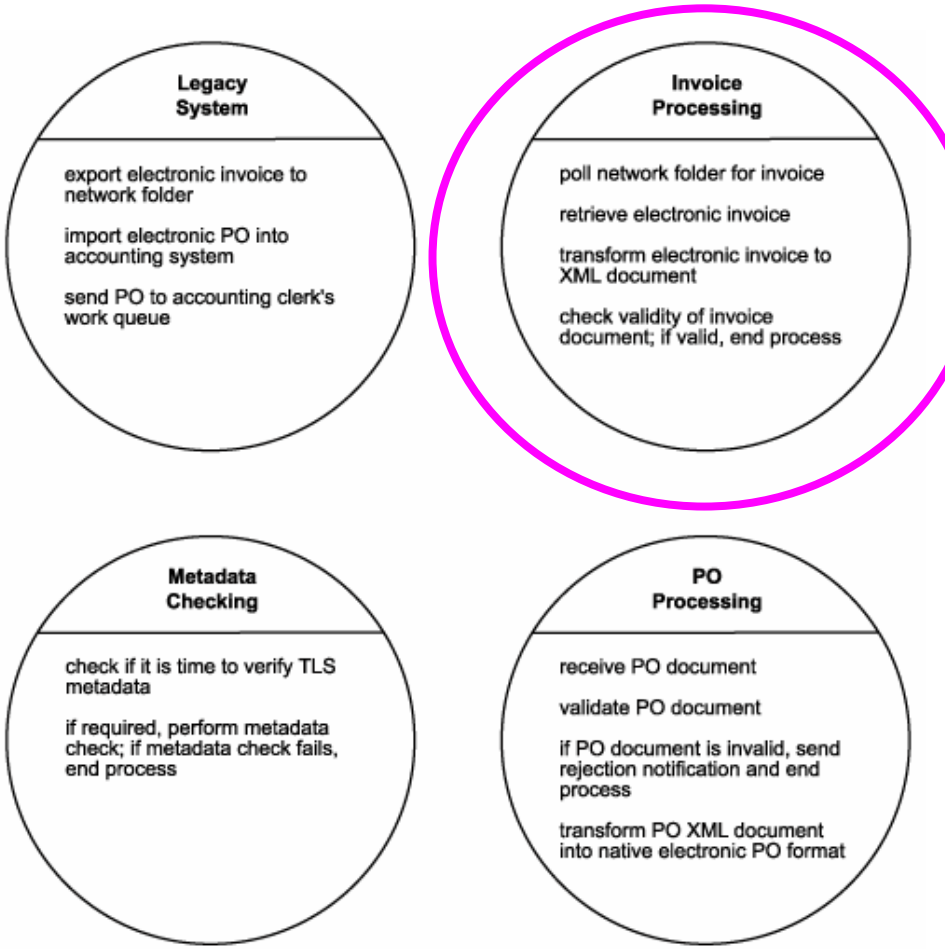


Adjustments to Invoice Processing Service (2)

- While adjusting this service, we not only changed some of the service operation candidates, but also decided to create another, new service candidate
  - Invoice Processing Service
    - ~~Poll network folder for invoice~~. -→ ~~poll folder for new documents~~
    - ~~Retrieve electronic invoice.~~
    - ~~Transform electronic invoice to XML document.~~
    - ~~Check validity of invoice document. If invalid, end process.~~
    - Retrieve and transform invoice document.
  - Polling Notification Service:
    - Poll folder for new documents.
    - If documents arrive for which there are subscribers, issue notifications.

Adjustments to the Invoice Processing Service (1)

- The revised Invoice Processing Service
  - Retrieve and transform invoice document.
- Newly created Polling Notification Service
  - Poll folder for new documents.
  - If documents arrive for which there are subscribers, issue notifications.



- When refining the design of the services, what kinds of adjustments were made?
  - Some service operations candidates were deleted
  - Two service operations were combined into one
  - Some service operations candidates changed their name
  - New service candidates were created
  - New service operation candidates were created
  - Service operation candidates were moved from one service candidate to another
  - Some service operations candidates remained the same
  - …

Multiple Choice(multiple)

Points: 1

设置

When refining the design of the services, what kinds of adjustments were made? (choose all answers that you think are correct)

A Some service operations candidates were deleted

B Two service operations were combined into one

C Some service operations candidates changed their name

D New service candidates were created

E New service operation candidates were created

F Service operation candidates were moved from one service candidate to another

G Some service operations candidates remained the same

提交

- We have changed the name of “Poll network folder for invoice” service operation candidate to “Poll folder for new documents”. Why this change was made? Which service design principle was applied?

Fill the blank(s)

Points: 0

设置

此题未设置答案，请点击右侧设置按钮

We have changed the name of “Poll network folder for invoice” service operation candidate to “Poll folder for new documents”. Why this change was made? Which service design principle was applied?

正常使用填空题需3.0以上版本雨课堂

作答

- We have also started adjusting PO Processing Service

Adjusting PO Processing Service (1)

- PO Processing Service – current service operation candidates
  - Receive PO document.
  - Validate PO document.
  - If PO document is invalid, send rejection notification and end process.
  - Transform PO XML document into native electronic PO format.

Adjusting PO Processing Service (2)

- Though listed as such, the "Receive PO document" is not a suitable service operation candidate, as receiving a message is a natural part of service operations (and therefore not generally explicitly accounted for).
- Adjustment
  - Remove "Receive PO document" from our list.

Adjusting PO Processing Service (3)

- PO Processing Service
  - ~~Receive PO document.~~
  - Validate PO document.
  - If PO document is invalid, send rejection notification and end process.
  - Transform PO XML document into native electronic PO format.

Adjusting PO Processing Service (4)

- We then detect a direct dependency between the "If PO document is invalid, send rejection notification and end process" and "Validate PO document" actions.
- Adjustments
  - As a result we decide to combine these into a single operation candidate called "Validate PO document and send rejection notification, if required.“

Adjusting PO Processing Service (5)

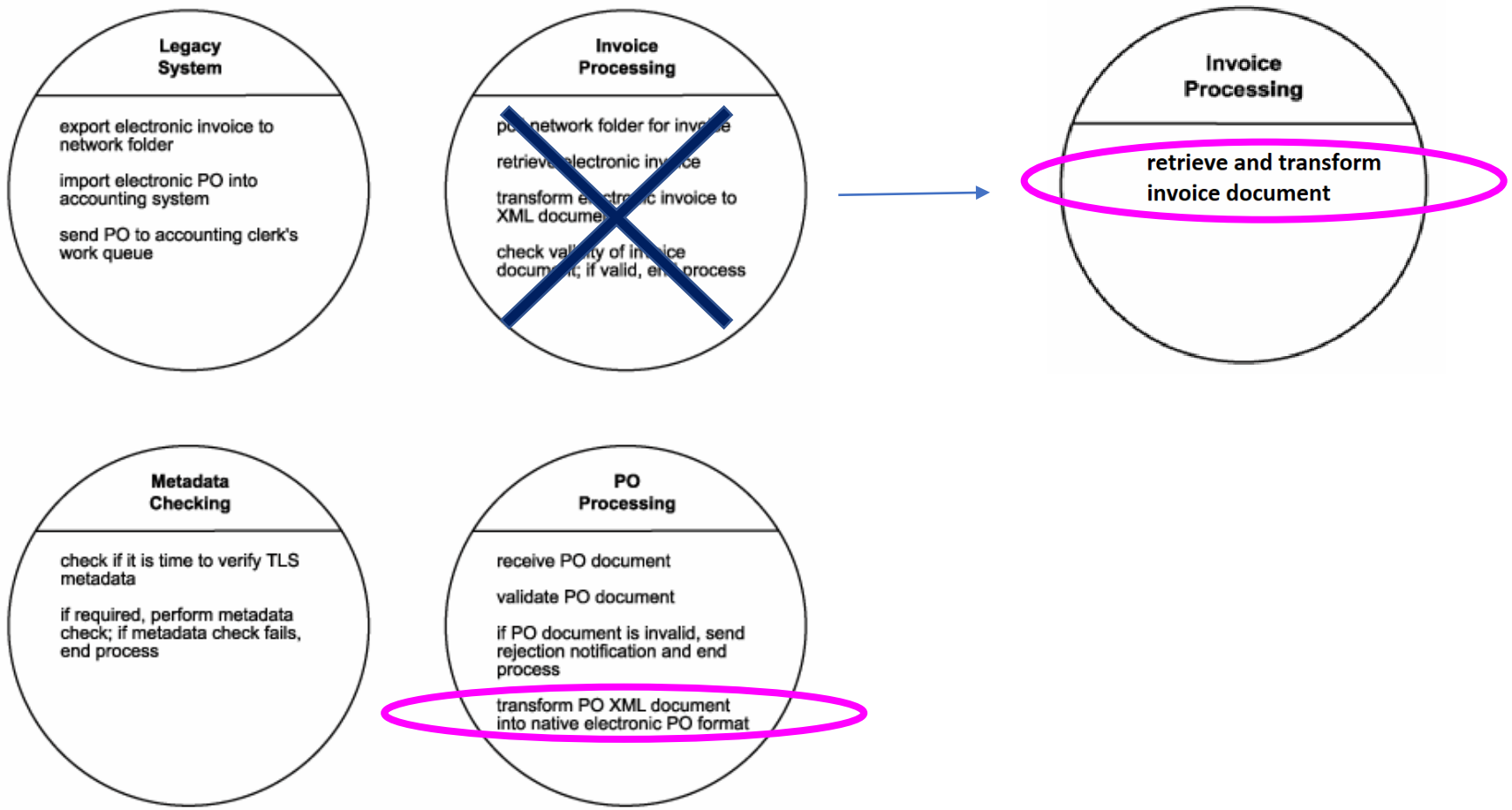
- PO Processing Service
  - ~~Receive PO document.~~
  - ~~Validate PO document.~~
  - ~~If PO document is invalid, send rejection notification and end process.~~
  - Validate PO document and send rejection notification, if required
  - Transform PO XML document into native electronic PO format.

- Today, we will continue to study the SOA delivery lifecycle
- First, let us finish performing step 5 “refine and apply service orientation” on the PO Processing Service

Step 5 RailCo example

- We move on to discover commonality between the "Transform PO XML document into native electronic PO format" action and the "Retrieve and transform invoice document" action from our Invoice Processing Service list.

Step 5 RailCo example



Open Question Points: 10

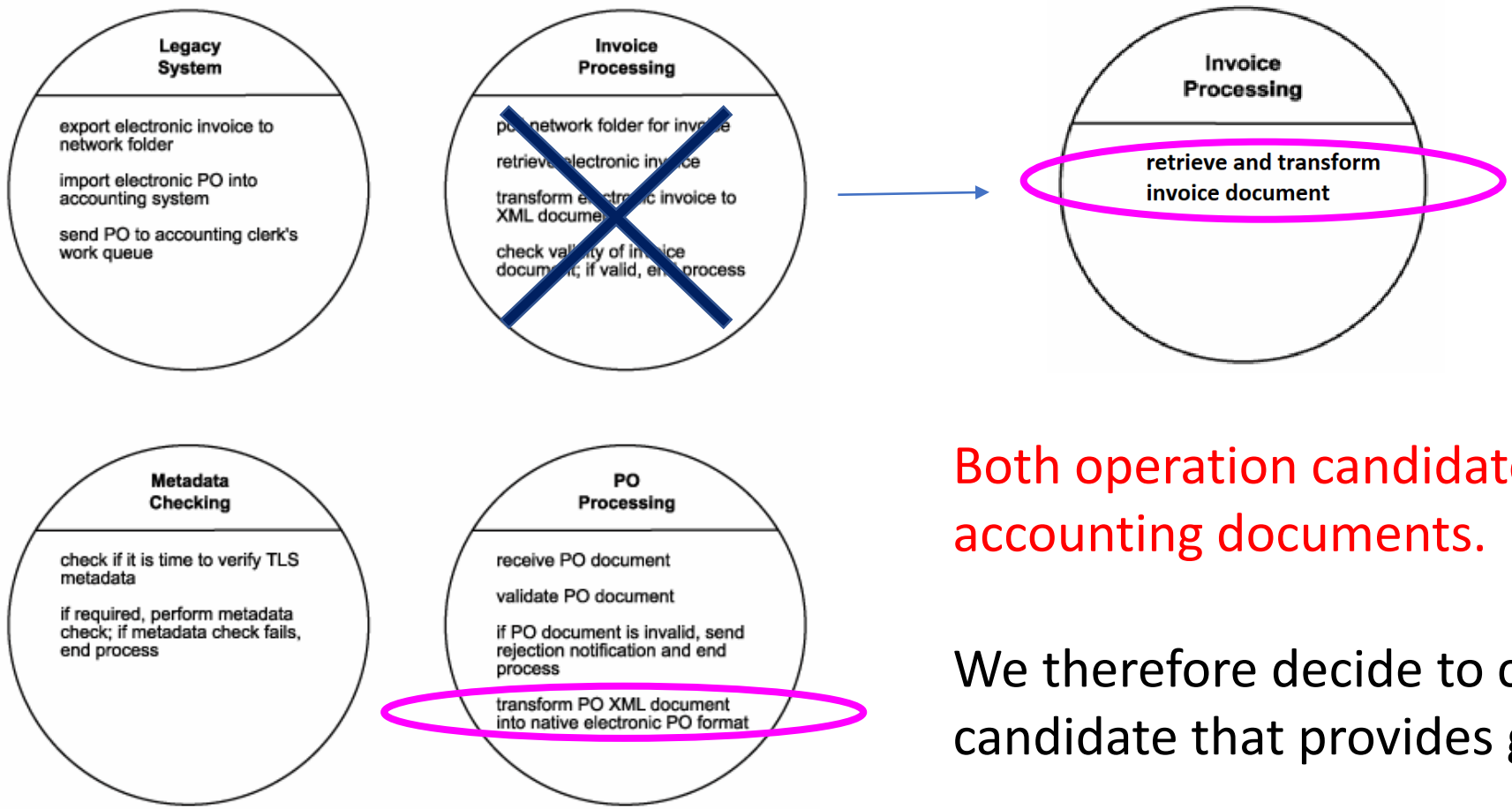
Setting

We discover that there is a commonality between the "Transform PO XML document into native electronic PO format" action and the "Retrieve and transform invoice document" action from our Invoice Processing Service list. Can you specify what these two operation candidates have in common?

Open Question is only supported on Version 2.0 or newer.

Answer

Step 5 RailCo example



Both operation candidates transform accounting documents.

We therefore decide to create a new service candidate that provides generic transformation.

Step 5 RailCo example

- We move on to discover commonality between the "Transform PO XML document into native electronic PO format" action and the "Retrieve and transform invoice document" action from our Invoice Processing Service list.
- Both operation candidates transform accounting documents. We therefore decide to create a new service candidate that provides generic transformation.
- Adjustment – The result is a new grouping category:
  - Transform Accounting Documents Service:
    - Transform XML documents to native format.
    - Transform native documents to XML.

Adjusting PO Processing Service

- Invoice Processing Service has 4 operation candidates at the moment
  - ~~Retrieve and transform invoice document.~~
- PO Processing Service
  - ~~Receive PO document.~~
  - ~~Validate PO document.~~
  - ~~If PO document is invalid, send rejection notification and end process.~~
  - Validate PO document and send rejection notification, if required
  - ~~Transform PO XML document into native electronic PO format~~
- Transform Accounting Documents Service:
  - Transform XML documents to native format.
  - Transform native documents to XML.

Step 5 RailCo example (11)

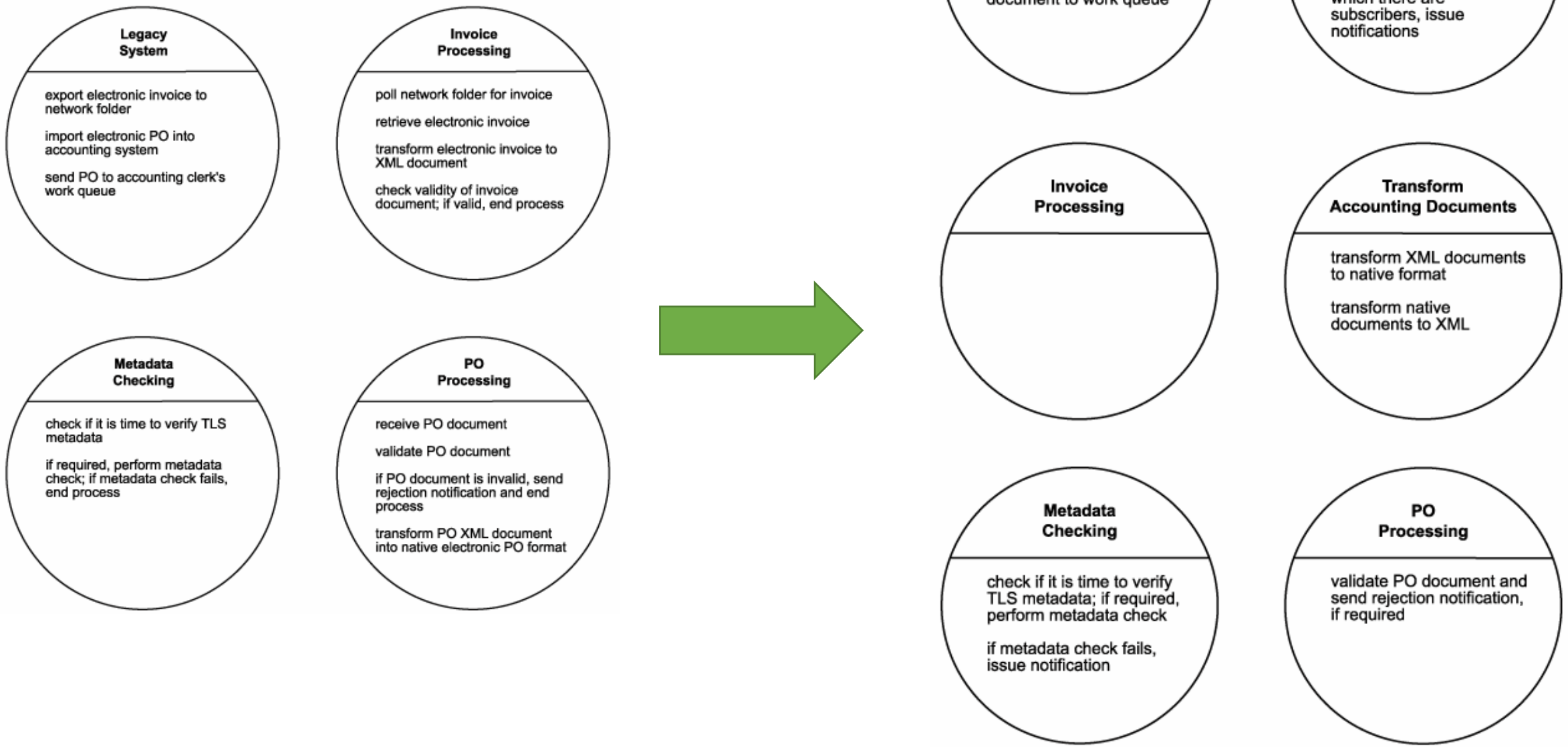
- The revised PO Processing Service list is left with just one step:
  - Validate PO document and send rejection notification, if required.

Step 5 RailCo example

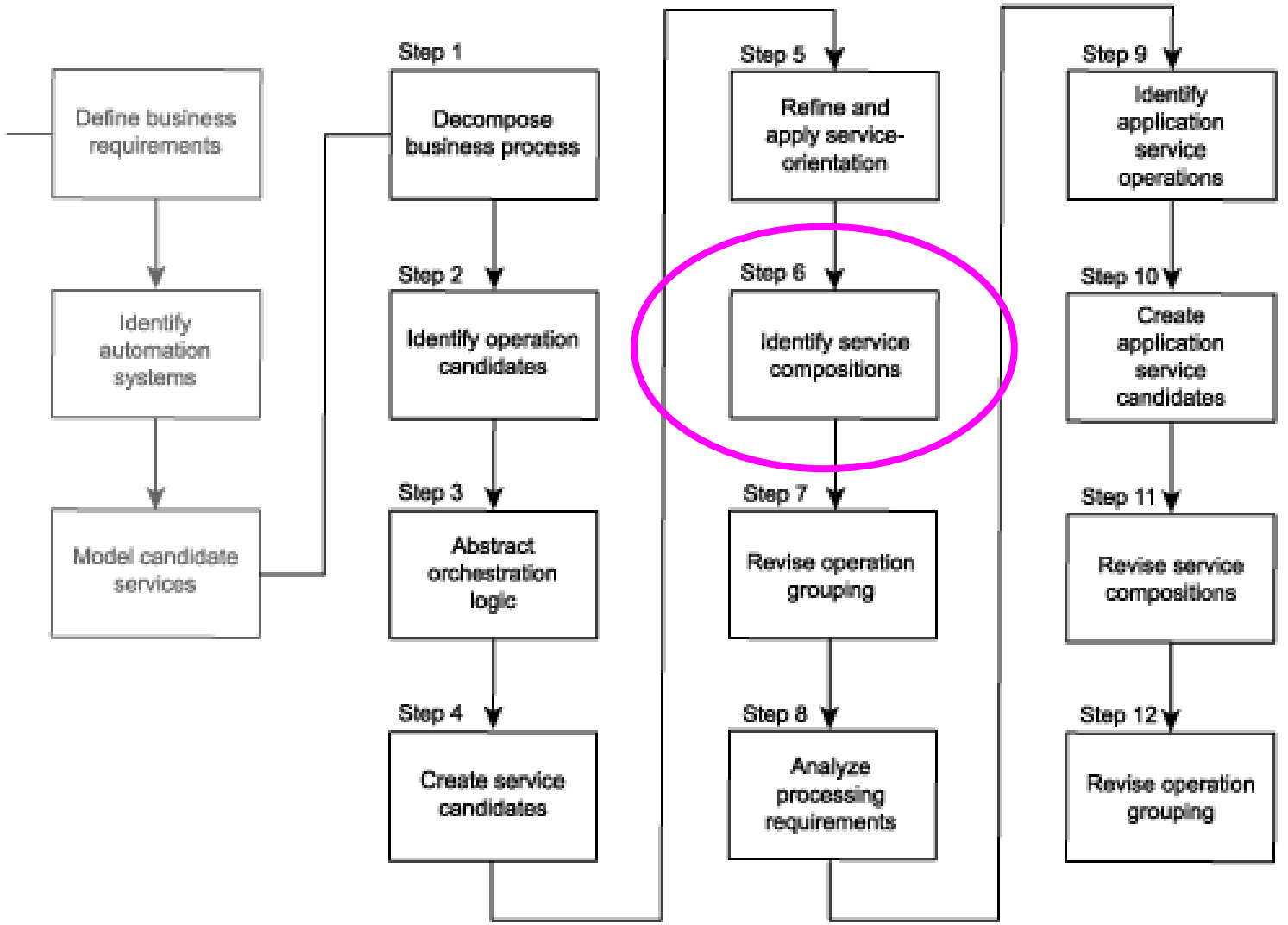
- Finally, our last group of operation candidates is reviewed. The candidates themselves are still relatively suitable for what we intended. However, because we've abstracted these into a generic service candidate, we need to revise the wording to better reflect this. Specifically, we add a notification feature to our Metadata Checking Service candidate.
- The revised Metadata Checking Service list contains the following steps:
  - Check if it is time to verify TLS metadata. If required, perform metadata check.
  - If metadata check fails, issue notification.

- We have finished making adjustments to our initial design
- Let us have a look what service candidates and service operation candidates are considered in our new design

The revised design



Common steps of modeling process



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 6 RailCo example (1)

- We will continue our RailCo case to illustrate this step

Step 6 RailCo example (2)

- Let's recap some of the service candidates established so far.
  - Legacy System Service
  - Polling Notification Service
  - Transform Accounting Documents Service
  - Metadata Checking Service
- Each of these service candidates represents generic, reusable, and business-agnostic logic. In other words, these can all be classified as application service candidates.
- Collectively they establish a preliminary application service layer.

Step 6 RailCo example (3)

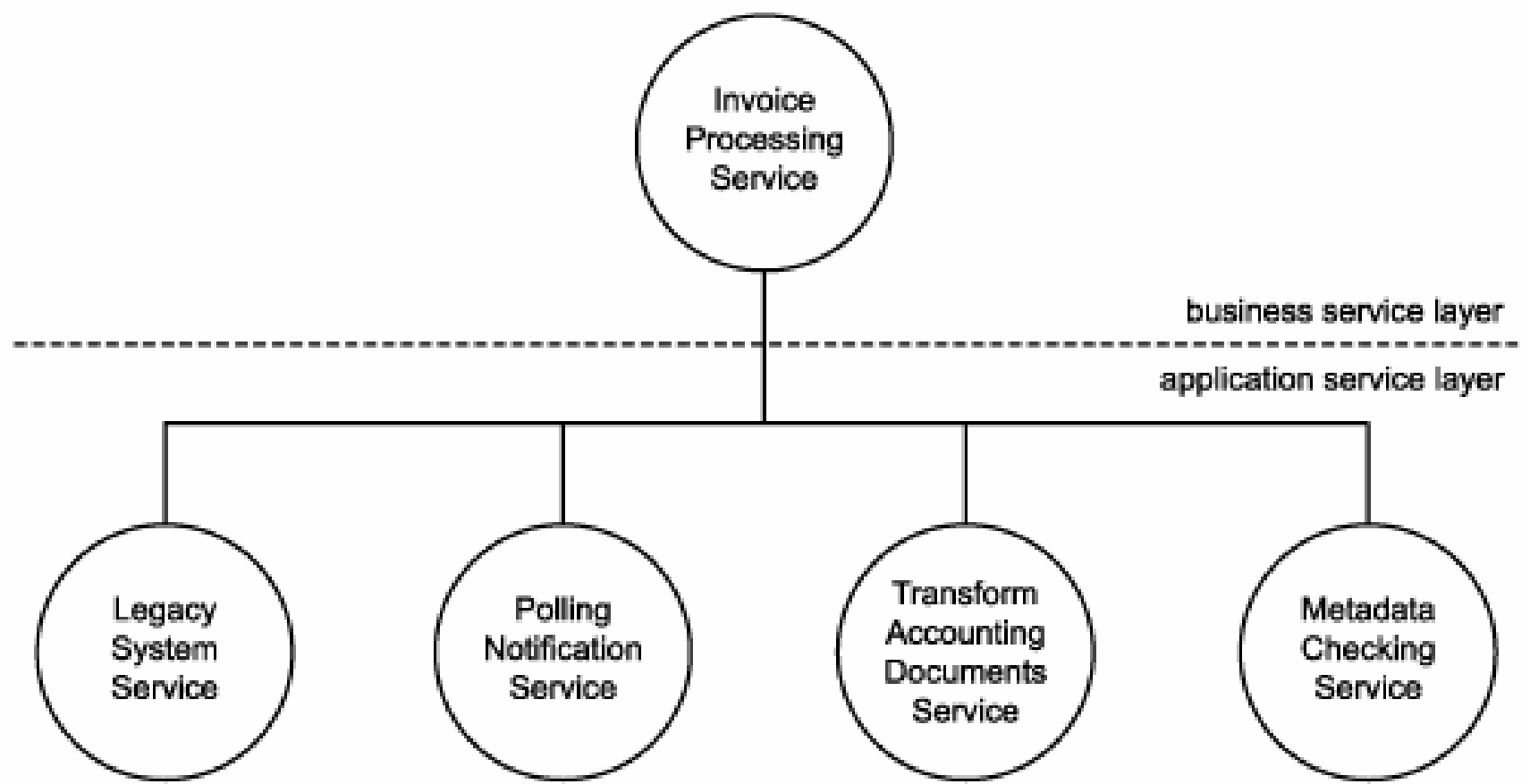
- But what about our business service candidates?
  - PO Processing Service candidate still had one action associated with it
  - Invoice Processing Service operation candidates disappeared after we abstracted all of the associated process actions into separate application service candidates

Step 6 RailCo example (4)

- The fact that we've reduced the processing requirements of these two service candidates does not mean that we don't have a need for them.
- The primary role of task-centric business services is to act as controllers, composing application services to carry out the required business logic.
- Both the PO Processing and Invoice Processing Service candidates establish a preliminary parent business service layer and contain all of the process logic required to compose the underlying application service candidates

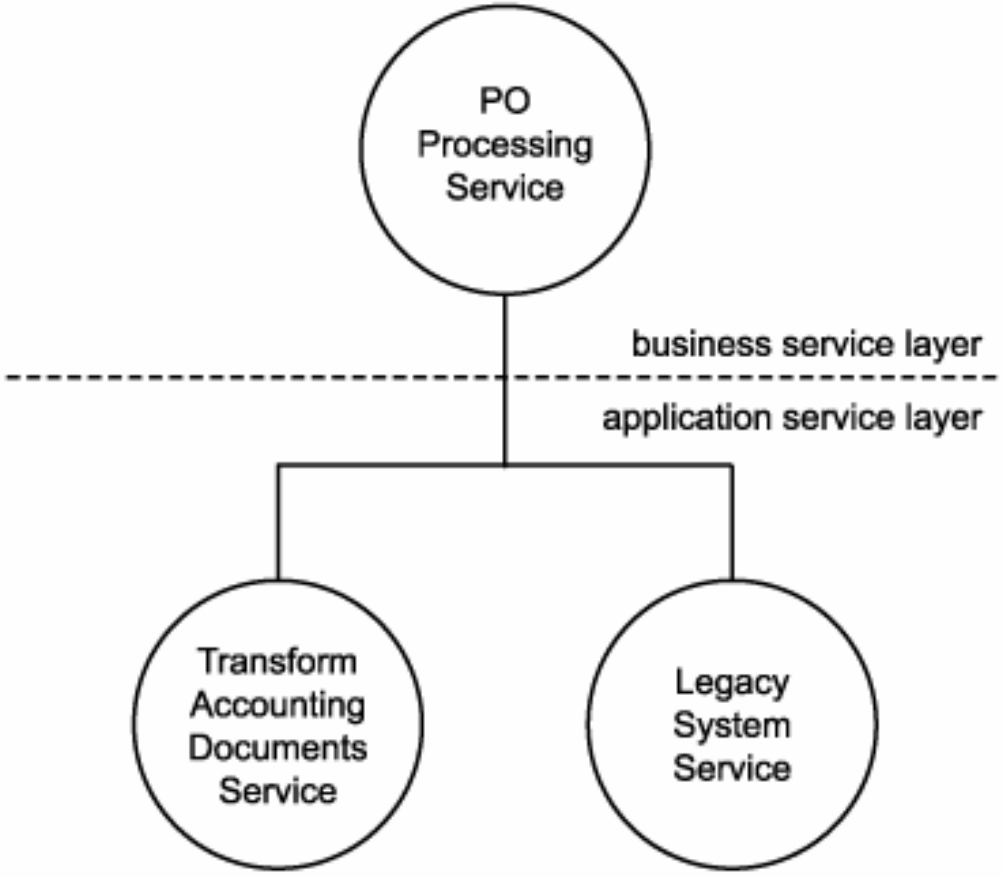
Step 6 RailCo example (5)

- A sample composition representing the Invoice Submission Process.

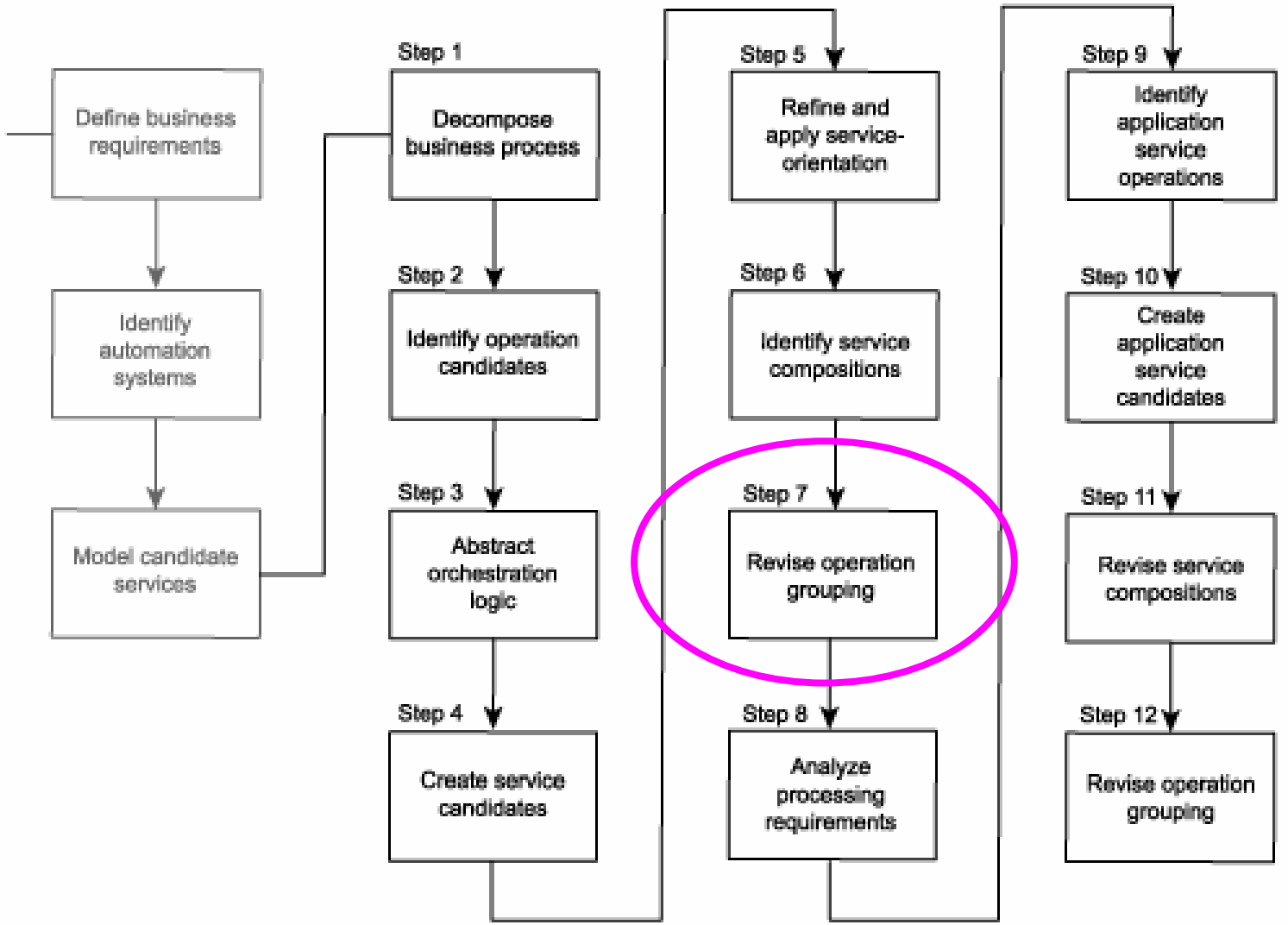


Step 6 RailCo example (6)

- A sample composition representing the Order Fulfillment Process.



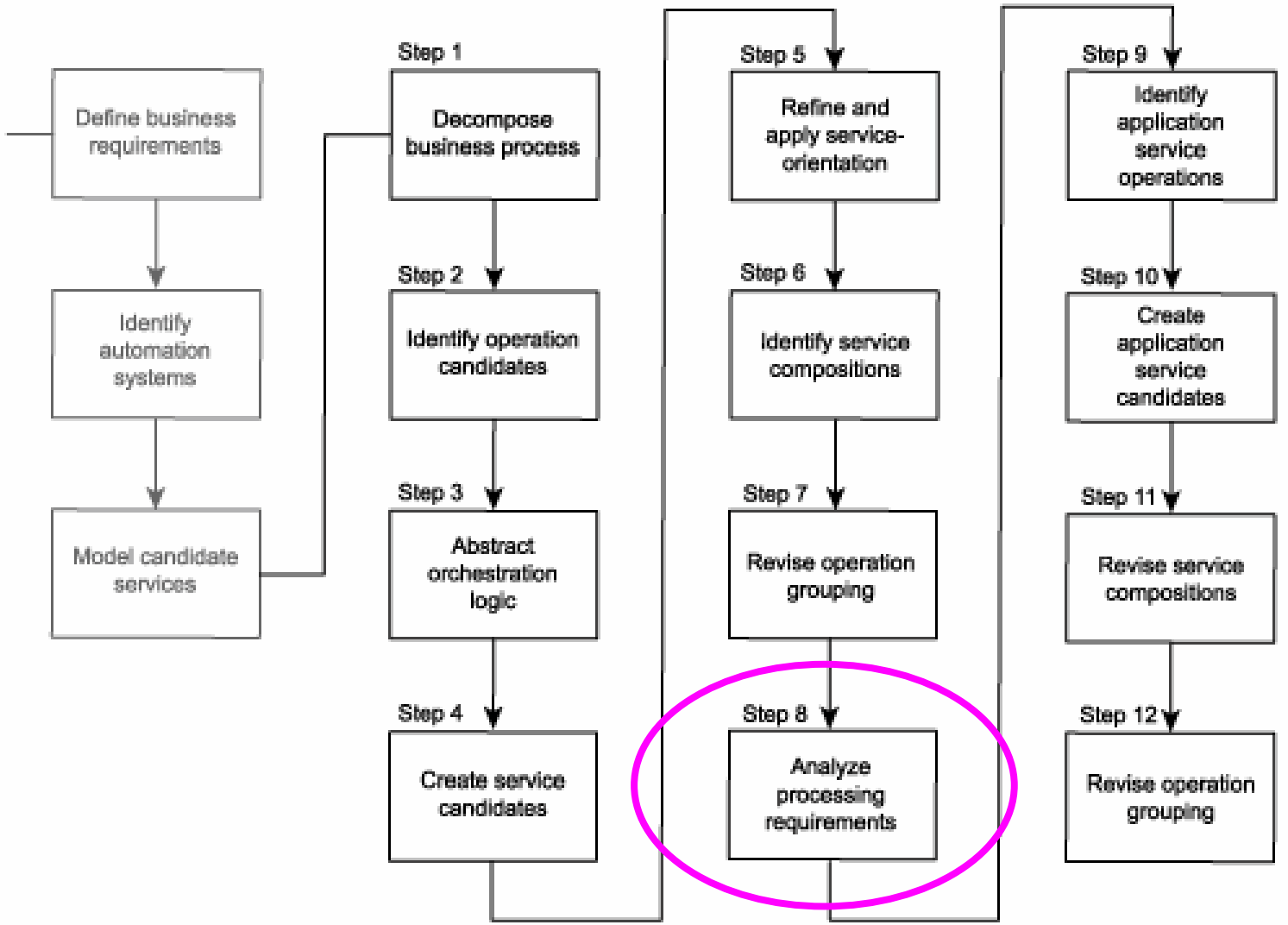
Common steps of modeling process



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.

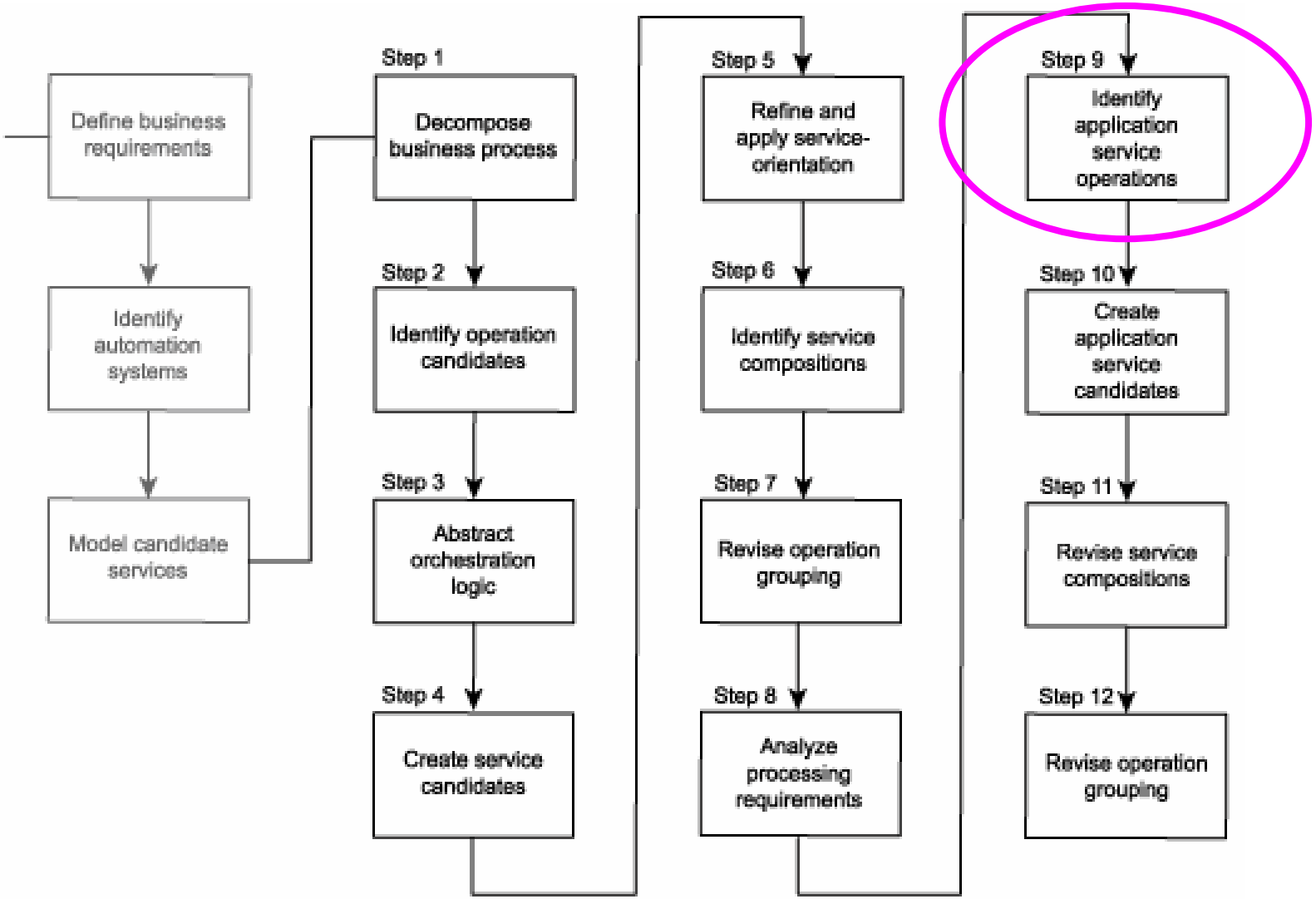
Common steps of modeling process



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?

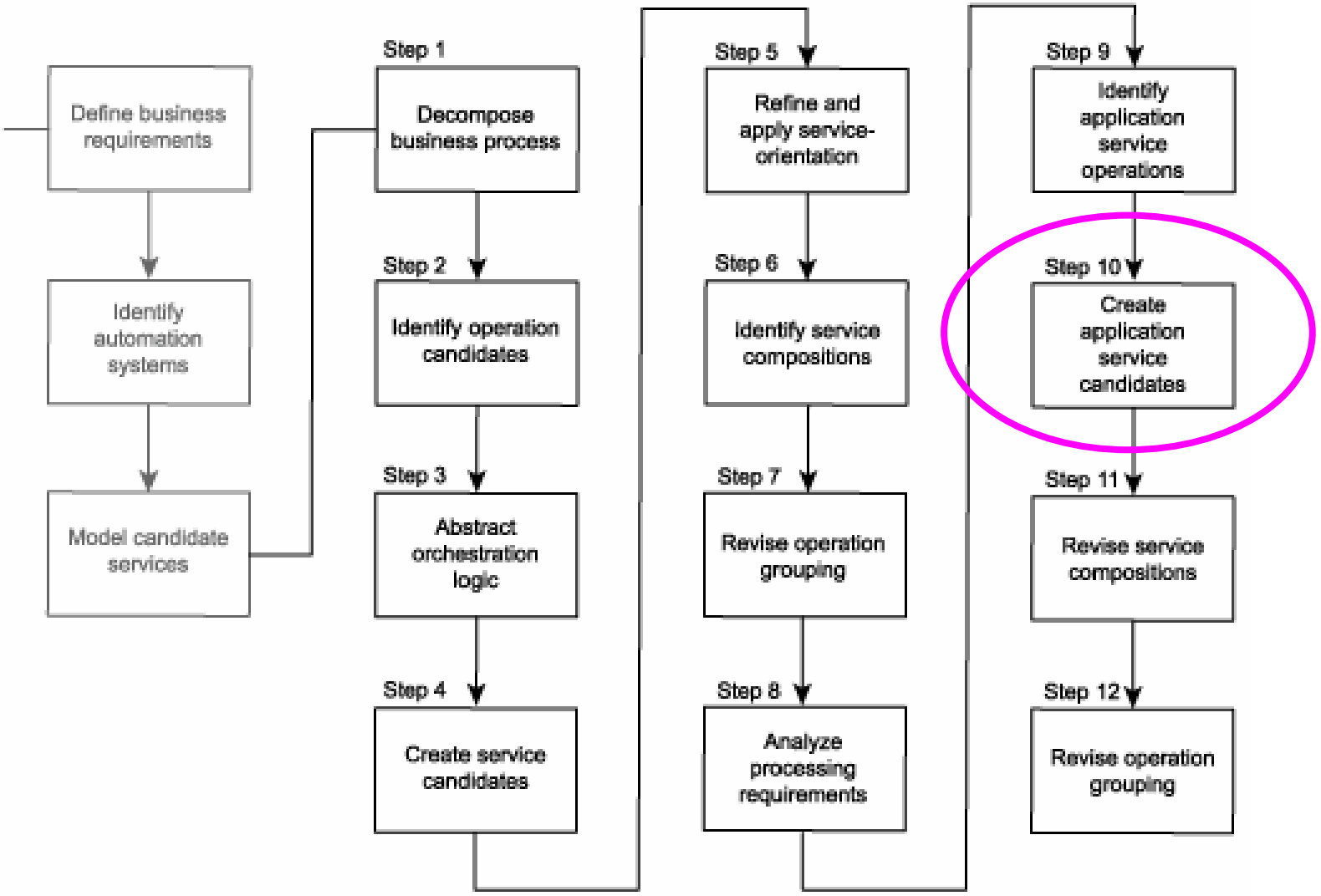
Common steps of modeling process



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.

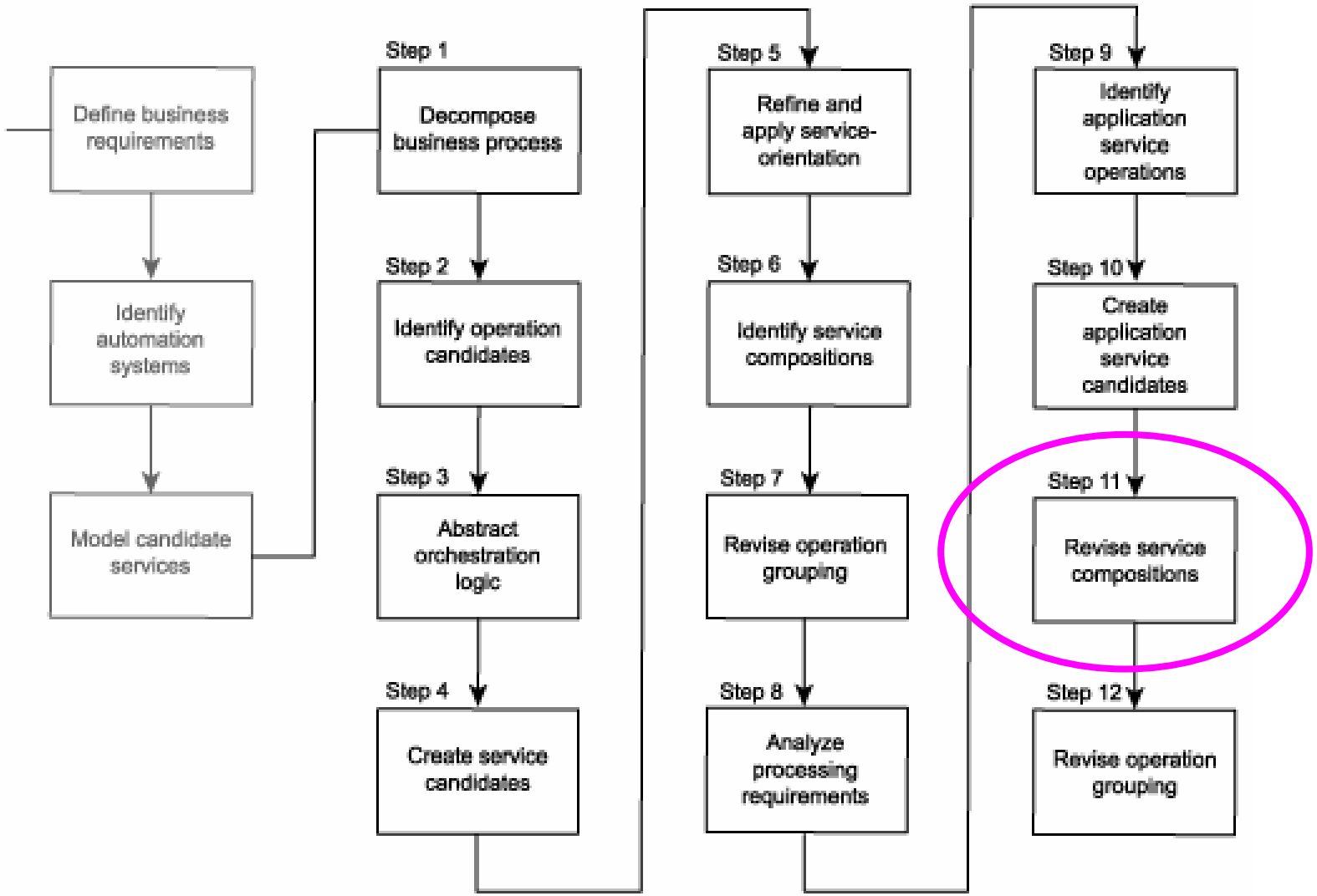
Common steps of modeling process



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

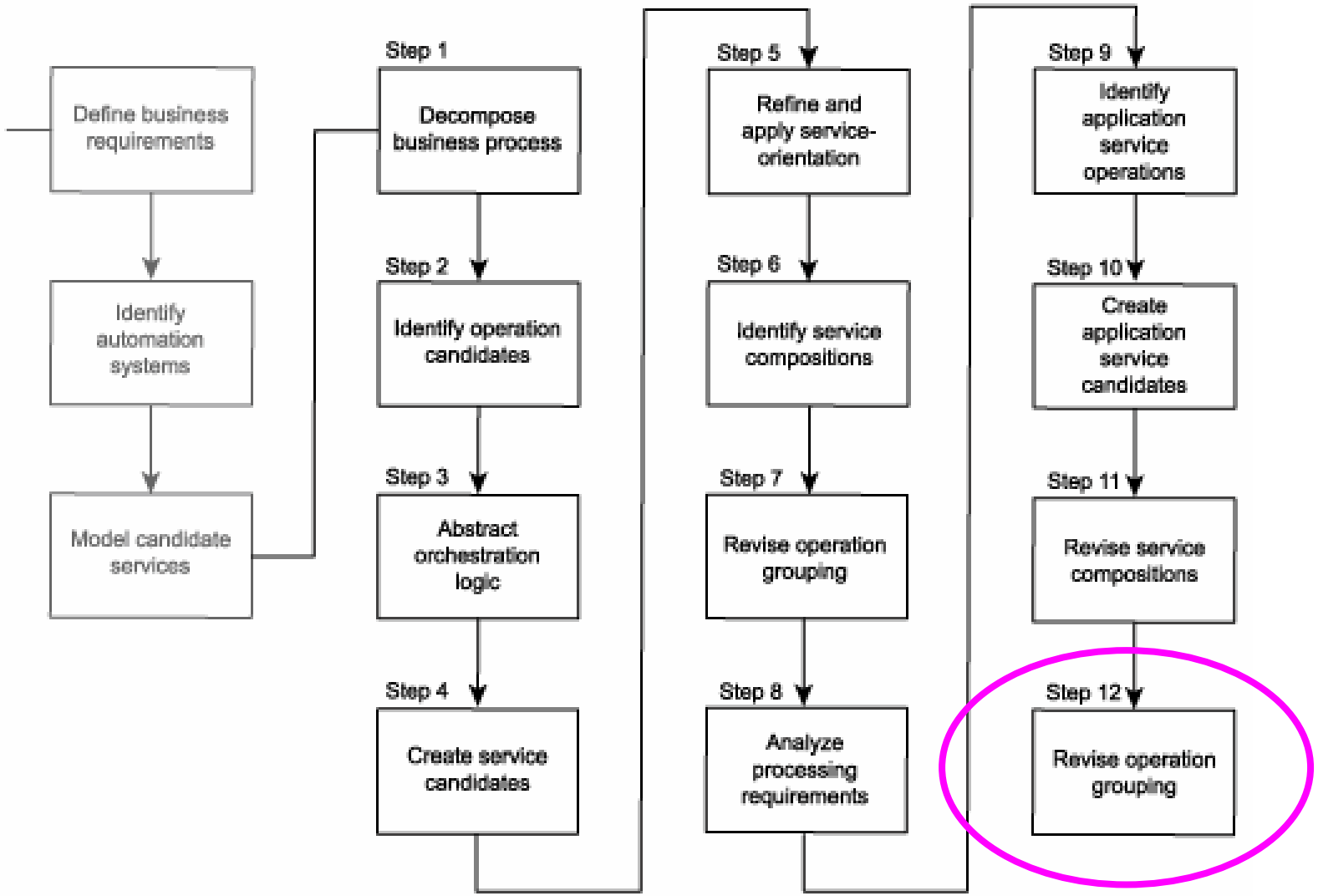
Common steps of modeling process



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.

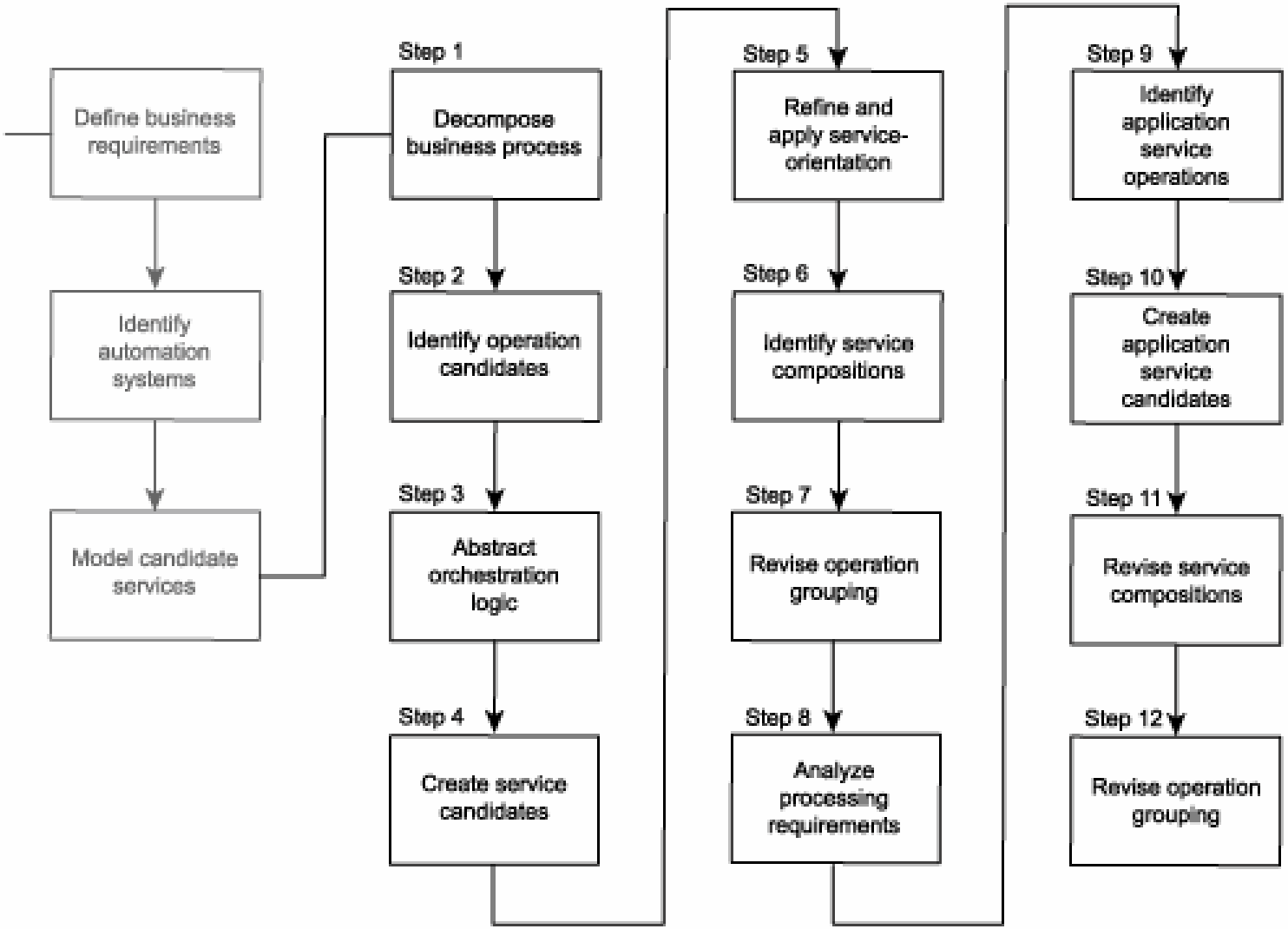
Common steps of modeling process



Step 12: Revise application service operation grouping

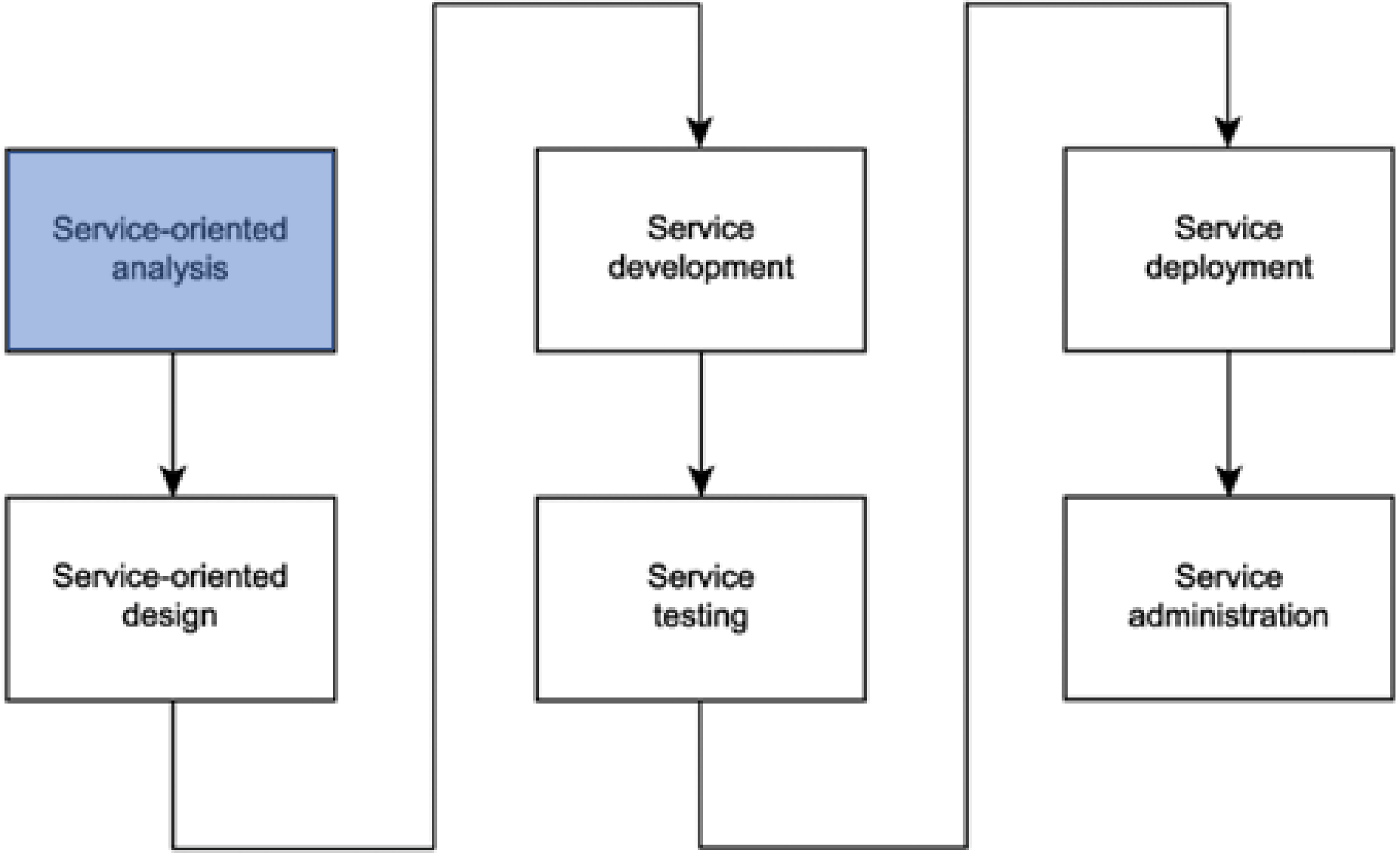
- This step is optional

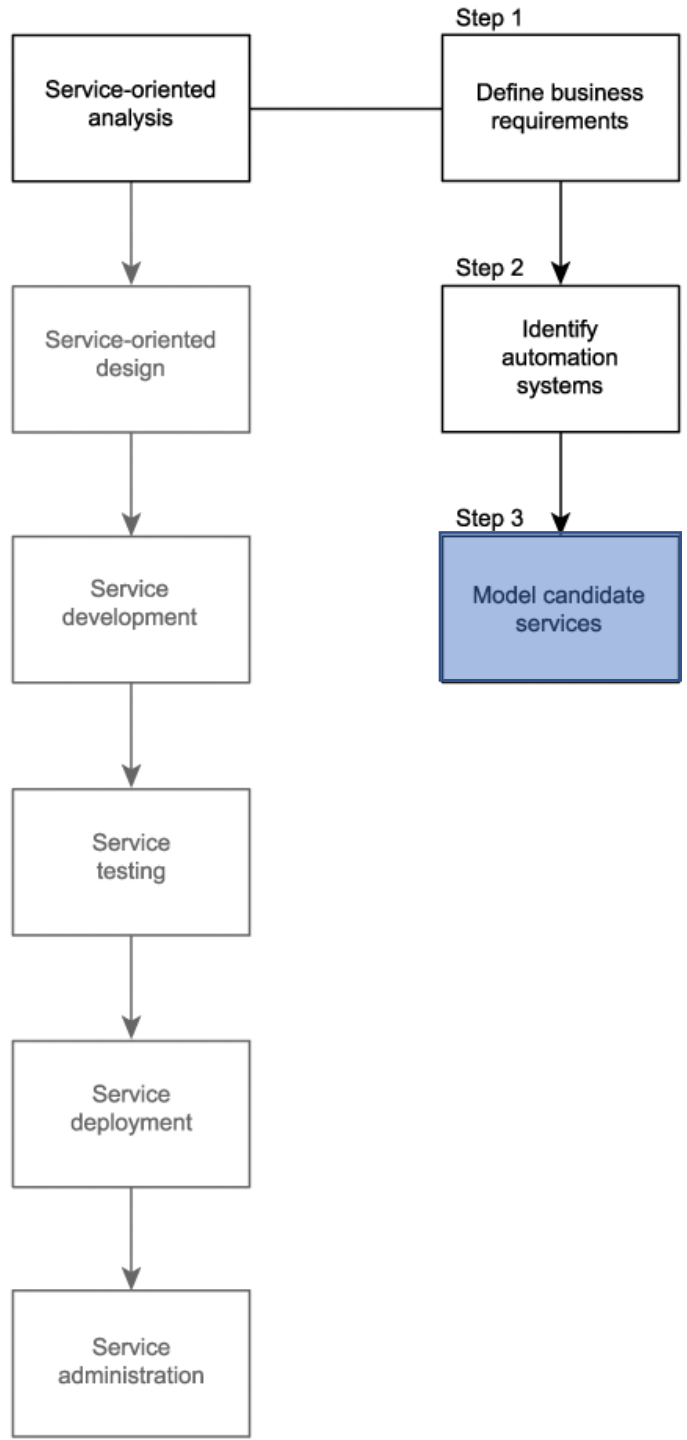
Common steps of modeling process



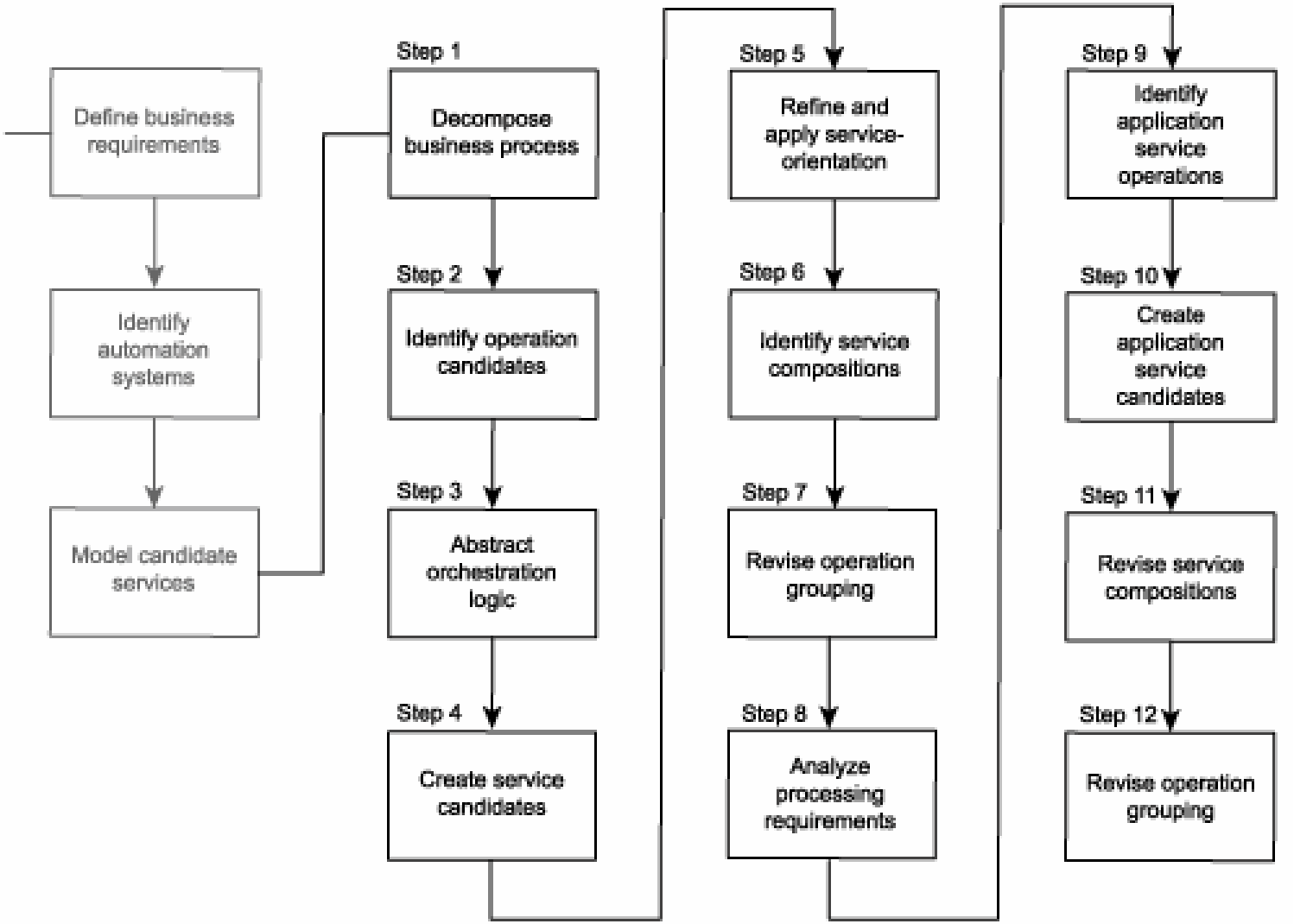
service-oriented analysis review

SOA delivery lifecycle

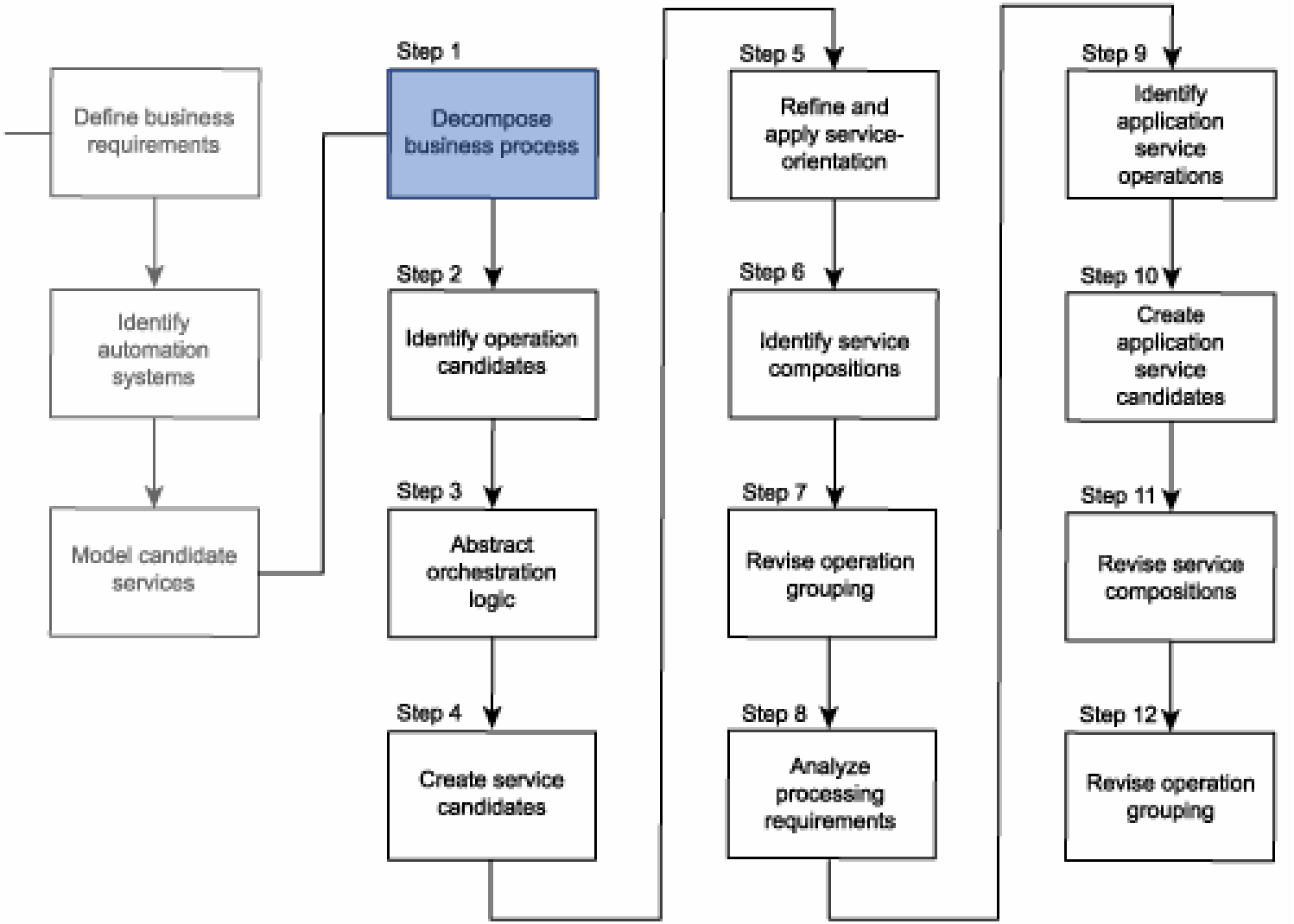



Step 1
Service-oriented analysis
Define business requirements
Step 2
Service-oriented design
Identify automation systems
Step 3
Service development
Model candidate services
Service testing
Service deployment
Service administration

Common steps of modelling process



Step 1: Decompose the business process



Step 1: Decompose the business process

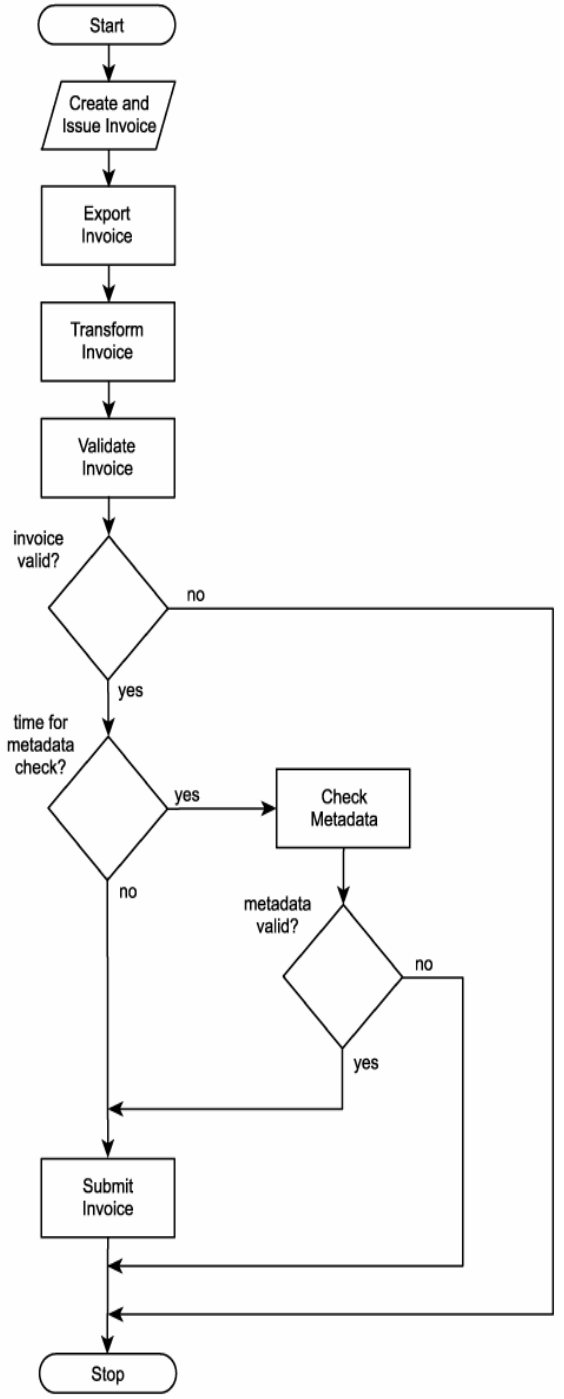
- Input: business process document
- Output: decomposed business process

Input: 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.

Output: Decomposed Invoice Submission Process

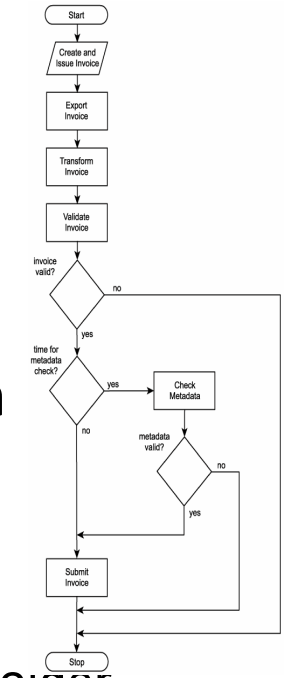
- Create electronic invoice.
- Issue electronic invoice.
- Export electronic invoice to network folder.
- Poll network folder.
- Retrieve electronic invoice.
- Transform electronic invoice to XML document.
- Check validity of invoice document. If invalid, end process.
- Check if it is time to verify TLS metadata.
- If required, perform metadata check. If metadata check fails, end process.



From business process to decomposed business process

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.



Decomposed Invoice Submission Process

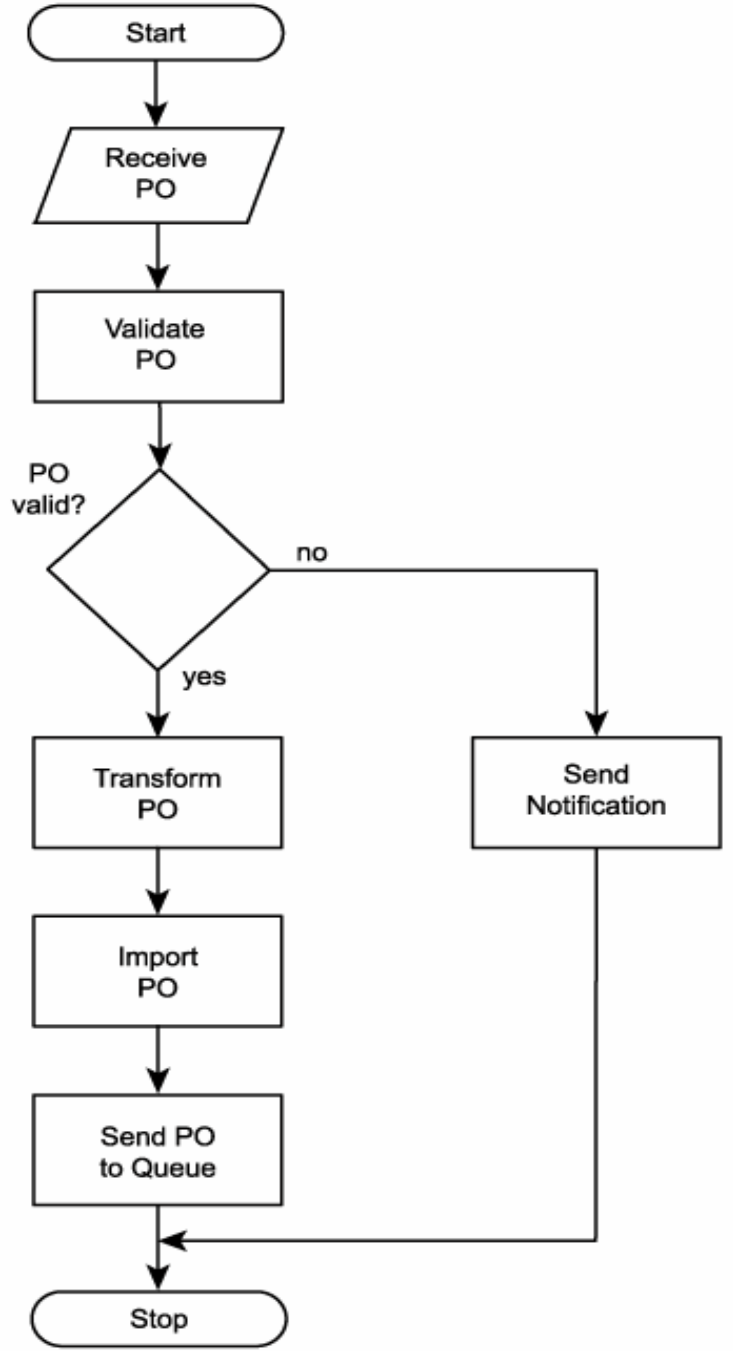
- Create electronic invoice.
- Issue electronic invoice.
- Export electronic invoice to network folder.
- Poll network folder.
- Retrieve electronic invoice.
- Transform electronic invoice to XML document.
- Check validity of invoice document. If invalid, end process.
- Check if it is time to verify TLS metadata.
- If required, perform metadata check. If metadata check fails, end process.

Input: 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.

Output: Decomposed Order Fulfilment Process

- Receive PO document.
- Validate PO document.
- If PO document is invalid, send rejection notification and end process.
- Transform PO XML document into native electronic PO format.
- Import electronic PO into accounting system.
- Send PO to accounting clerk's work queue.



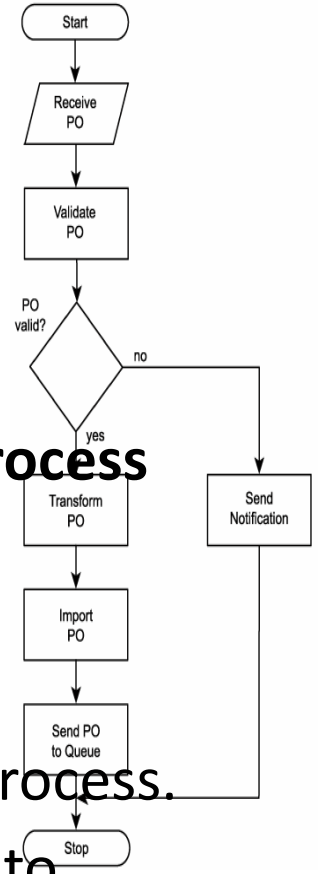
From business process to decomposed business process

Order Fulfilment Process

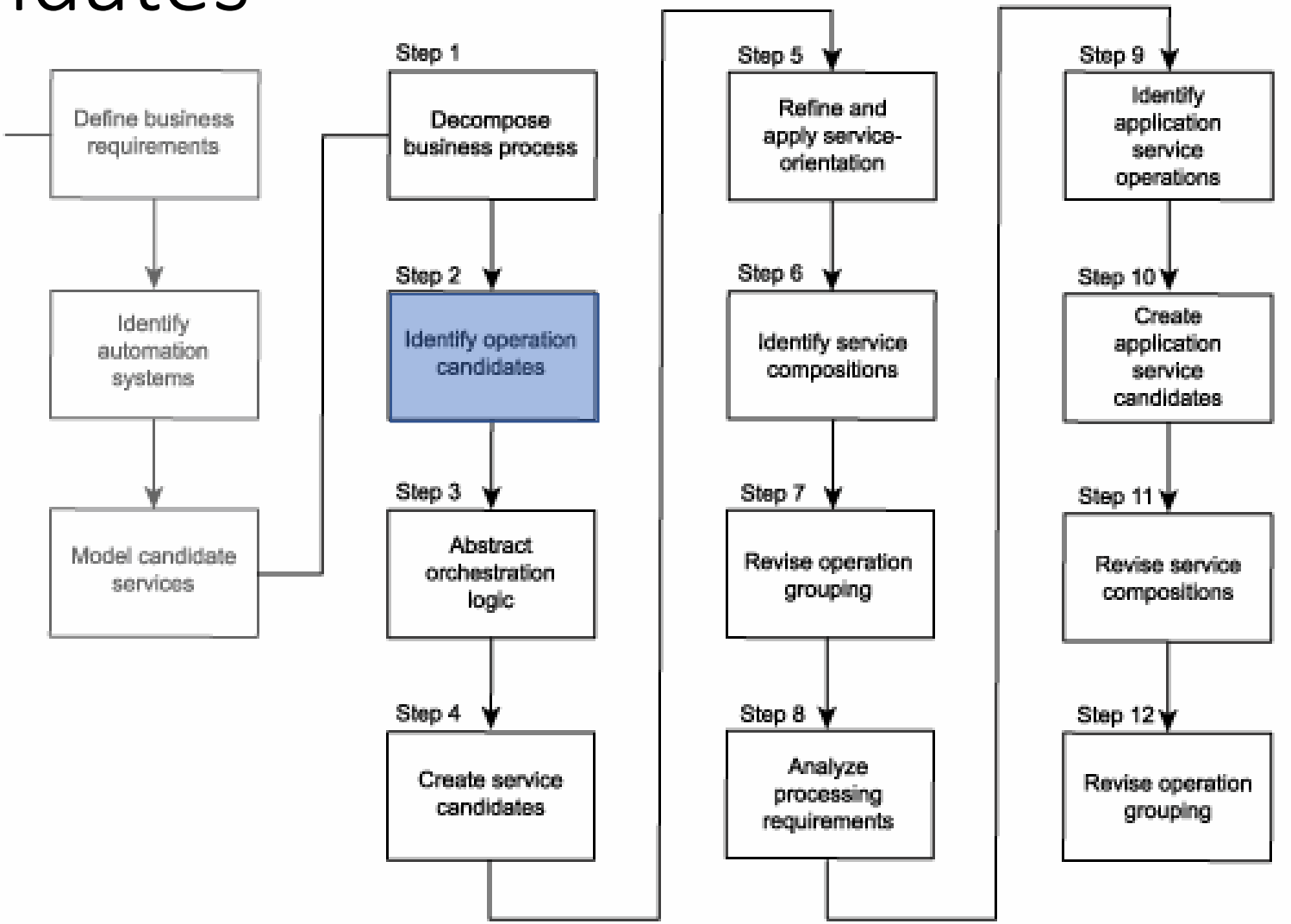
- The RailCo Order Fulfilment Service receives a SOAP message from TLS, containing a payload consisting of a TLS purchase order document.
- The service validates the incoming document. If valid, the document is passed to a custom component. If the TLS PO fails validation, a rejection notification message 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.

Decomposed Order Fulfilment Process

- Receive PO document.
- Validate PO document.
- If PO document is invalid, send rejection notification and end process.
- Transform PO XML document into native electronic PO format.
- Import electronic PO into accounting system.
- Send PO to accounting clerk's work queue.



Step 2: Identify business service operation candidates



Step 2: Identify business service operation candidates

- Input: decomposed business process
- Output: business service operation candidates: filtered out decomposed business process with some steps identified as not belonging to the service oriented solution and others identified as possible service operation candidates

Input: Decomposed Invoice Submission Process

- Create electronic invoice.
- Issue electronic invoice.
- Export electronic invoice to network folder.
- Poll network folder.
- Retrieve electronic invoice.
- Transform electronic invoice to XML document.
- Check validity of invoice document. If invalid, end process.
- Check if it is time to verify TLS metadata.
- If required, perform metadata check. If metadata check fails, end process.

Output: Invoice Submission operation candidates:

- ~~Create electronic invoice.~~ (A manual step performed by the accounting clerk.)
- ~~Issue electronic invoice.~~ (A manual step performed by the accounting clerk.)
- Export electronic invoice to network folder. (Currently a custom developed extension of the legacy system. Could be made part of a generic service candidate.)
- Poll network folder. (Currently performed by a custom developed component. Could be made part of a service candidate.)
- Retrieve electronic invoice. (Same as previous.)
- Transform electronic invoice to XML document. (Same as previous.)
- Check validity of invoice document. If invalid, end process. (Is currently being performed as part of the Invoice Submission Service's parsing routine. No foreseeable need to change this.)
- Check if it is time to verify TLS metadata. (Is currently being performed as part of the Invoice Submission Service's parsing routine. Looks like a potentially reusable operation candidate. Could be moved to a separate service candidate.)
- If required, perform metadata check. If metadata check fails, end process. (Same as previous.)

From decomposed business process to service operation candidates

Decomposed Invoice Submission Process

- Create electronic invoice.
- Issue electronic invoice.
- Export electronic invoice to network folder.
- Poll network folder.
- Retrieve electronic invoice.
- Transform electronic invoice to XML document.
- Check validity of invoice document. If invalid, end process.
- Check if it is time to verify TLS metadata.
- If required, perform metadata check. If metadata check fails, end process.

Invoice Submission operation candidates:

- ~~Create electronic invoice.~~ (A manual step performed by the accounting clerk.)
- ~~Issue electronic invoice.~~ (A manual step performed by the accounting clerk.)
- Export electronic invoice to network folder. (Currently a custom developed extension of the legacy system. Could be made part of a generic service candidate.)
- Poll network folder. (Currently performed by a custom developed component. Could be made part of a service candidate.)
- Retrieve electronic invoice. (Same as previous.)
- Transform electronic invoice to XML document. (Same as previous.)
- Check validity of invoice document. If invalid, end process. (Is currently being performed as part of the Invoice Submission Service's parsing routine. No foreseeable need to change this.)
- Check if it is time to verify TLS metadata. (Is currently being performed as part of the Invoice Submission Service's parsing routine. Looks like a potentially reusable operation candidate. Could be moved to a separate service candidate.)
- If required, perform metadata check. If metadata check fails, end process. (Same as previous.)

Input: Decomposed Order Fulfilment Process

- Receive PO document.
- Validate PO document.
- If PO document is invalid, send rejection notification and end process.
- Transform PO XML document into native electronic PO format.
- Import electronic PO into accounting system.
- Send PO to accounting clerk's work queue.

Output: Order Fulfillment operation candidates

- Receive PO document. (Is currently being performed by the Order Fulfillment Service. No foreseeable need to change this.)
- Validate PO document. (Same as previous.)
- If PO document is invalid, send rejection notification and end process. (Same as previous.)
- Transform PO XML document into native electronic PO format. (Currently performed by a custom developed component. Could be made part of a service candidate.)
- Import electronic PO into accounting system. (Currently a custom developed extension of the legacy system. Could be made part of a generic service candidate.)
- Send PO to accounting clerk's work queue. (Same as previous.)

From decomposed business process to service operation candidates

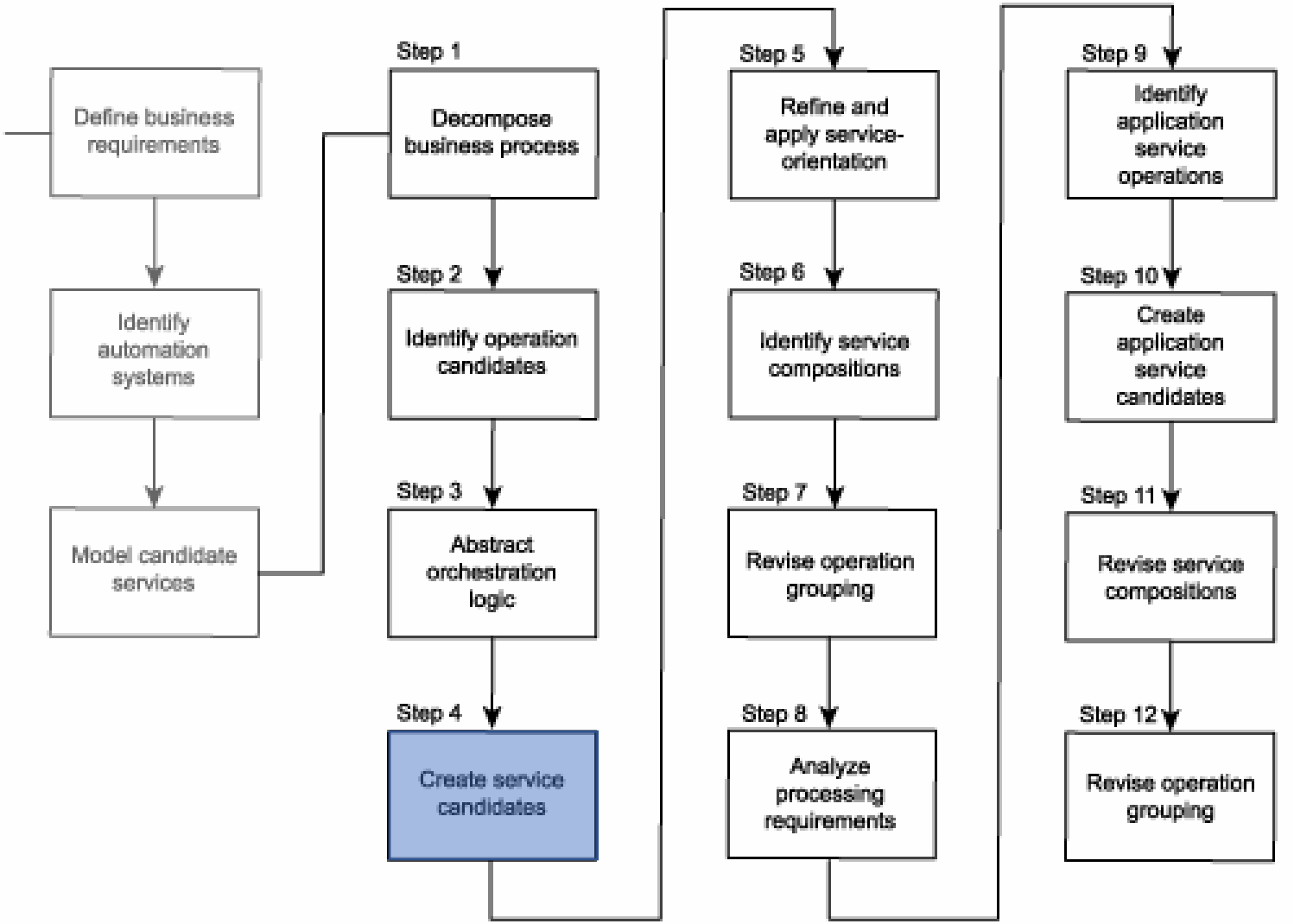
Decomposed Order Fulfilment Process

- Receive PO document.
- Validate PO document.
- If PO document is invalid, send rejection notification and end process.
- Transform PO XML document into native electronic PO format.
- Import electronic PO into accounting system.
- Send PO to accounting clerk's work queue.

Order Fulfillment operation candidates

- Receive PO document. (Is currently being performed by the Order Fulfillment Service. No foreseeable need to change this.)
- Validate PO document. (Same as previous.)
- If PO document is invalid, send rejection notification and end process. (Same as previous.)
- Transform PO XML document into native electronic PO format. (Currently performed by a custom developed component. Could be made part of a service candidate.)
- Import electronic PO into accounting system. (Currently a custom developed extension of the legacy system. Could be made part of a generic service candidate.)
- Send PO to accounting clerk's work queue. (Same as previous.)

Step 4: Create business service candidates



Step 4: Create business service candidates

- Input: business service operation candidates
- Output: business service candidates: business service operation candidates grouped into one or more logical contexts

Input: Invoice Submission operation candidates:

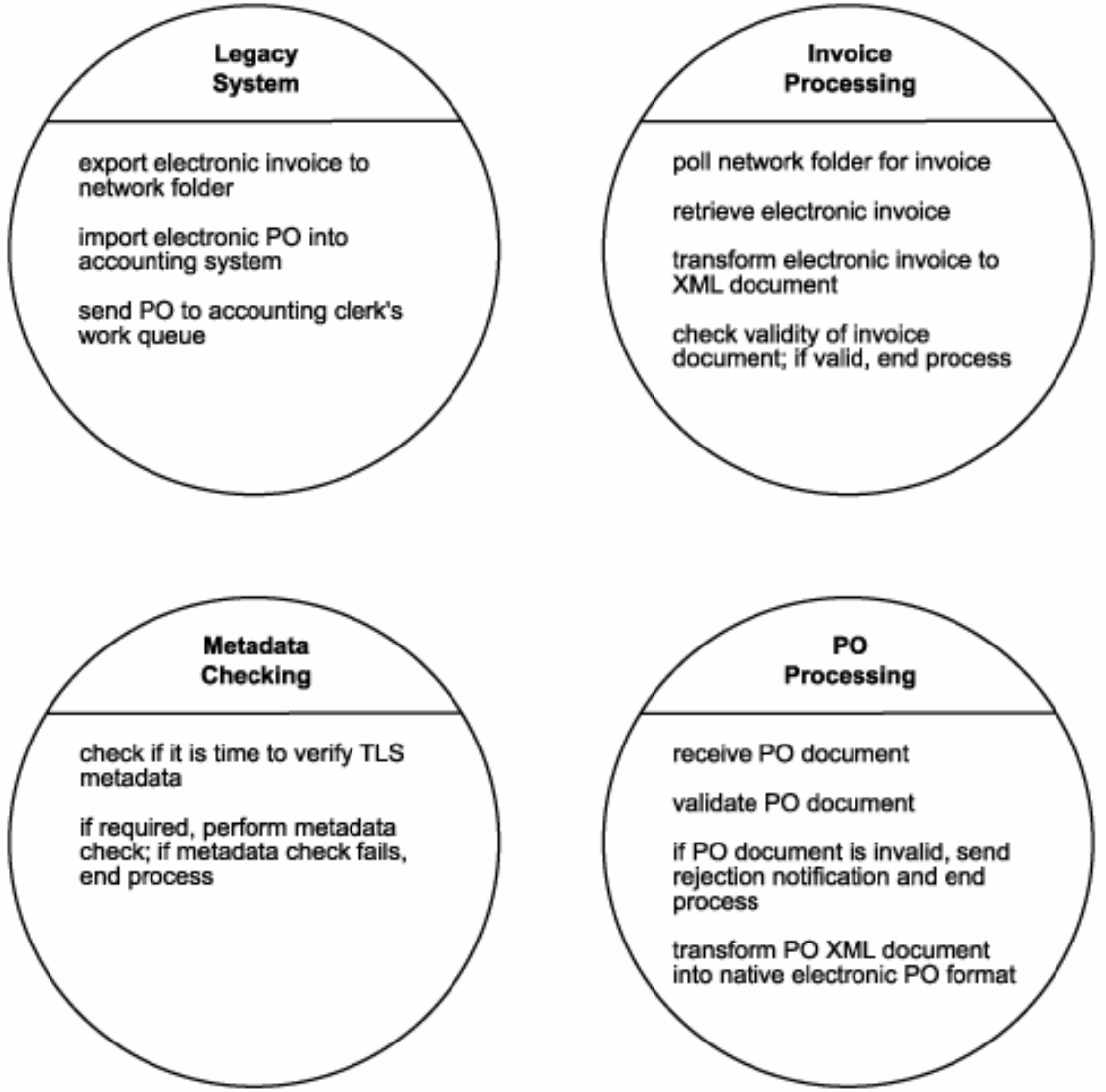
- ~~Create electronic invoice.~~ (A manual step performed by the accounting clerk.)
- ~~Issue electronic invoice.~~ (A manual step performed by the accounting clerk.)
- Export electronic invoice to network folder. (Currently a custom developed extension of the legacy system. Could be made part of a generic service candidate.)
- Poll network folder. (Currently performed by a custom developed component. Could be made part of a service candidate.)
- Retrieve electronic invoice. (Same as previous.)
- Transform electronic invoice to XML document. (Same as previous.)
- Check validity of invoice document. If invalid, end process. (Is currently being performed as part of the Invoice Submission Service's parsing routine. No foreseeable need to change this.)
- Check if it is time to verify TLS metadata. (Is currently being performed as part of the Invoice Submission Service's parsing routine. Looks like a potentially reusable operation candidate. Could be moved to a separate service candidate.)
- If required, perform metadata check. If metadata check fails, end process. (Same as previous.)

Input: Order Fulfillment operation candidates

- Receive PO document. (Is currently being performed by the Order Fulfillment Service. No foreseeable need to change this.)
- Validate PO document. (Same as previous.)
- If PO document is invalid, send rejection notification and end process. (Same as previous.)
- Transform PO XML document into native electronic PO format. (Currently performed by a custom developed component. Could be made part of a service candidate.)
- Import electronic PO into accounting system. (Currently a custom developed extension of the legacy system. Could be made part of a generic service candidate.)
- Send PO to accounting clerk's work queue. (Same as previous.)

Output: service candidates

- Legacy System Service
  - Export electronic invoice to network folder.
  - Import electronic PO into accounting system.
  - Send PO to accounting clerk's work queue.
- Invoice Processing Service
  - Poll network folder for invoice.
  - Retrieve electronic invoice.
  - Transform electronic invoice to XML document.
  - Check validity of invoice document. If invalid, end process.
- PO Processing Service
  - Receive PO document.
  - Validate PO document.
  - If PO document is invalid, send rejection notification and end process.
  - Transform PO XML document into native electronic PO format.
- Metadata Checking Service
  - Check if it is time to verify TLS metadata.
  - If required, perform metadata check. If metadata check fails, end process.



From service operation candidates to service candidates

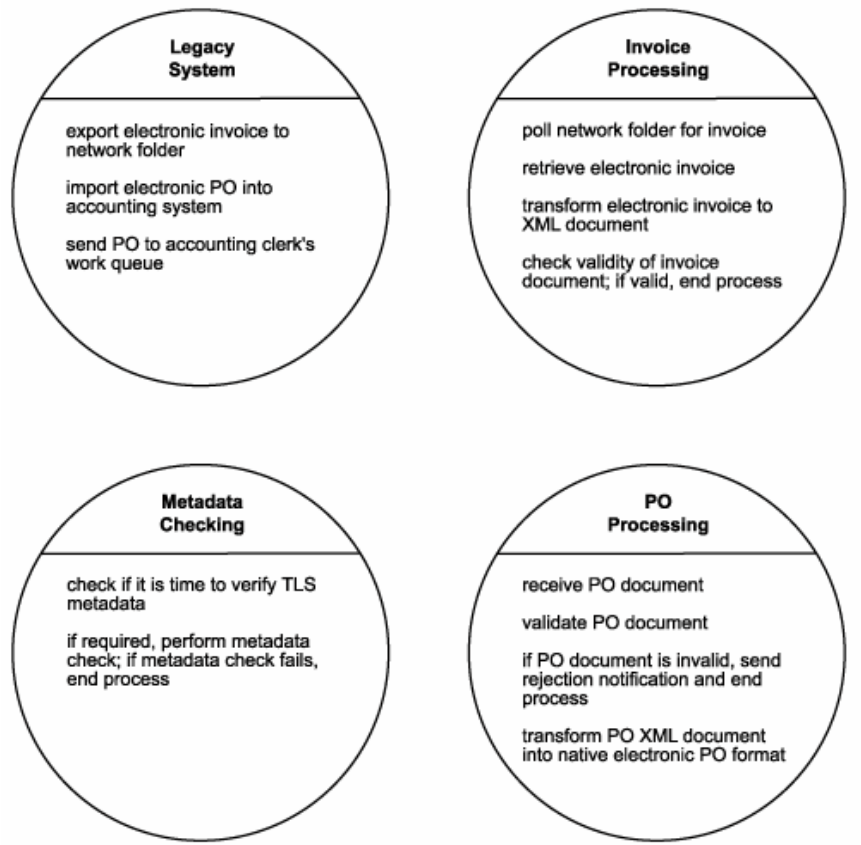
Invoice Submission operation candidates:

- ~~Create electronic invoice.~~ (A manual step performed by the accounting clerk.)
- ~~Issue electronic invoice.~~ (A manual step performed by the accounting clerk.)
- Export electronic invoice to network folder. (Currently a custom developed extension of the legacy system. Could be made part of a generic service candidate.)
- Poll network folder. (Currently performed by a custom developed component. Could be made part of a service candidate.)
- Retrieve electronic invoice. (Same as previous.)
- Transform electronic invoice to XML document. (Same as previous.)
- Check validity of invoice document. If invalid, end process. (Is currently being performed as part of the Invoice Submission Service's parsing routine. No foreseeable need to change this.)
- Check if it is time to verify TLS metadata. (Is currently being performed as part of the Invoice Submission Service's parsing routine. Looks like a potentially reusable operation candidate. Could be moved to a separate service candidate.)
- If required, perform metadata check. If metadata check fails, end process. (Same as previous.)

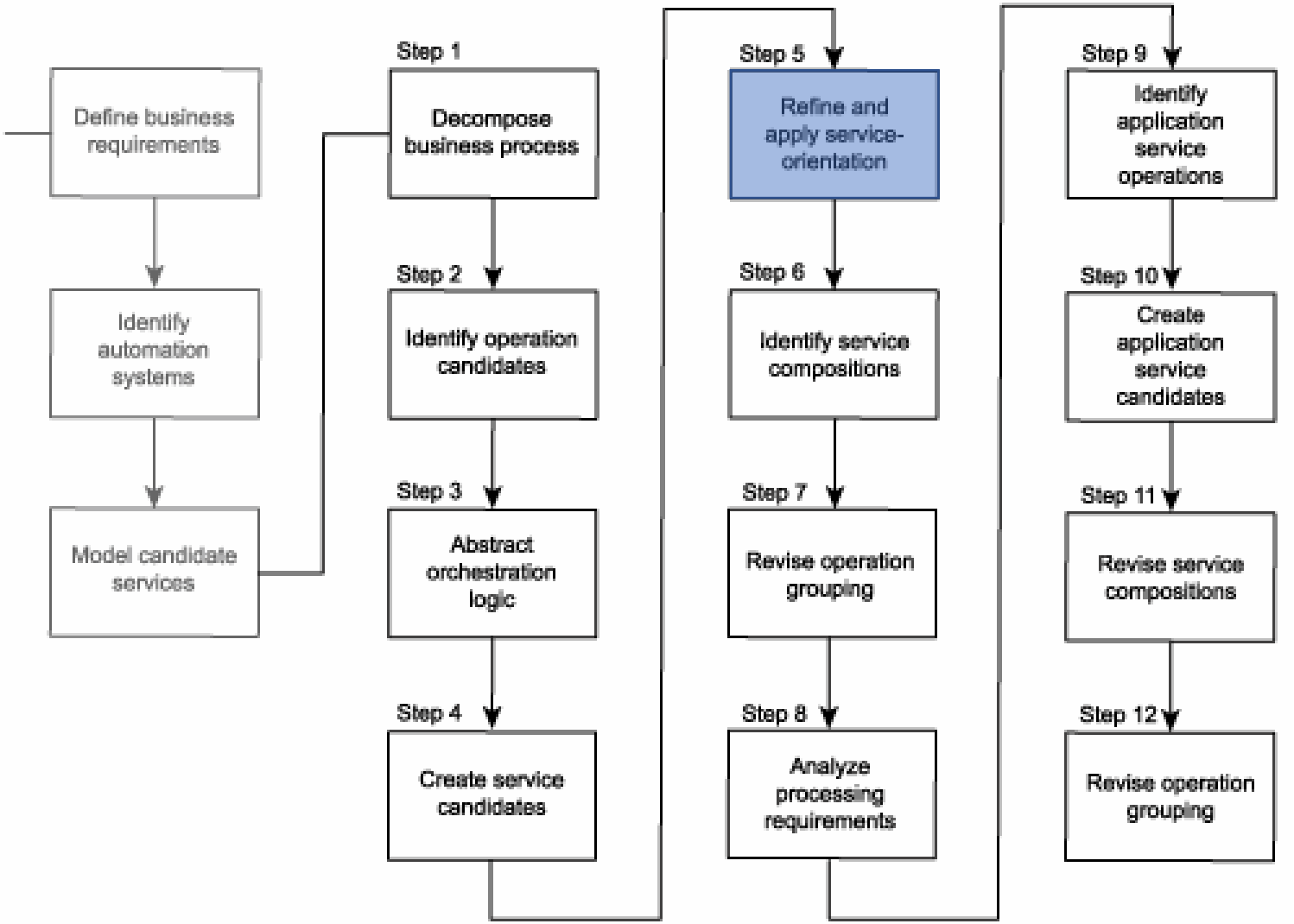
Order Fulfillment operation candidates:

- Receive PO document. (Is currently being performed by the Order Fulfillment Service. No foreseeable need to change this.)
- Validate PO document. (Same as previous.)
- If PO document is invalid, send rejection notification and end process. (Same as previous.)
- Transform PO XML document into native electronic PO format. (Currently performed by a custom developed component. Could be made part of a service candidate.)
- Import electronic PO into accounting system. (Currently a custom developed extension of the legacy system. Could be made part of a generic service candidate.)
- Send PO to accounting clerk's work queue. (Same as previous.)

service candidates



Common steps of modeling process

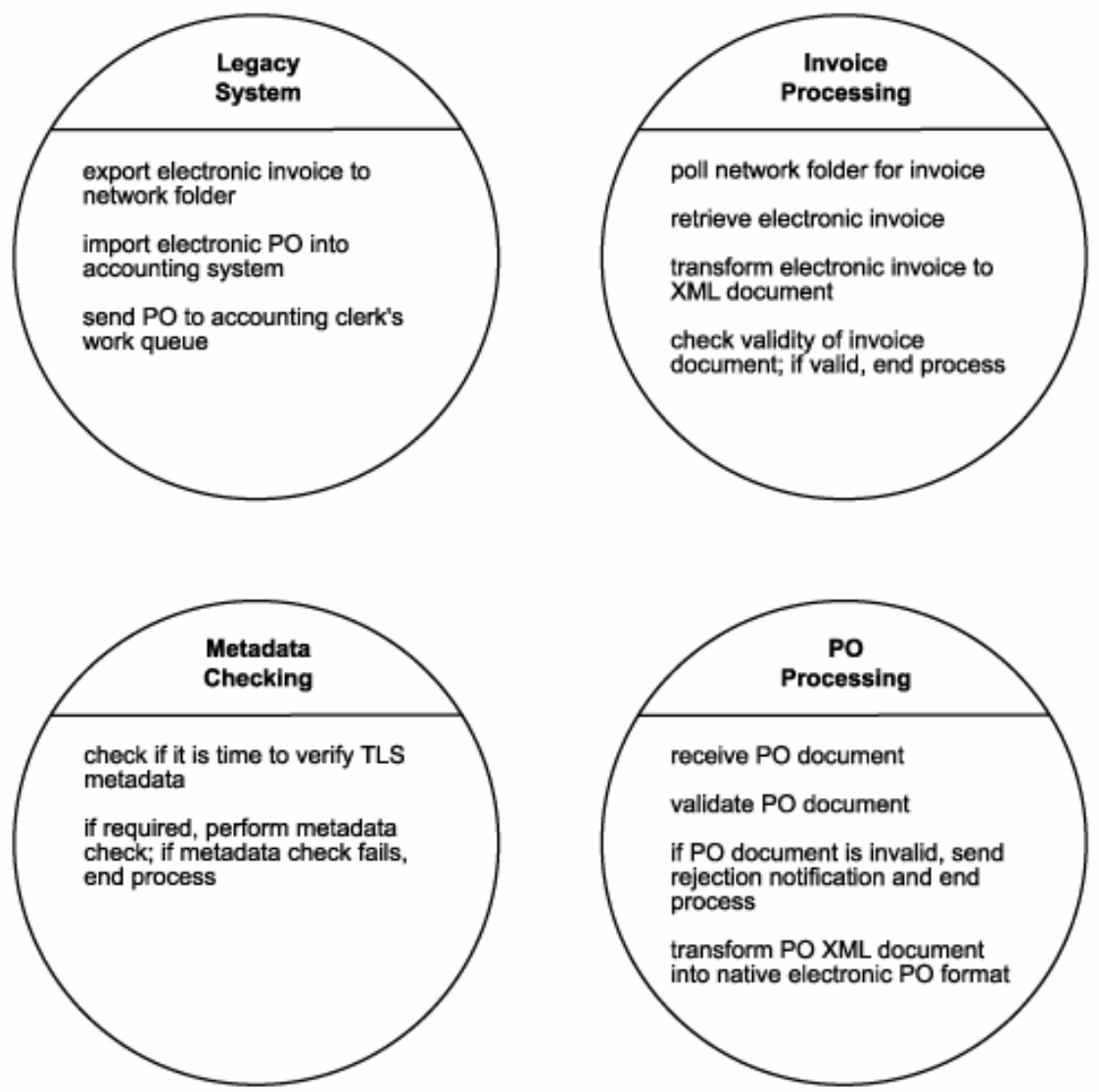


Step 5: Refine and apply principles of service-orientation

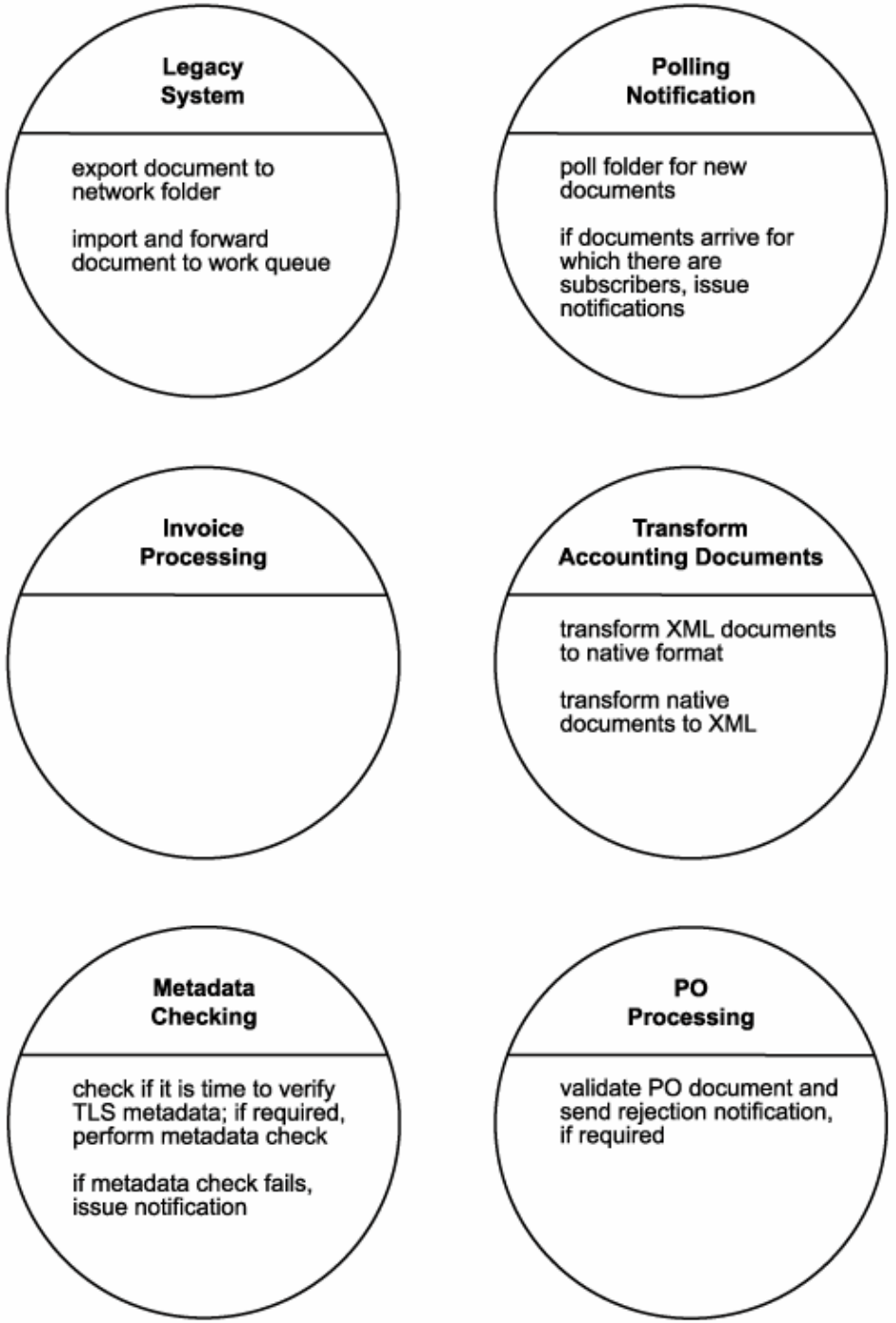
- Input: business service candidates: business service operation candidates grouped into one or more logical contexts
- Output: revised business service candidates (after applying key service-orientation principles: reusability, autonomy)

Input: service candidates

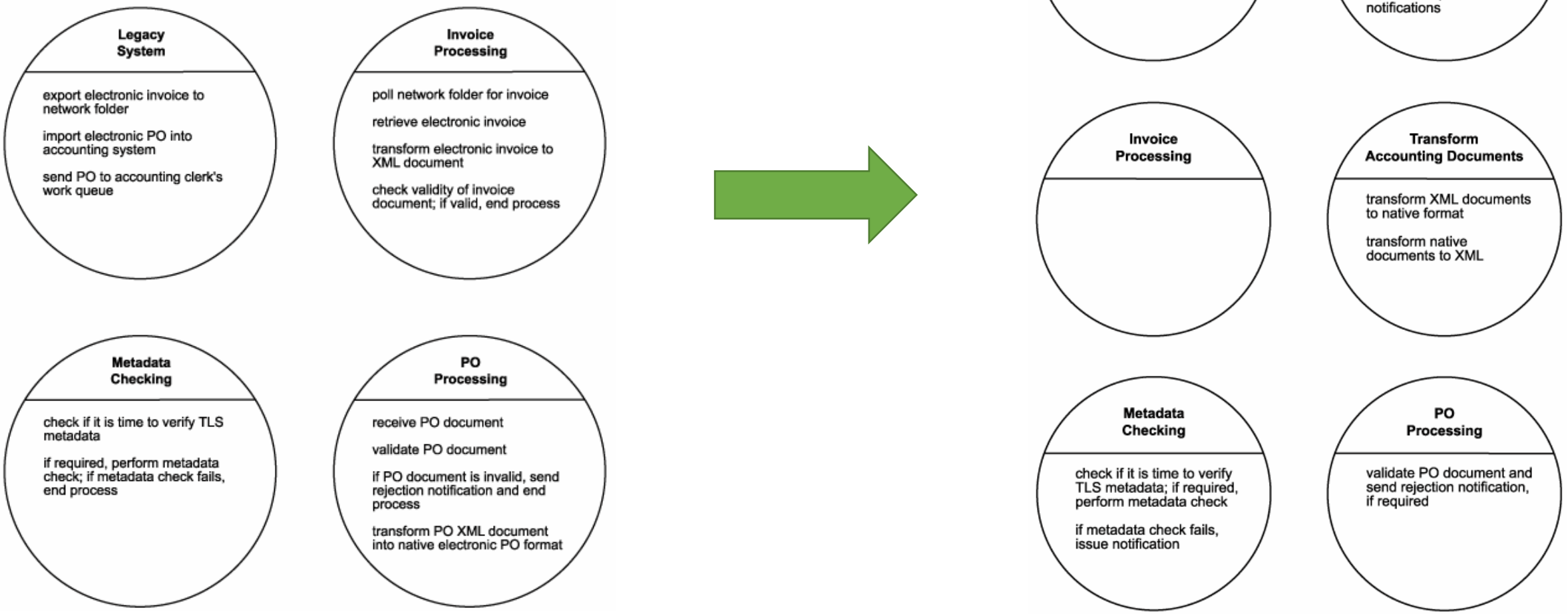
- Legacy System Service
  - Export electronic invoice to network folder.
  - Import electronic PO into accounting system.
  - Send PO to accounting clerk's work queue.
- Invoice Processing Service
  - Poll network folder for invoice.
  - Retrieve electronic invoice.
  - Transform electronic invoice to XML document.
  - Check validity of invoice document. If invalid, end process.
- PO Processing Service
  - Receive PO document.
  - Validate PO document.
  - If PO document is invalid, send rejection notification and end process.
  - Transform PO XML document into native electronic PO format.
- Metadata Checking Service
  - Check if it is time to verify TLS metadata.
  - If required, perform metadata check. If metadata check fails, end process.



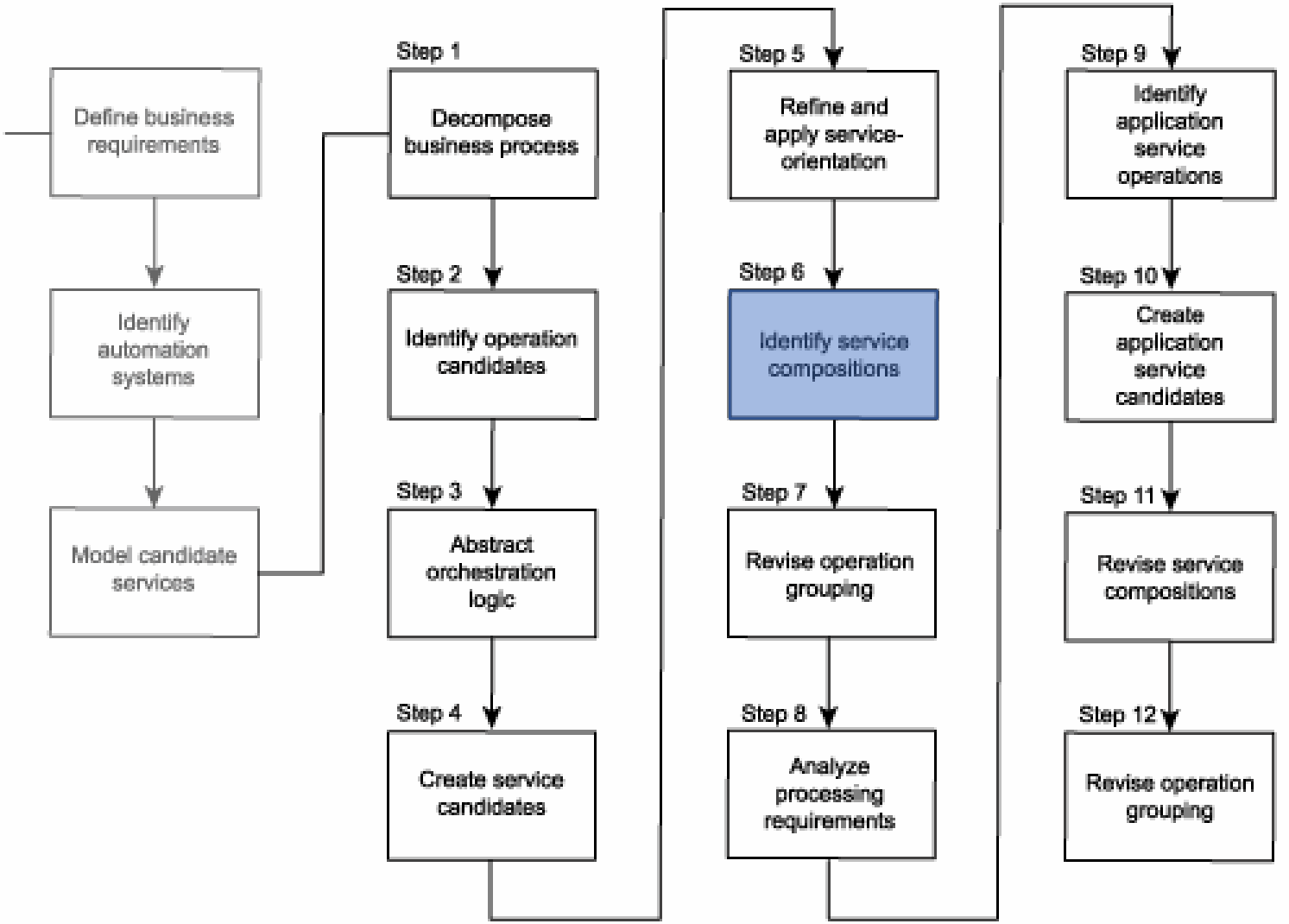
Output: revised service candidates



From service candidates to refined service candidates

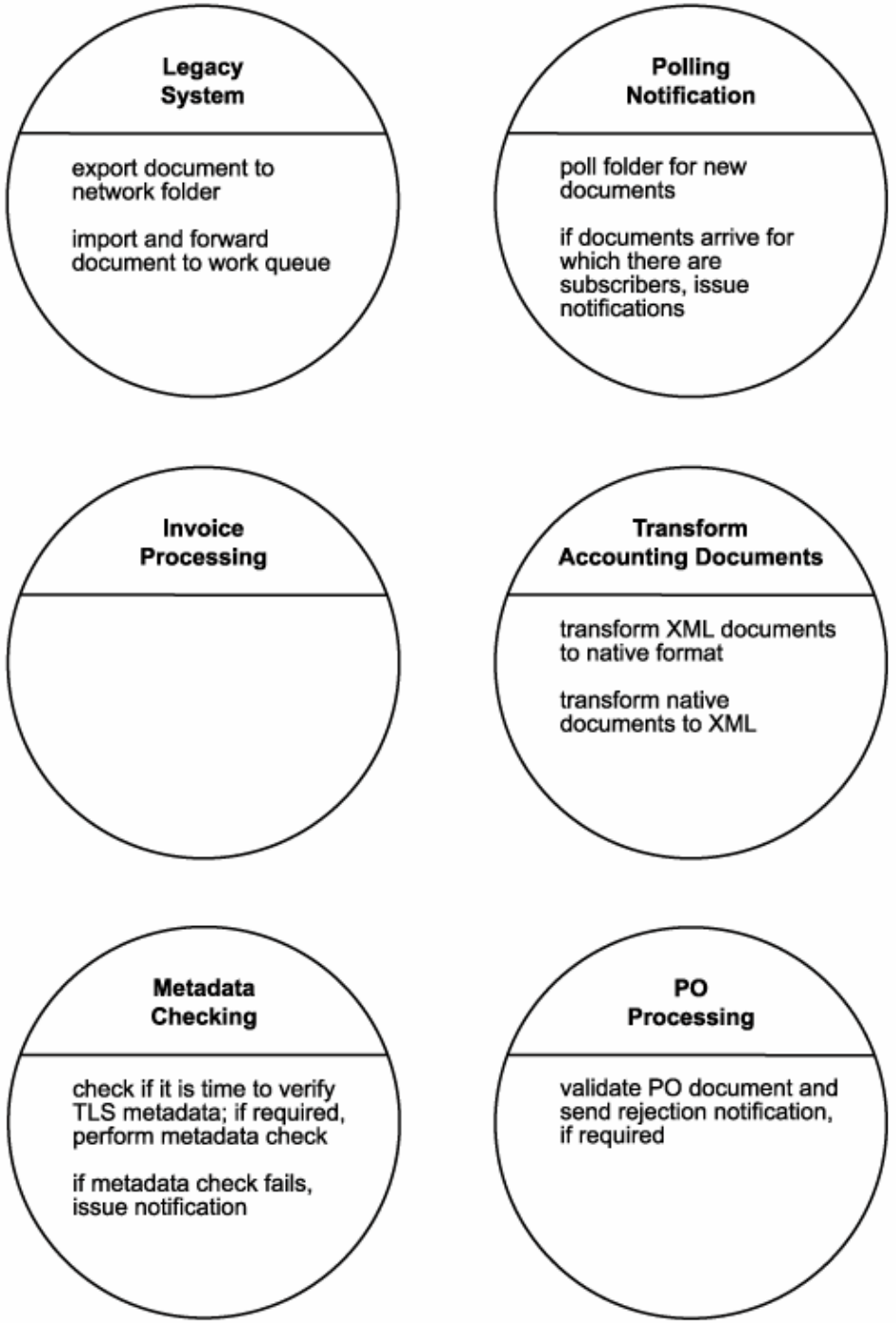


Common steps of modeling process



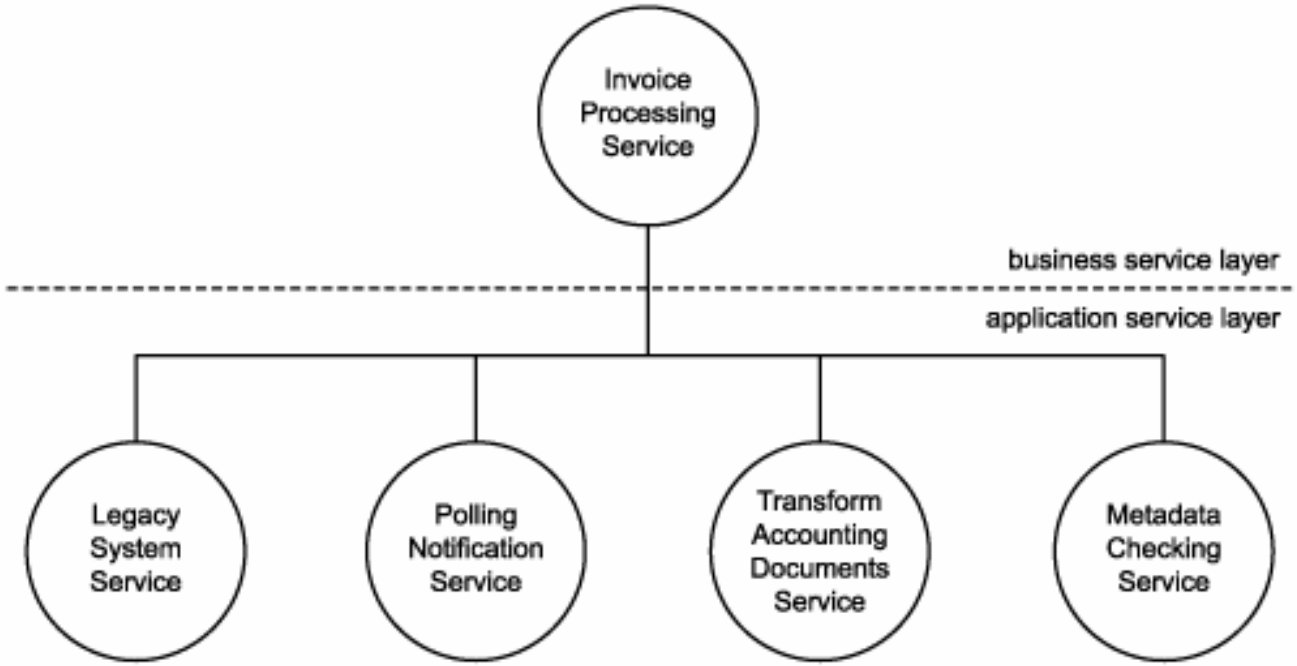
Step 6: Identify candidate service compositions

Input: revised service candidates



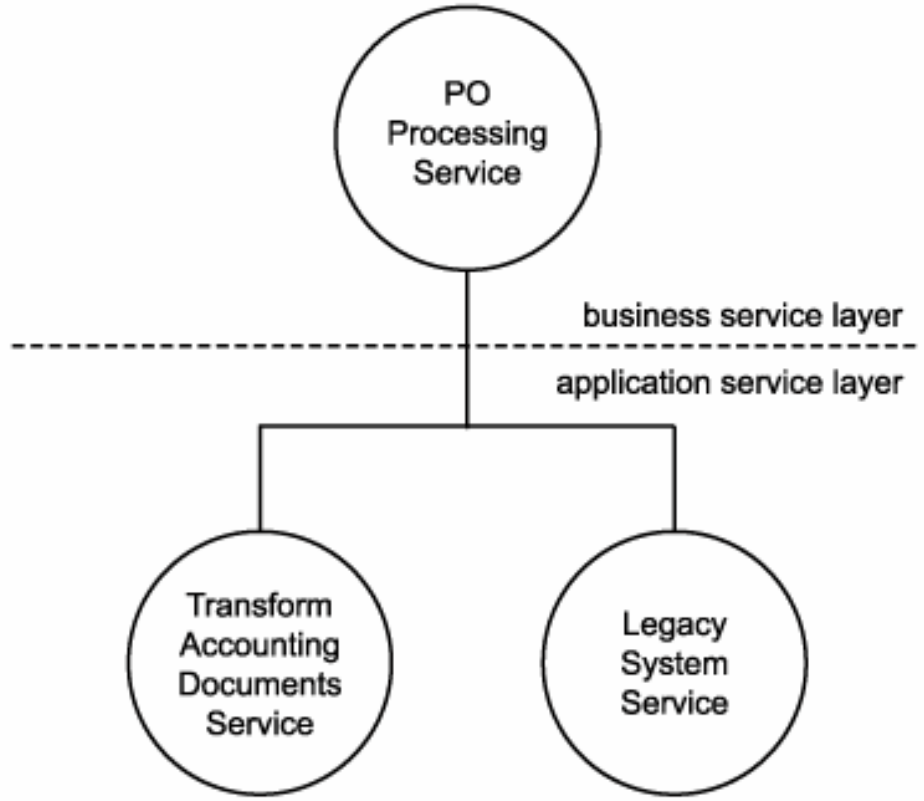
Output: Candidate service compositions

- A sample composition representing the Invoice Submission Process.

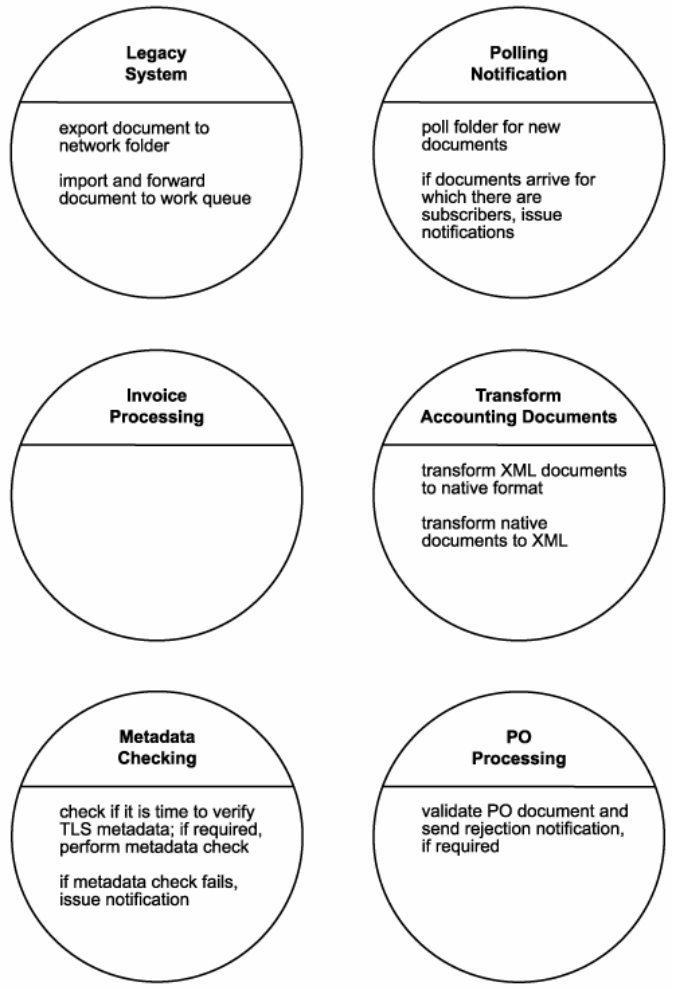


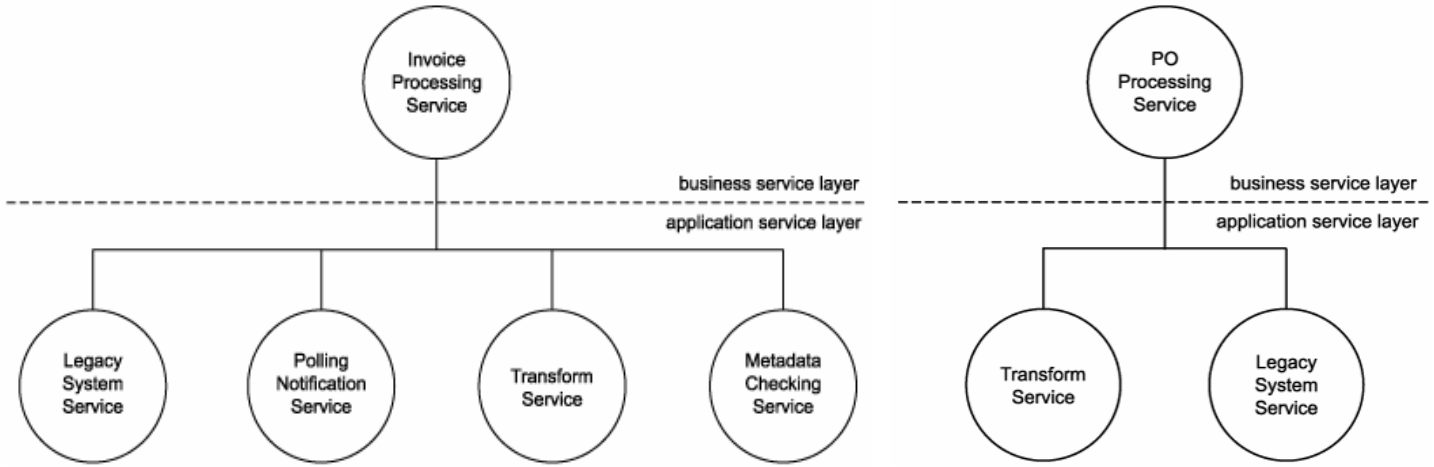
Output: Candidate service compositions

- A sample composition representing the Order Fulfillment Process.

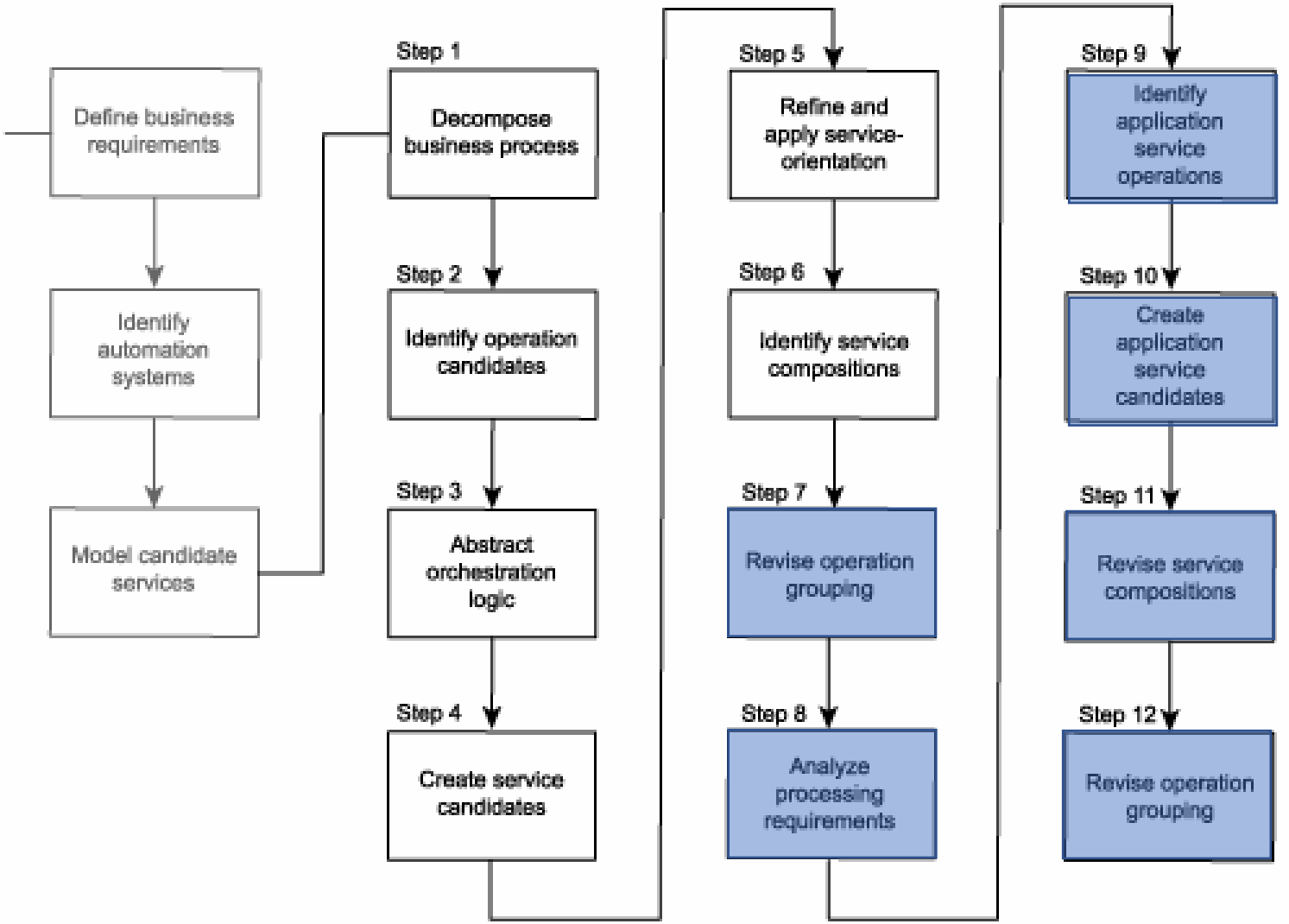


From service candidates to candidate service compositions



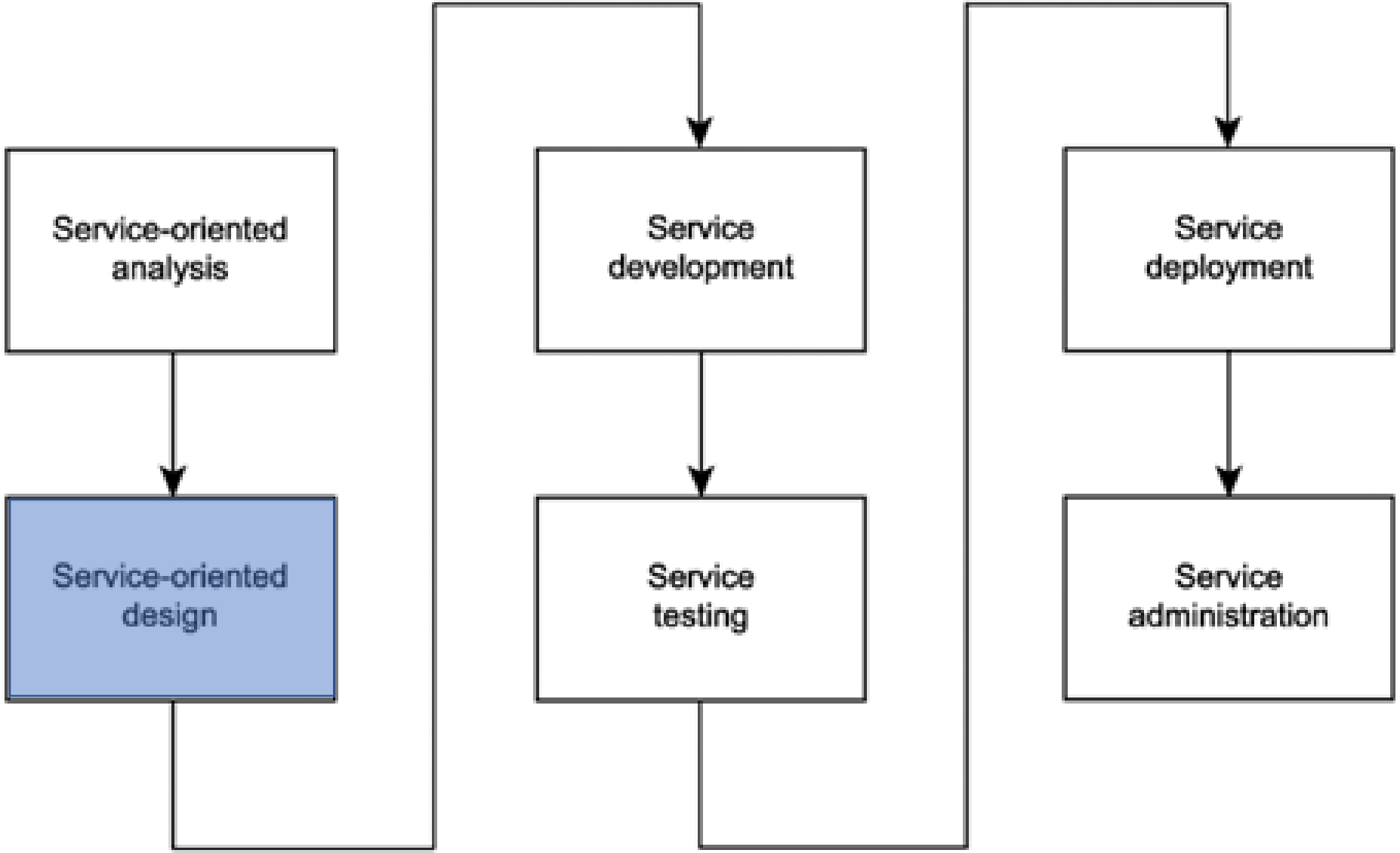


Common steps of modeling process



End of service-oriented analysis review

SOA delivery lifecycle



Service-oriented design

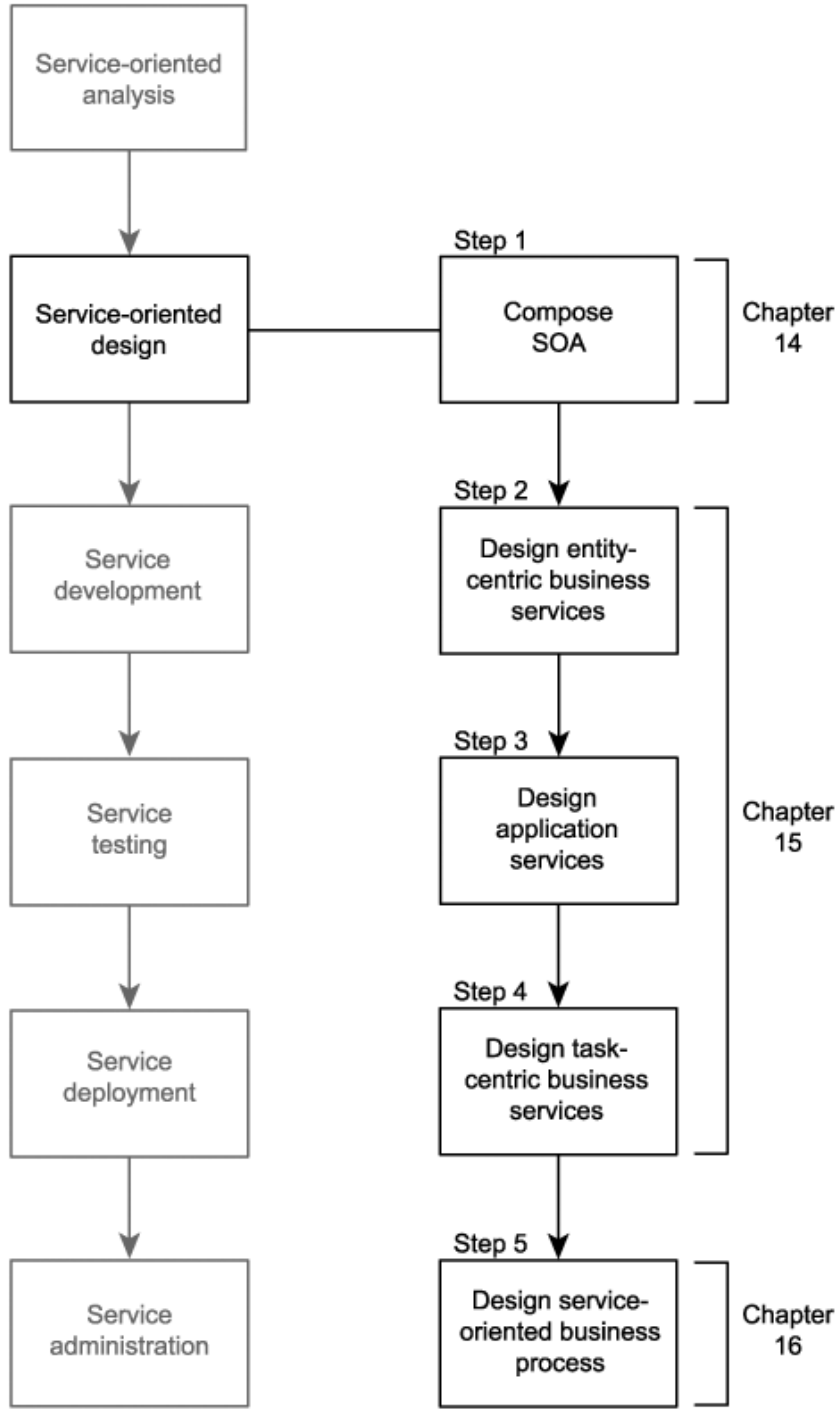
- Service-oriented design is the process by which concrete physical service designs are derived from logical service candidates and then assembled into abstract compositions that implement a business process.

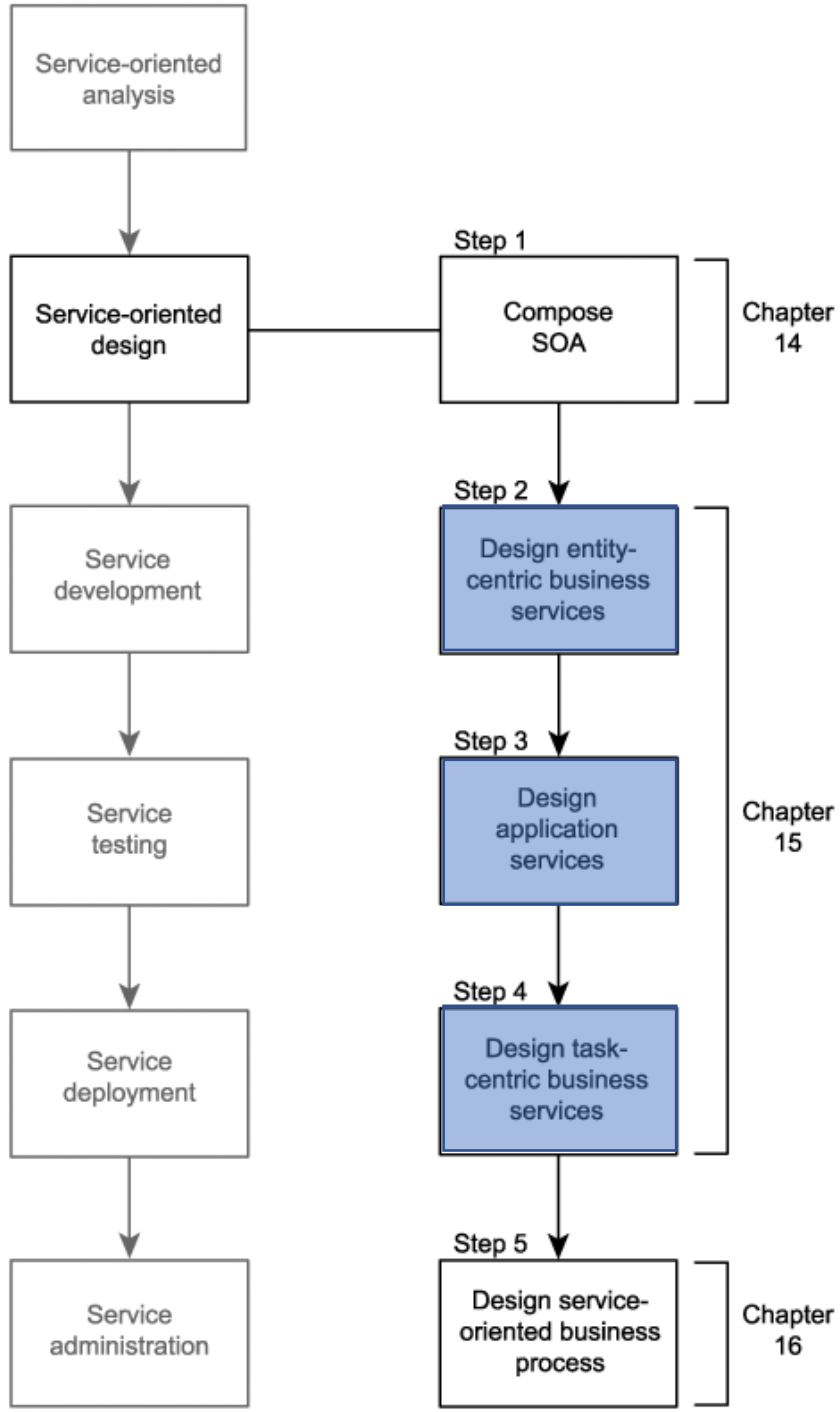
Analysis in this process

- How can physical service interface definitions be derived from the service candidates modeled during the service-oriented analysis phase?
- What SOA characteristics do we want to realize and support?
- What industry standards and extensions will be required by our SOA to implement the planned service designs and SOA characteristics?

Objectives of service-oriented design

- Determine the core set of architectural extensions.
- Set the boundaries of the architecture.
- Identify required design standards.
- Define abstract service interface designs.
- Identify potential service compositions.
- Assess support for service-orientation principles.
- Explore support for characteristics of contemporary SOA.

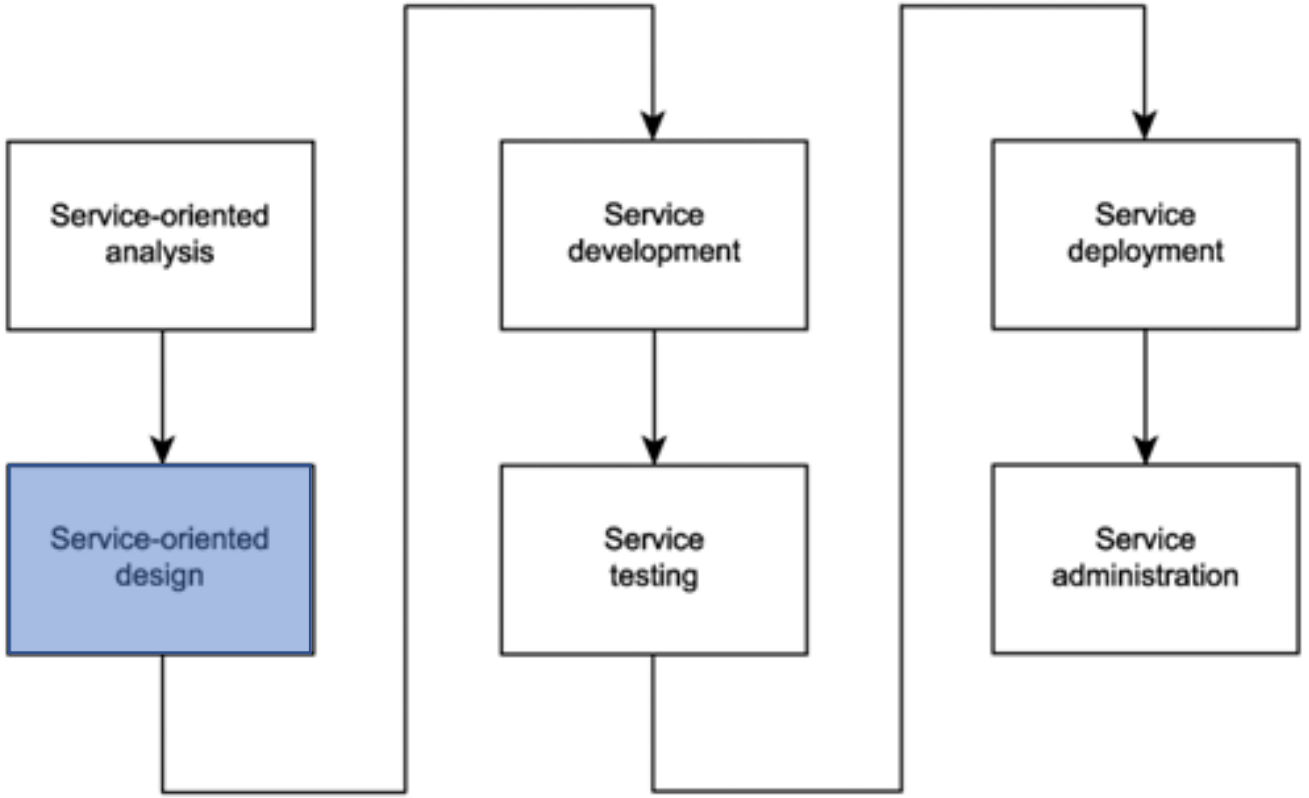

Service-oriented analysis
Service-oriented design
Service development
Service testing
Service deployment
Service administration
Step 1
Compose SOA
Chapter 14
Step 2
Design entity-centric business services
Step 3
Design application services
Step 4
Design task-centric business services
Chapter 15
Step 5
Design service-oriented business process
Chapter 16


Service-oriented analysis
Service-oriented design
Service development
Service testing
Service deployment
Service administration
Step 1
Compose SOA
Chapter 14
Step 2
Design entity-centric business services
Step 3
Design application services
Step 4
Design task-centric business services
Chapter 15
Step 5
Design service-oriented business process
Chapter 16

"WSDL first" Service Design

- Before we can develop a service, we need to have defined the interface of that service - "WSDL first"
- Defining the service interface prior to development is important to establishing a highly standardized service-oriented architecture and required to realize a number of the characteristics we identified as being part of contemporary SOA

Who should participate in this step?



Open Question Points: 10

Setting

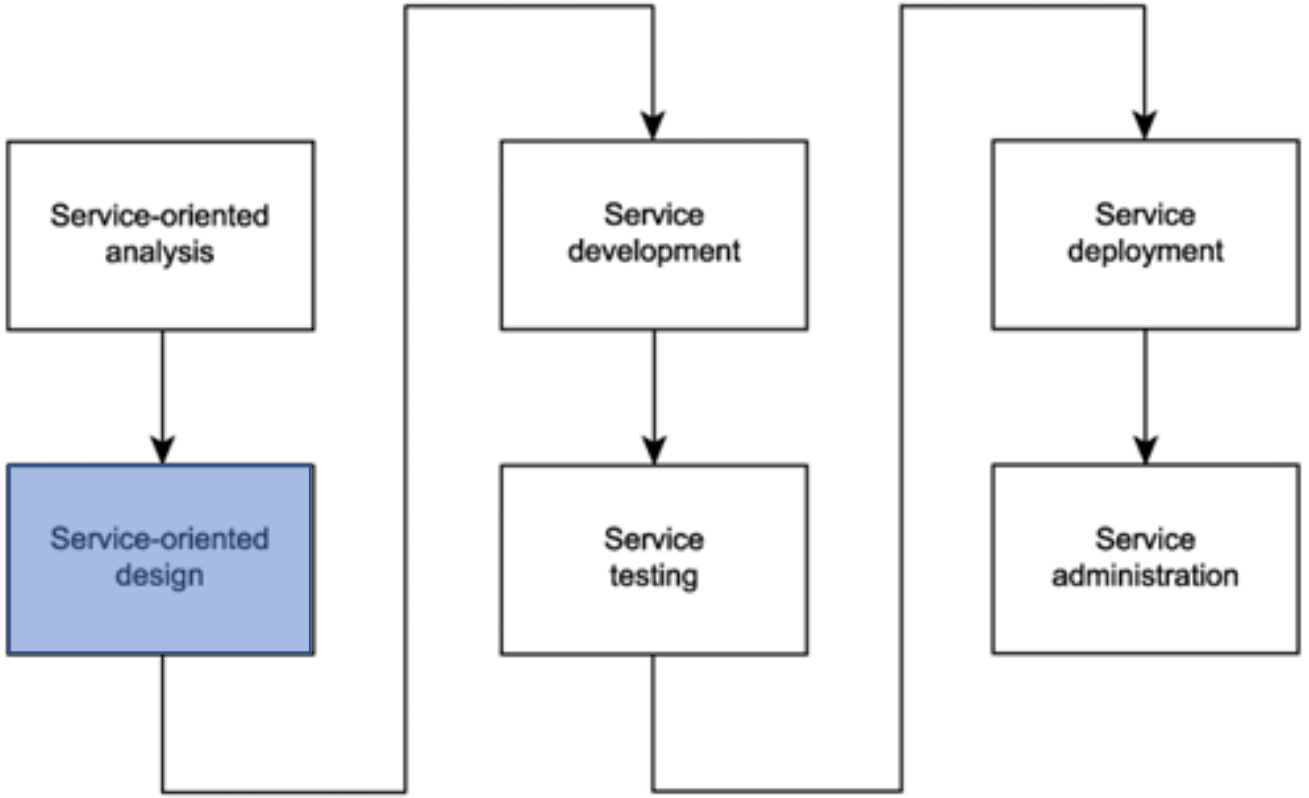
What kind of expertise is required to define the above documents?

Open Question is only supported on Version 2.0 or newer.

Answer

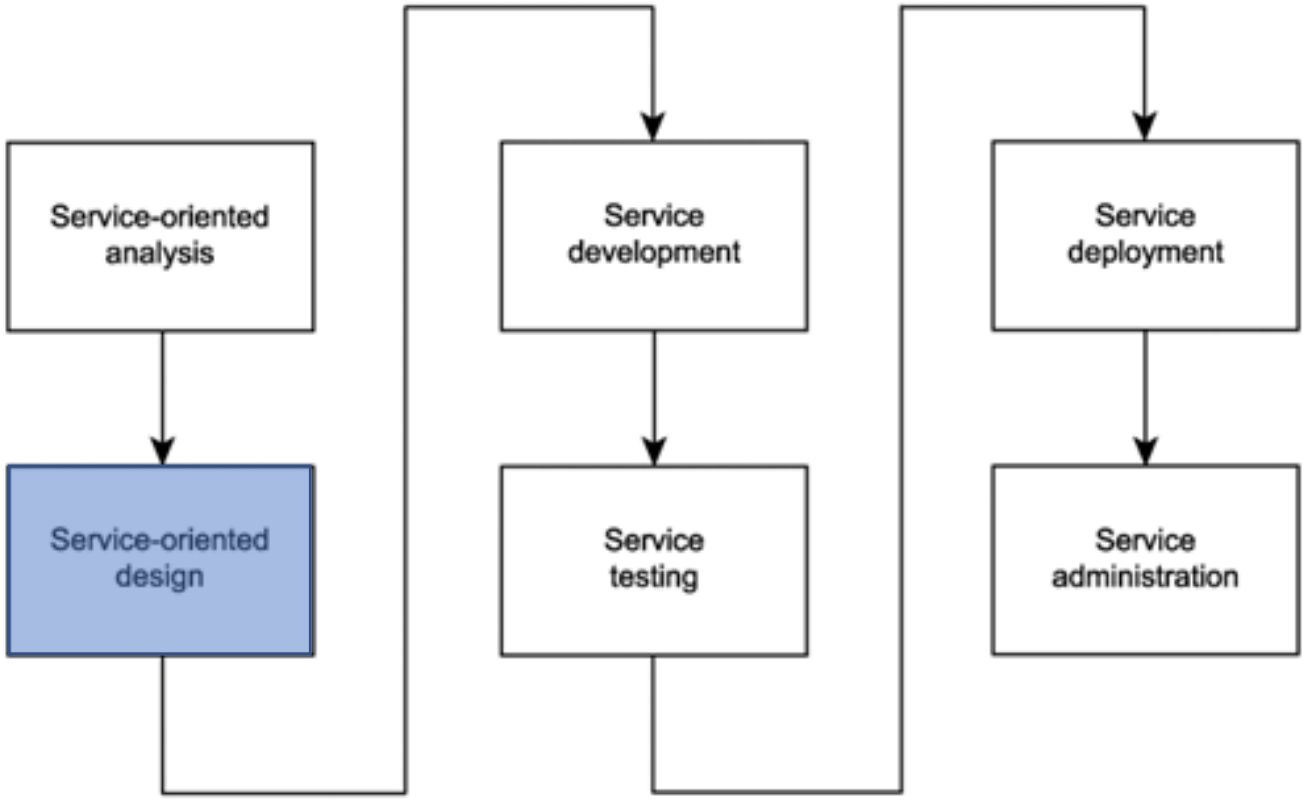
Who should participate in this step?

- Best defined by those who understand the enterprise technical environments the most



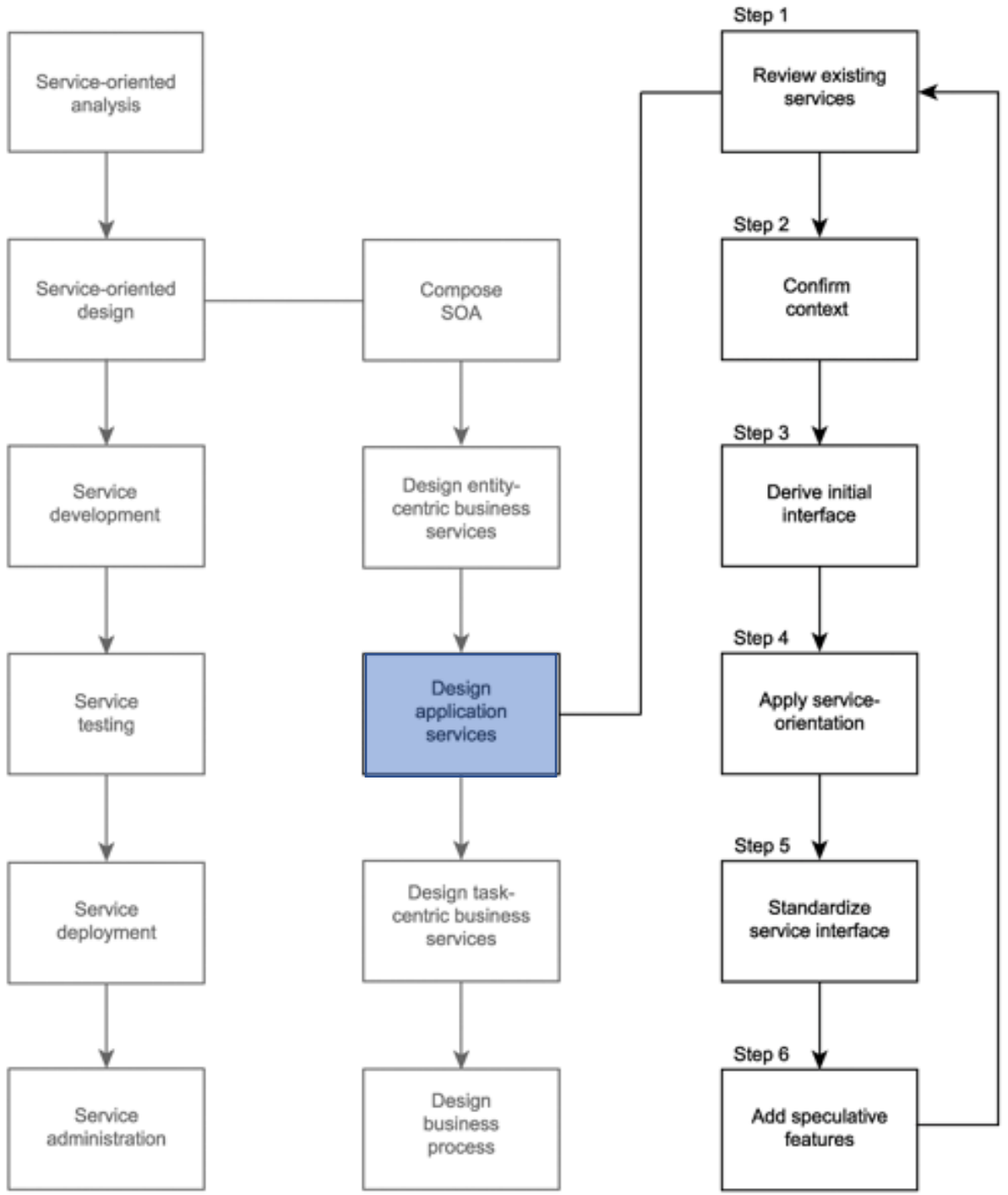
Who should participate in this step?

- Best defined by those who understand the enterprise technical environments the most
- Business analysis expertise not required

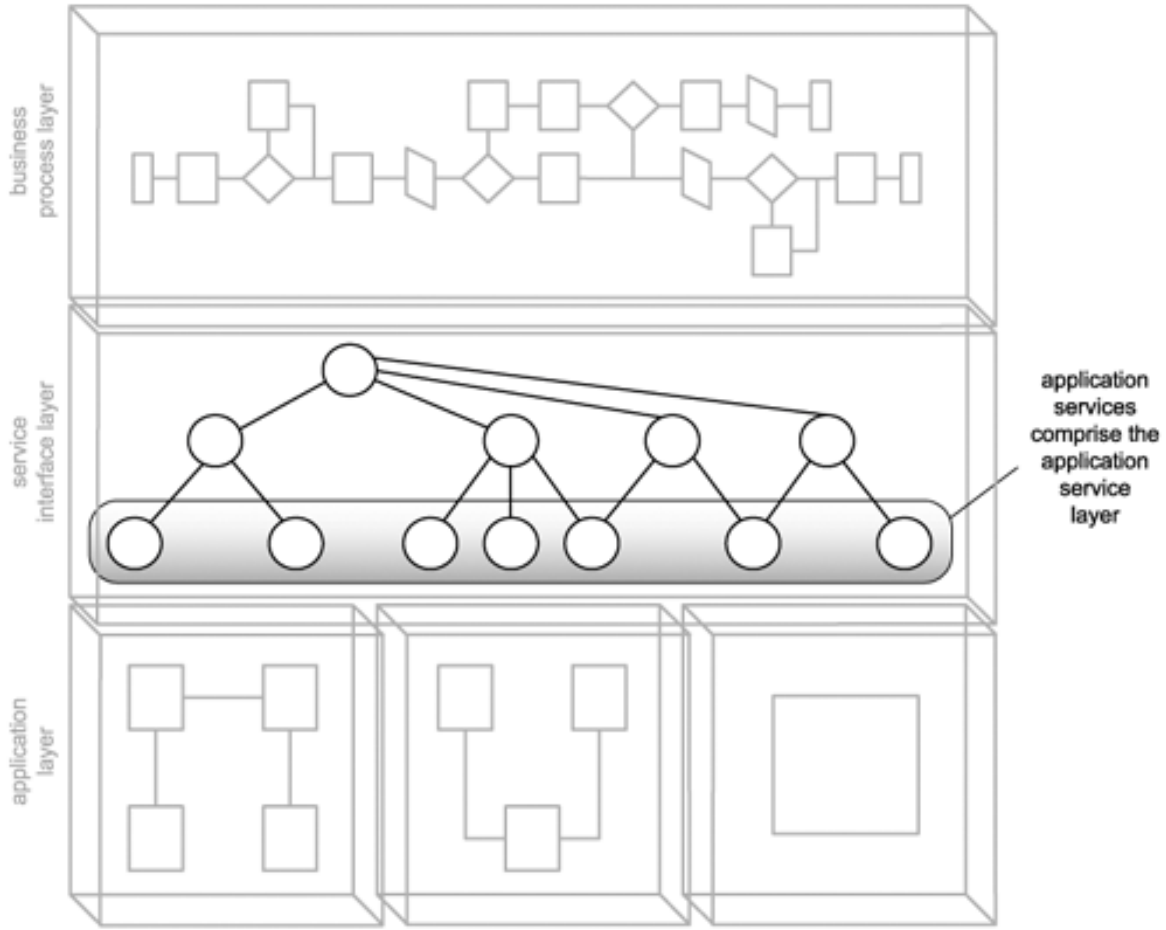


Service-oriented design

Design application services



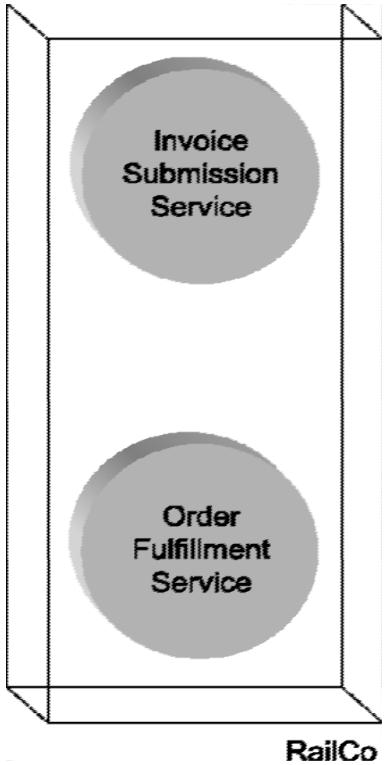
Design application services



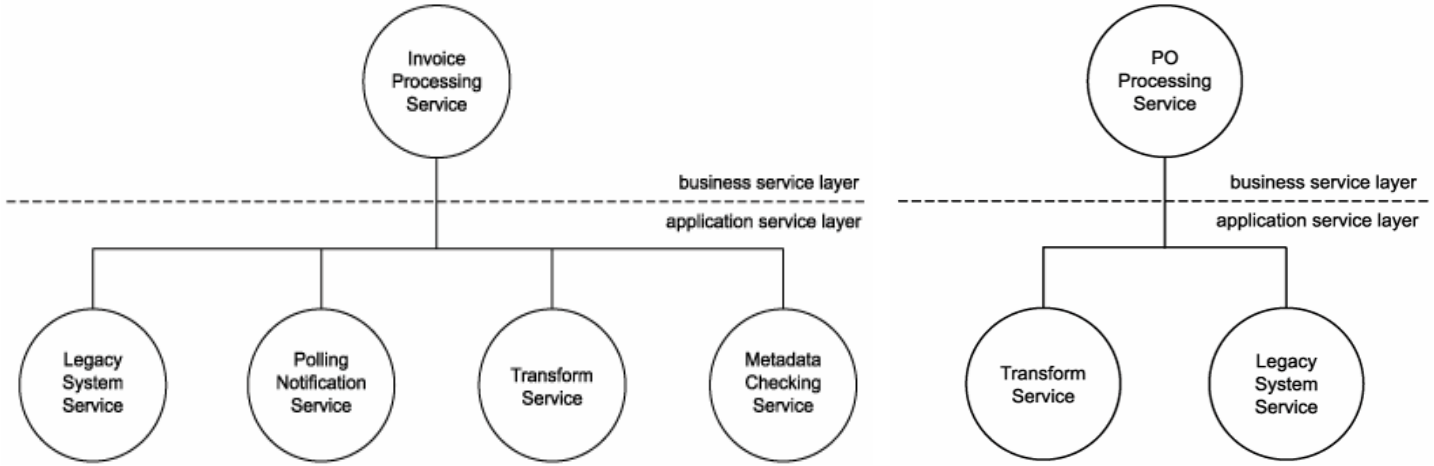
The design of the Transform application service candidate

Recall RailCo's service candidates

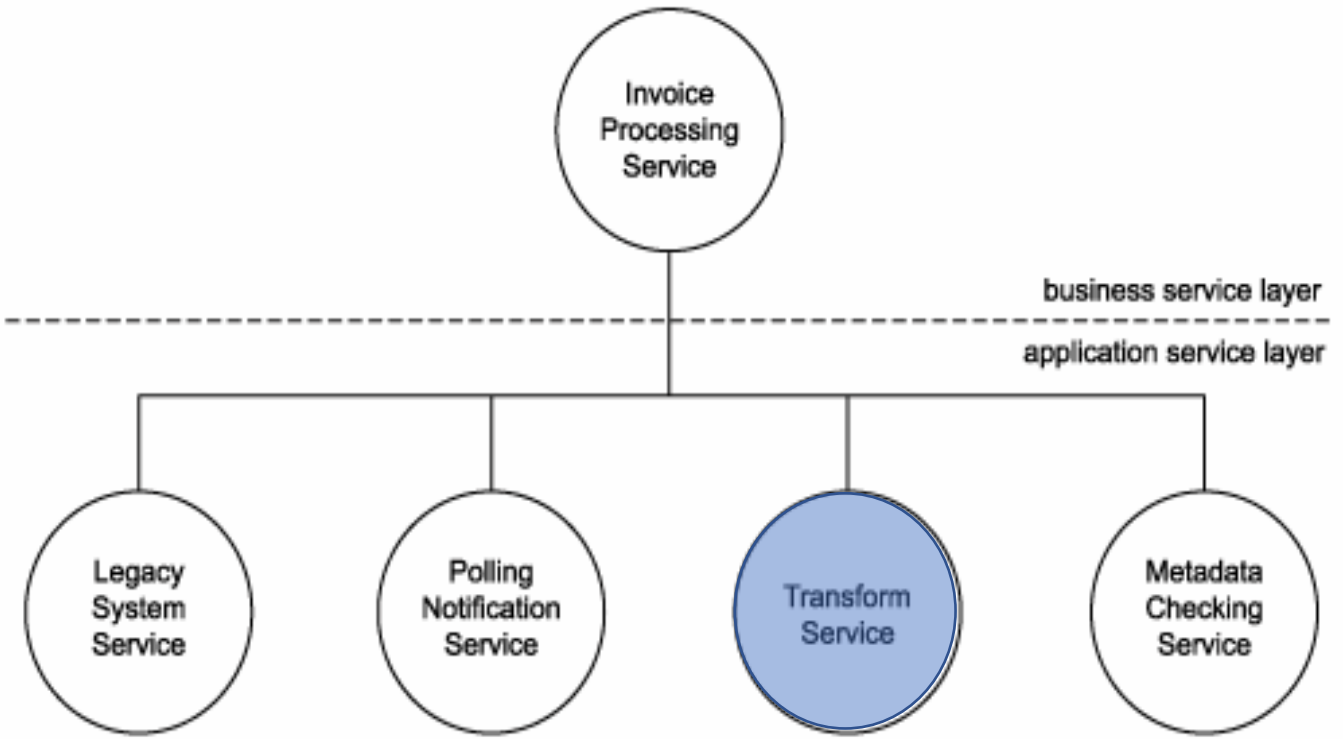
- Old design

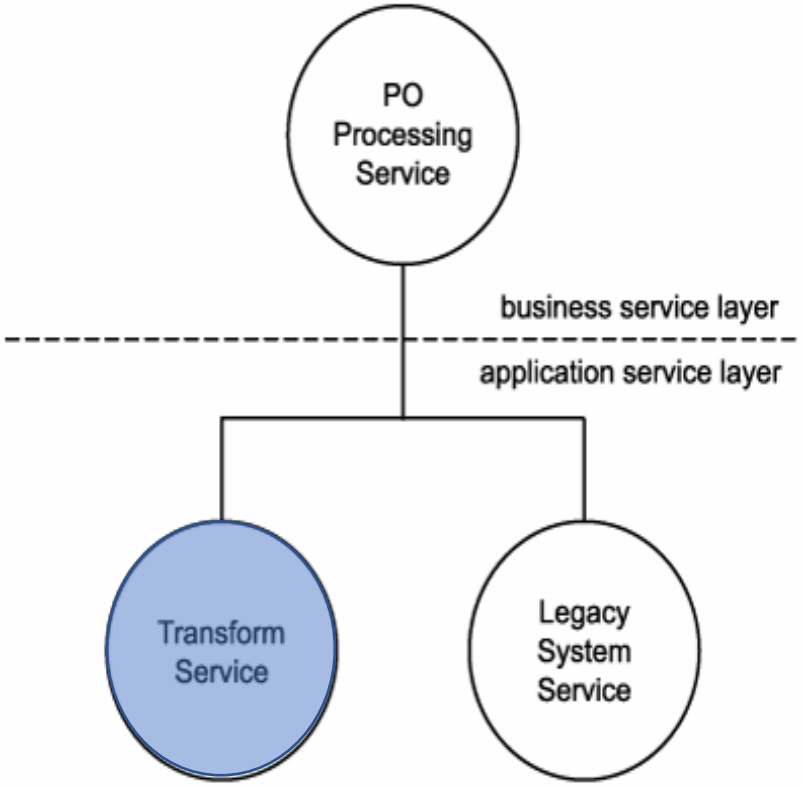


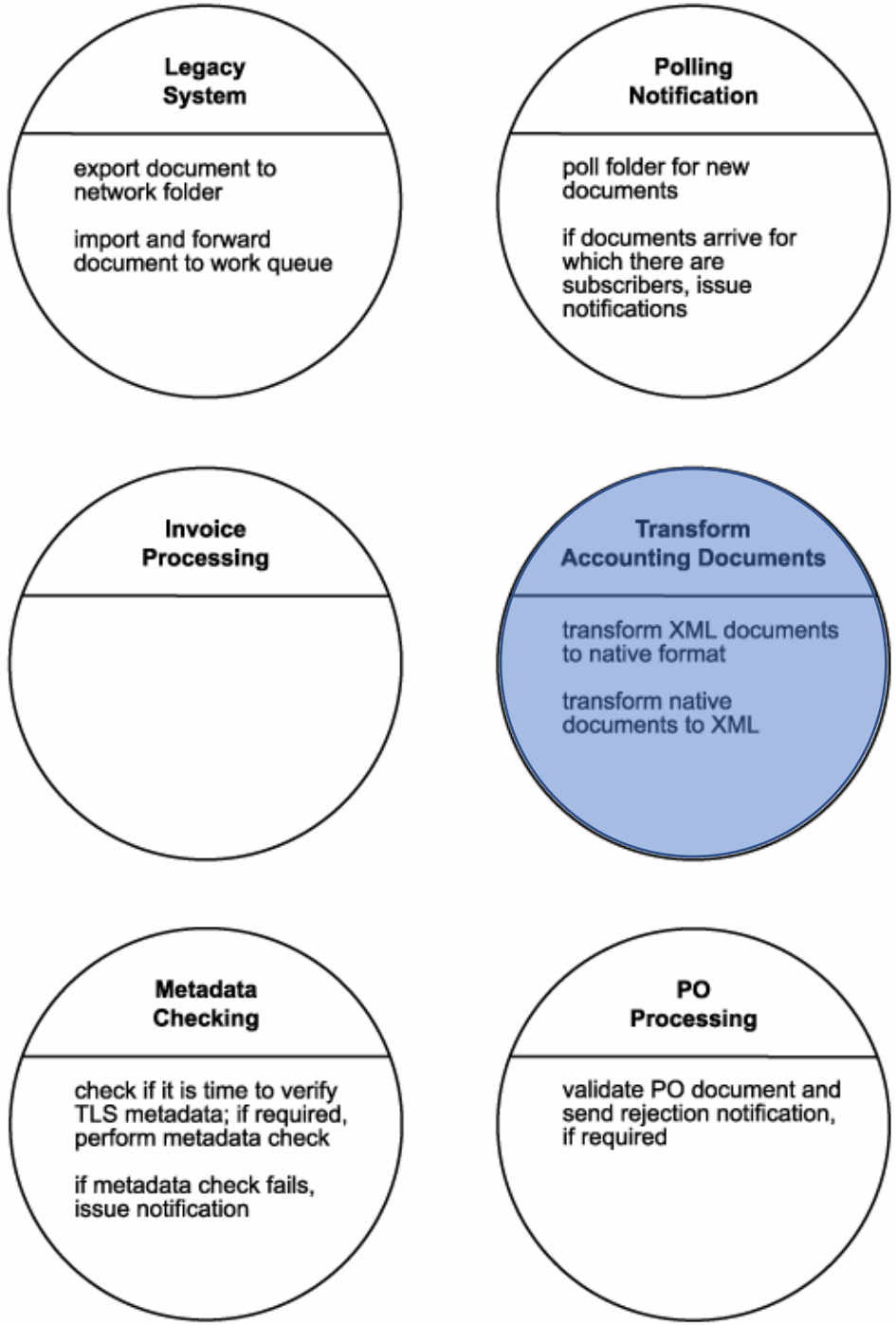
- New service candidates



Transform Service

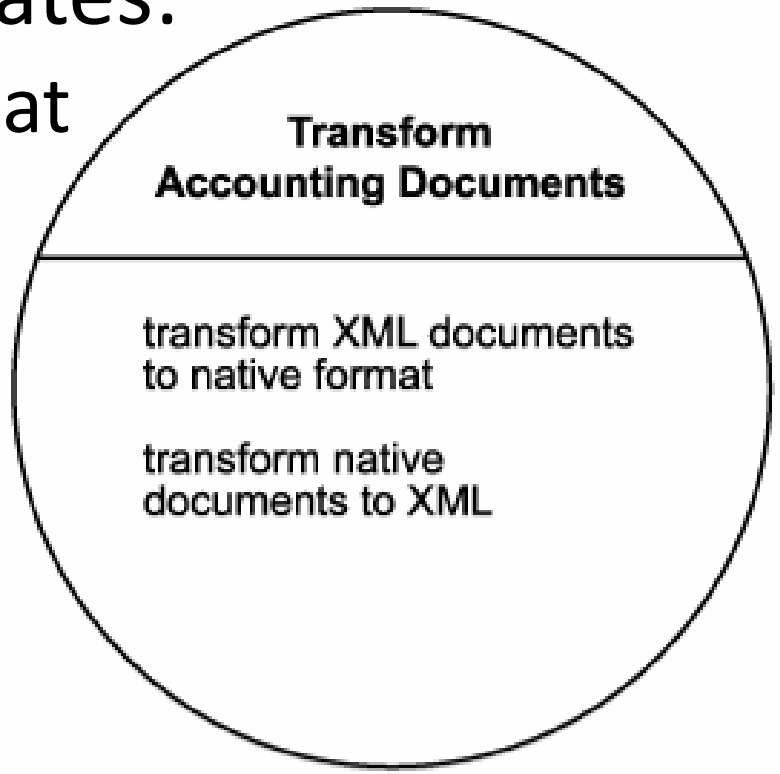





Legacy System
export document to network folder
import and forward document to work queue
Polling Notification
poll folder for new documents
if documents arrive for which there are subscribers, issue notifications
Invoice Processing
Transform Accounting Documents
transform XML documents to native format
transform native documents to XML
Metadata Checking
check if it is time to verify TLS metadata; if required, perform metadata check
if metadata check fails, issue notification
PO Processing
validate PO document and send rejection notification, if required

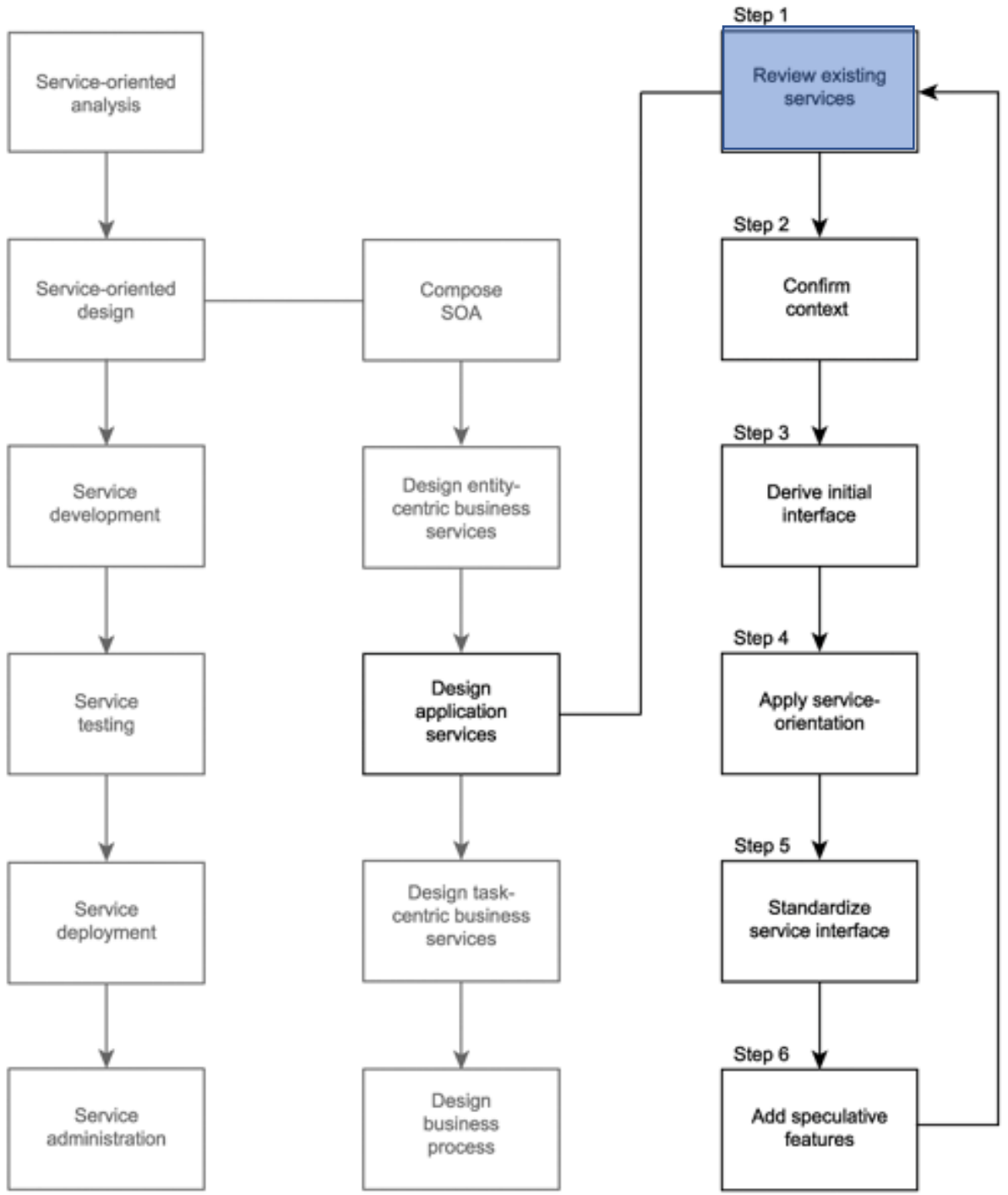
Service Design - Transform Service

- This candidate establishes a "document transformation context," which justifies the grouping of its two very similar operation candidates:
  - transform XML documents to native format
  - transform native documents to XML



- Today we will study some design decisions that RailCo has made when developing this service

Service-oriented design



Step 1: Review existing services

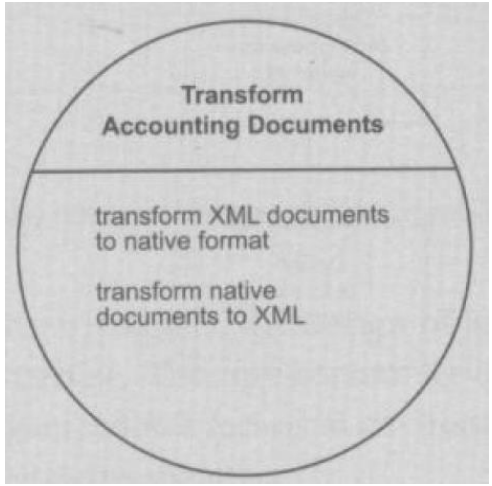
- Check existing services
  for redundant features.

Step 1: Review existing services

- Check existing services for redundant features.
- Check if the required feature can be purchased or leased from vendors.

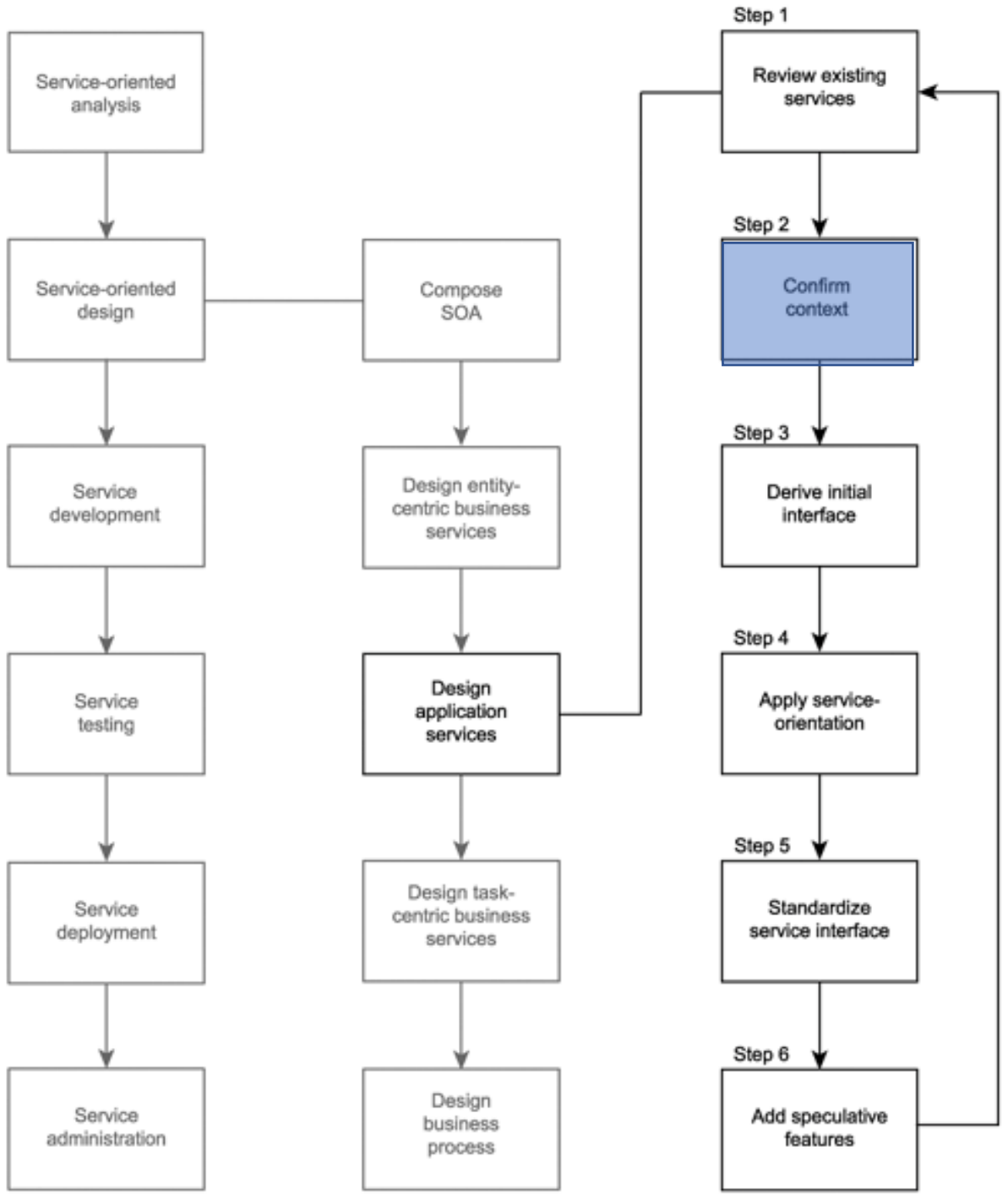
Step 1: Review existing services

- Check existing services for redundant features.
- Check if the required feature can be purchased or leased from vendors.



- Checking names, service descriptions and metadata of existing services (i.e. TLS Subscription Service), RailCo concludes that no overlap exists

Service-oriented design



Step 2: Confirm context

- Reassess service context if operation candidate grouping from the analysis phase is appropriate, e.g. operations may better belong in other services.

Discussion

- Why there is a need for the design phase to reassess what was already decided during the analysis phase?

Open Question Points: 10

Setting

Why there is a need for the design phase to reassess what was already decided during the analysis phase?

Open Question is only supported on Version 2.0 or newer.

Answer

Step 2: Confirm context

- Reassess service context if operation candidate grouping from the analysis phase is appropriate, e.g. operations may better belong in other services.
- Review of Transform Accounting Documents service against TLS Subscription service confirms that grouping context is valid.