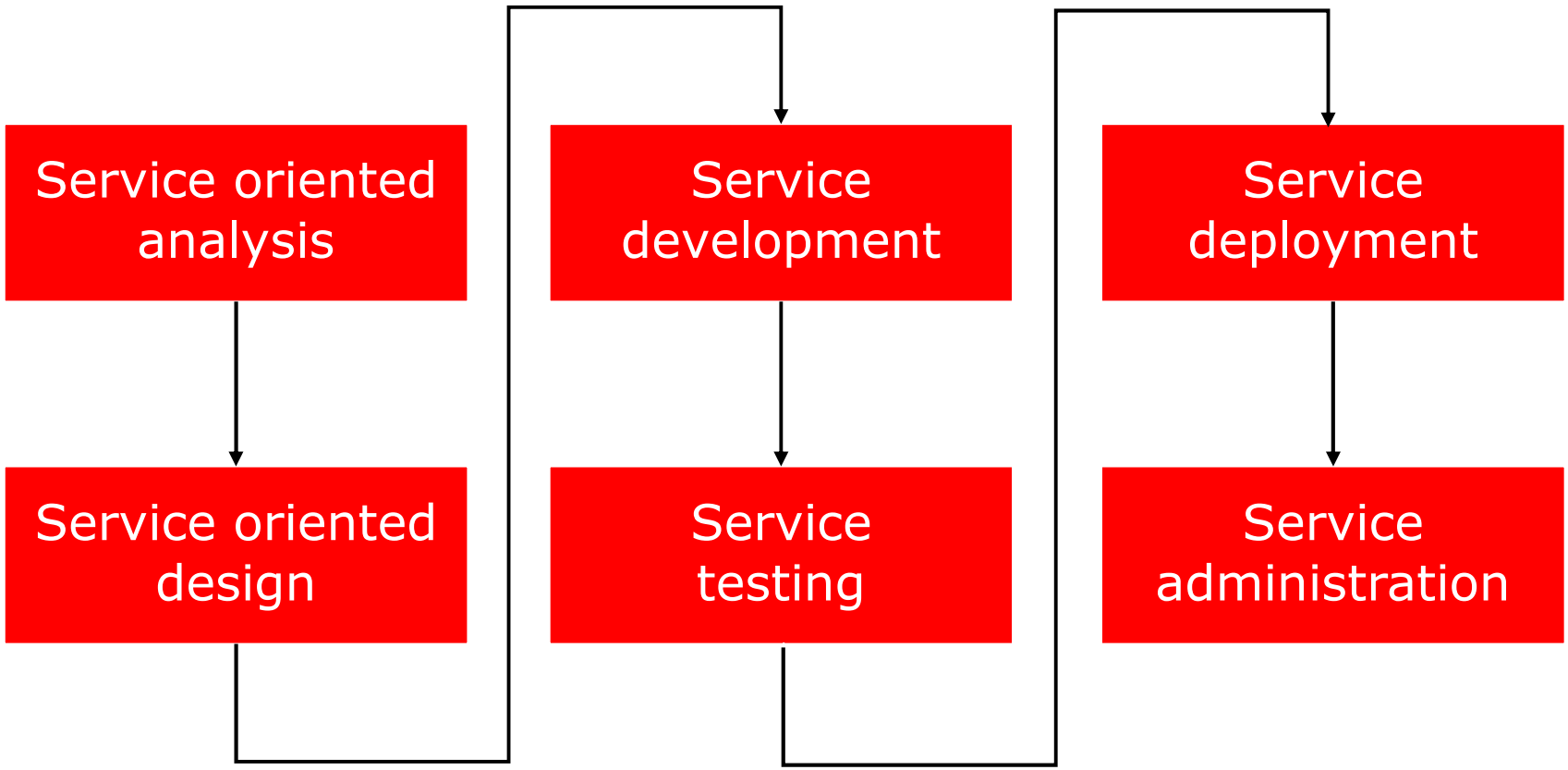
Service Oriented Analysis and Design

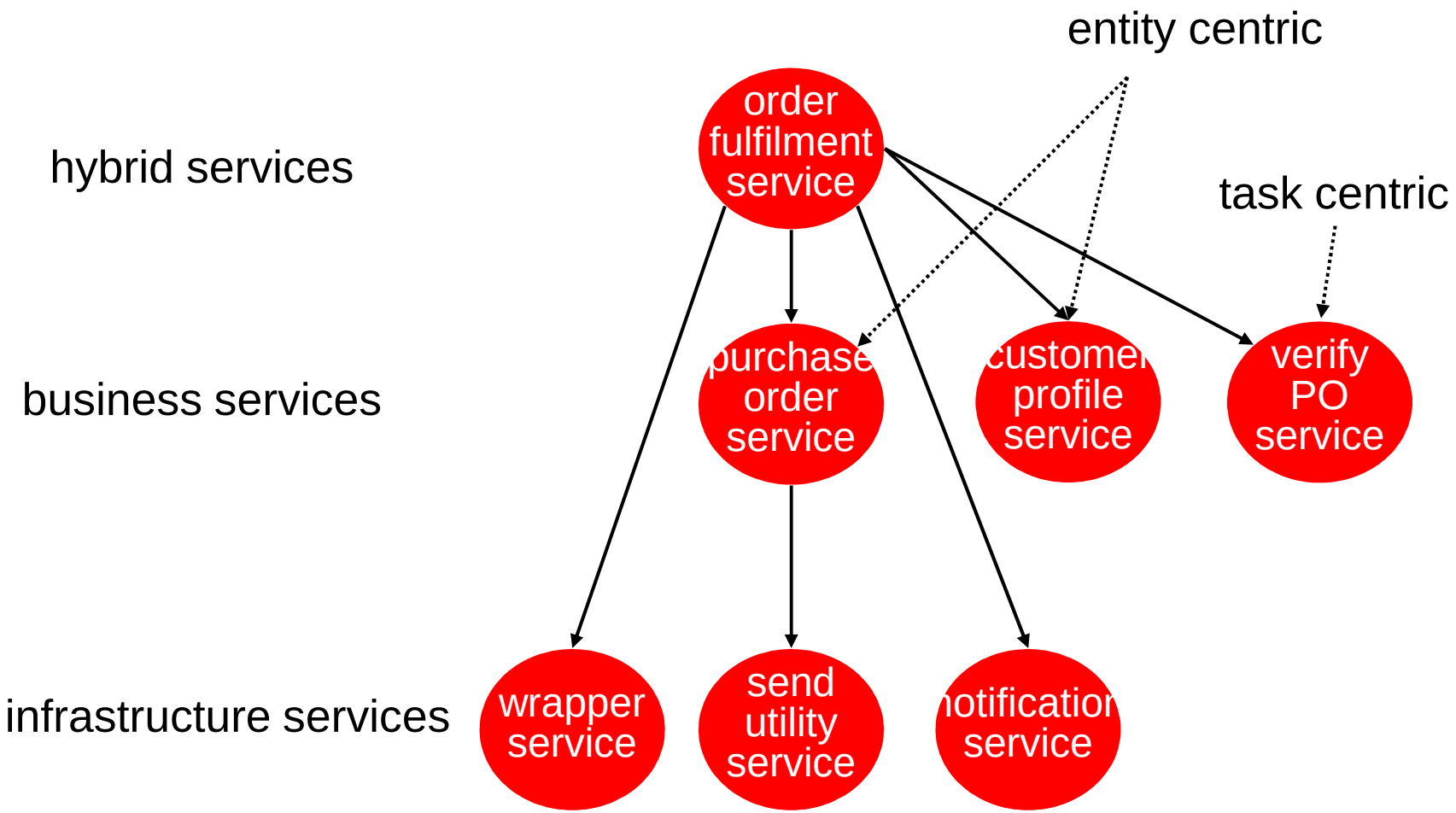
SOSE: Service-Oriented Software Engineering - life cycle



Terminology

- service oriented environment (or service oriented *ecosystem)*
- *business process + supporting services*
  - *application (infrastructure) service*
  - *business service*
    - *Task-centric business service*
    - *Entity-centric business service*
  - *hybrid service*

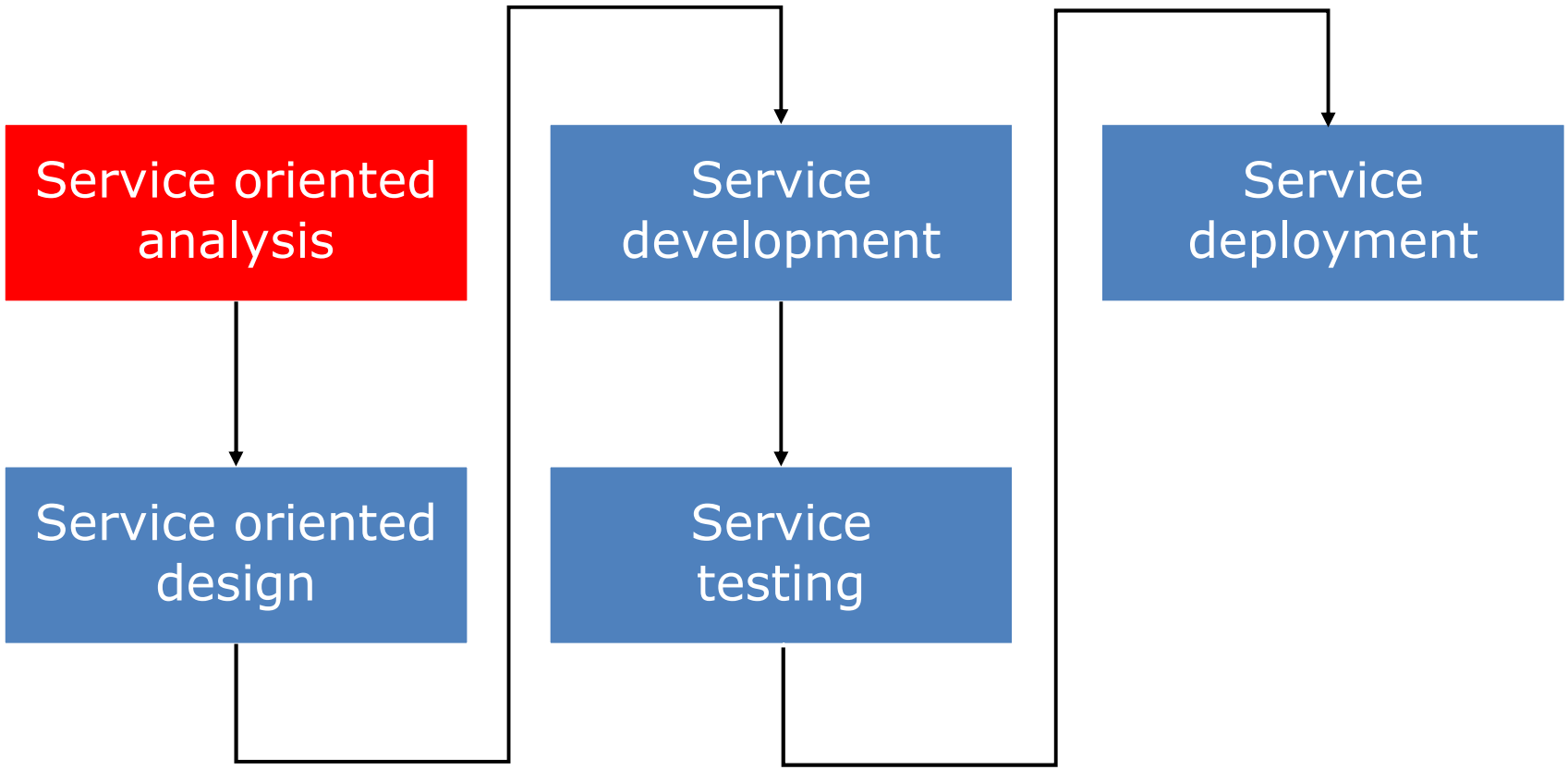
Terminology



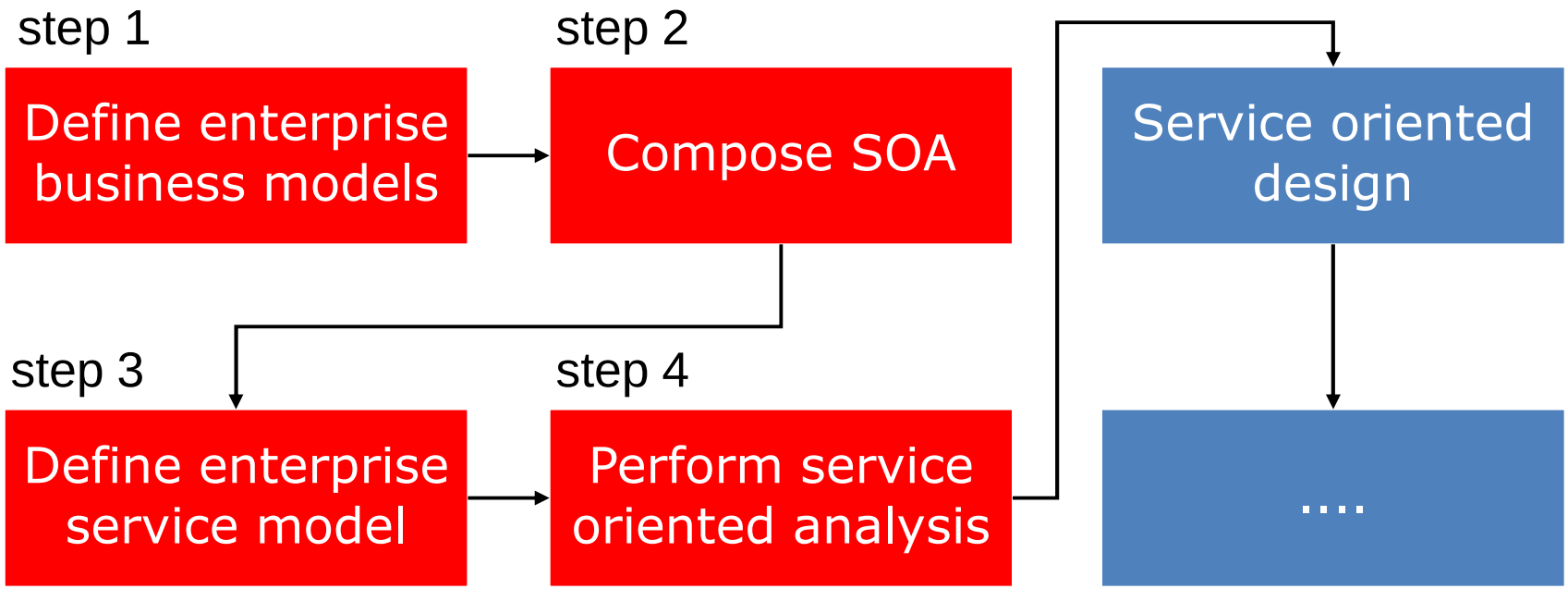
Strategies for life cycle organization

- Top-down strategy
- Bottom-up strategy
- Agile strategy

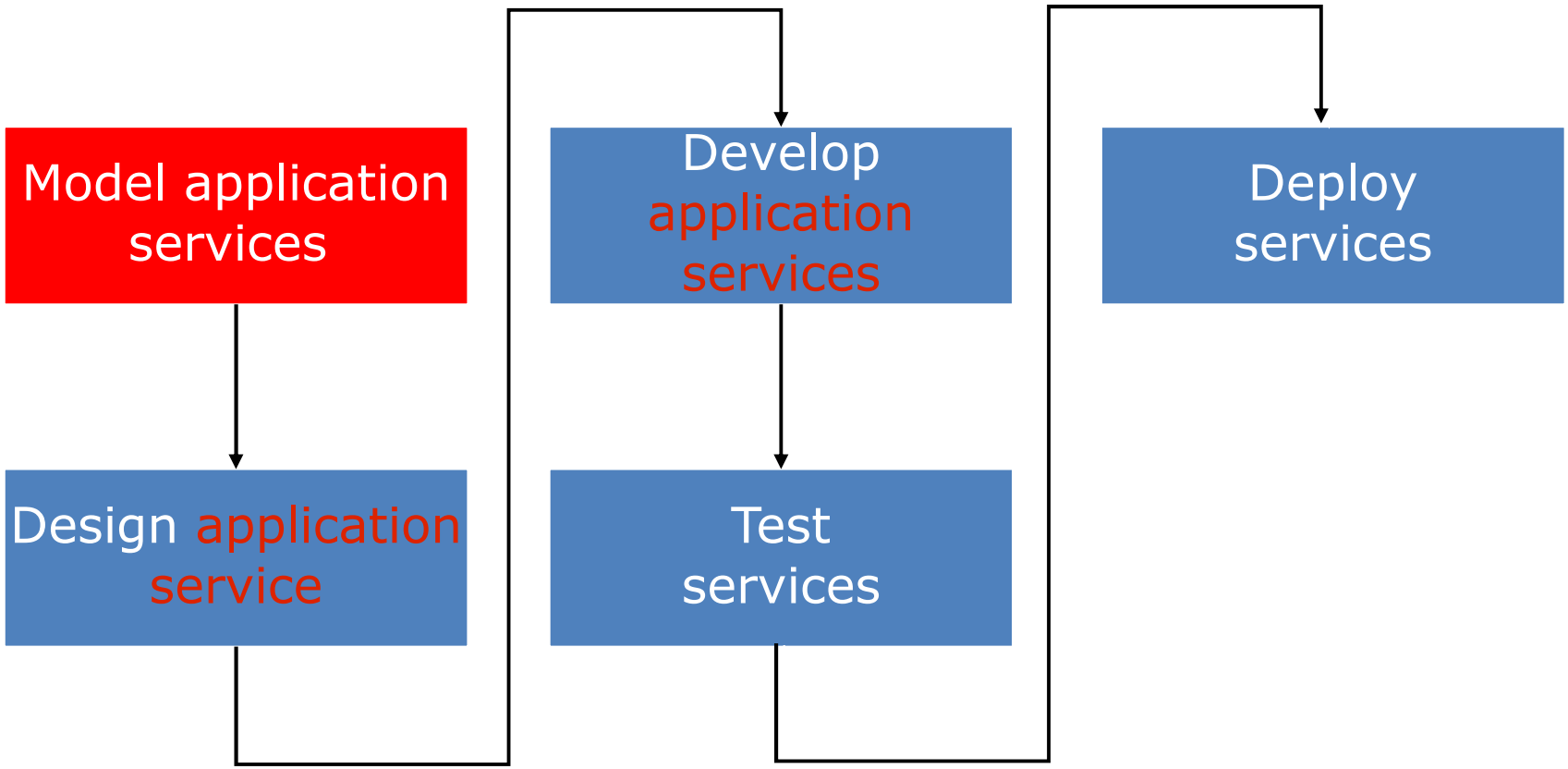
Top-down strategy



Top-down SO analysis

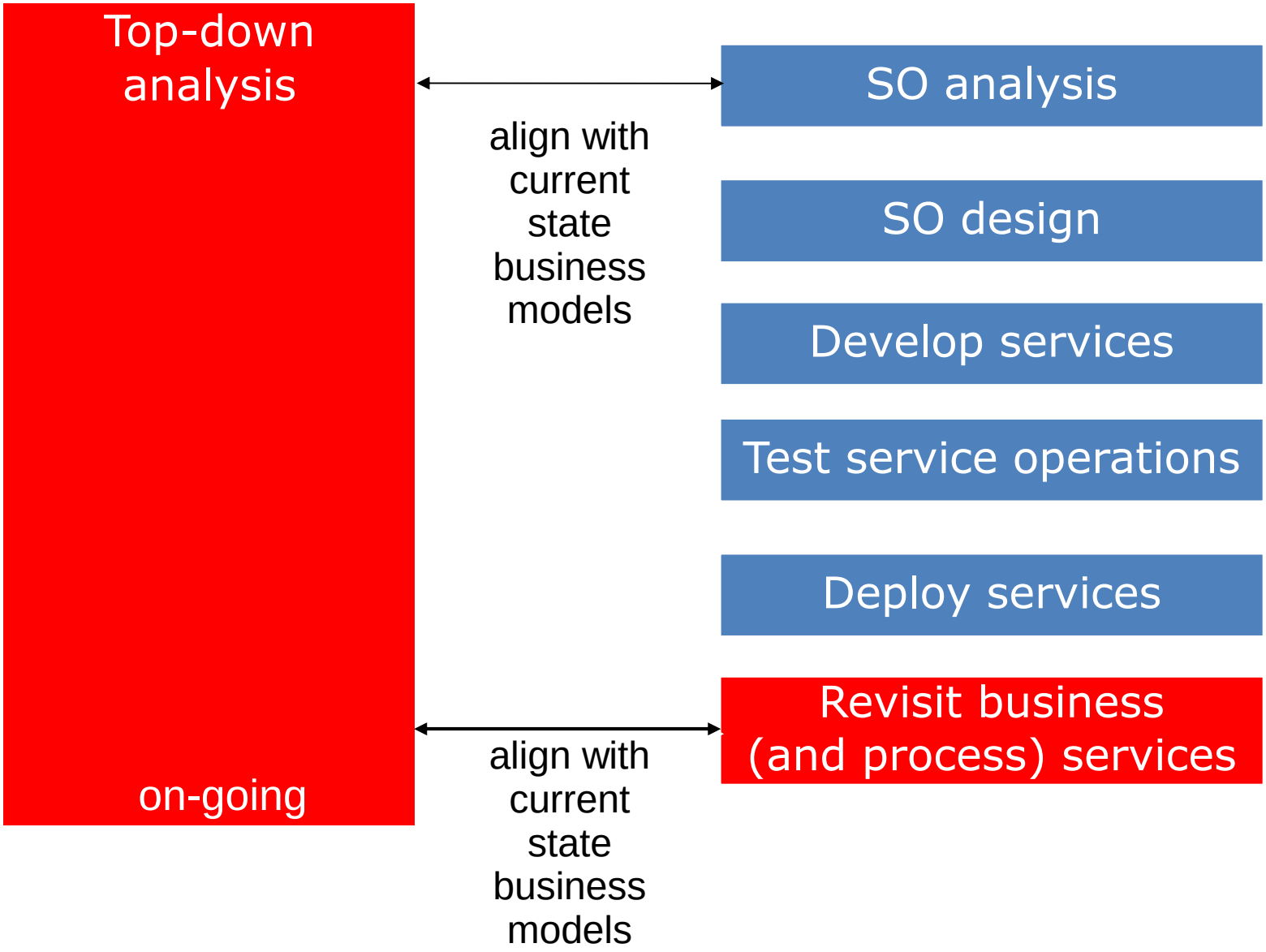


Bottom-up strategy



application service = infrastructure service

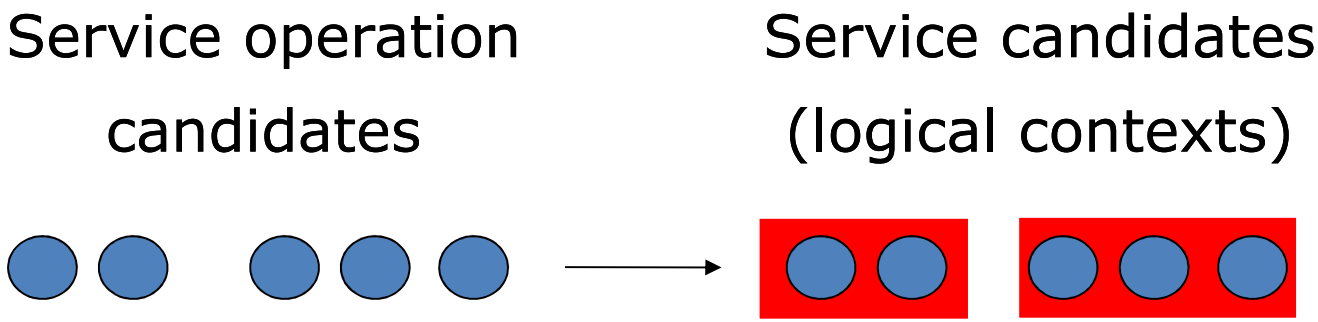
Agile strategy



Service oriented analysis

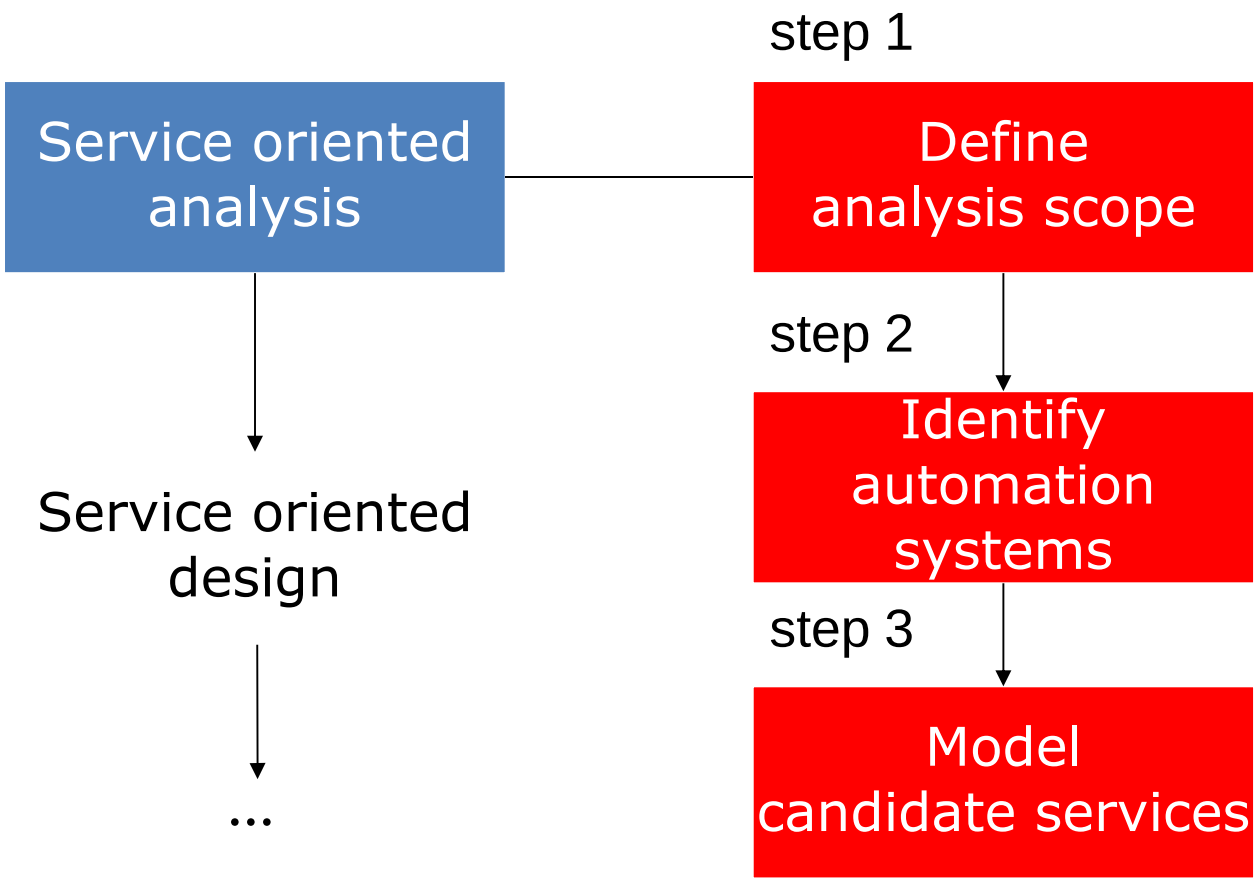
- The process of determining how business automation requirements can be represented through service orientation

Goals of SO analysis



- Appropriateness for intended use
- Identify preliminary issues that may challenge required service autonomy
- Define known preliminary composition models

3 Analysis sub-steps



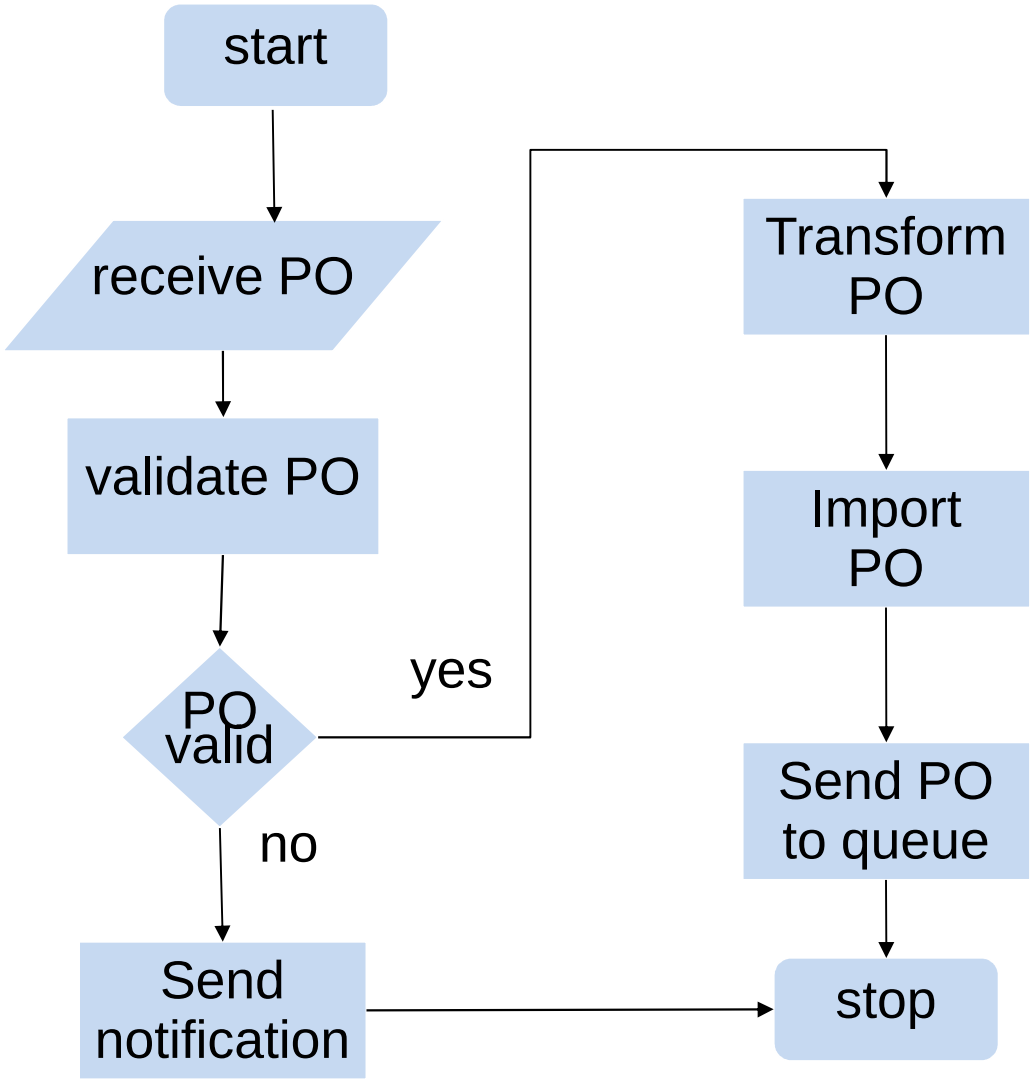
Step 1: Define analysis scope

– Mature and understood business requirements
  - S = ∑i Si, where smaller services may still be quite complex

– Can lead to
  - process-agnostic services/service operations (generic service portfolio)
  - services delivering business-specific tasks

– Models: UML use case or activity diagrams

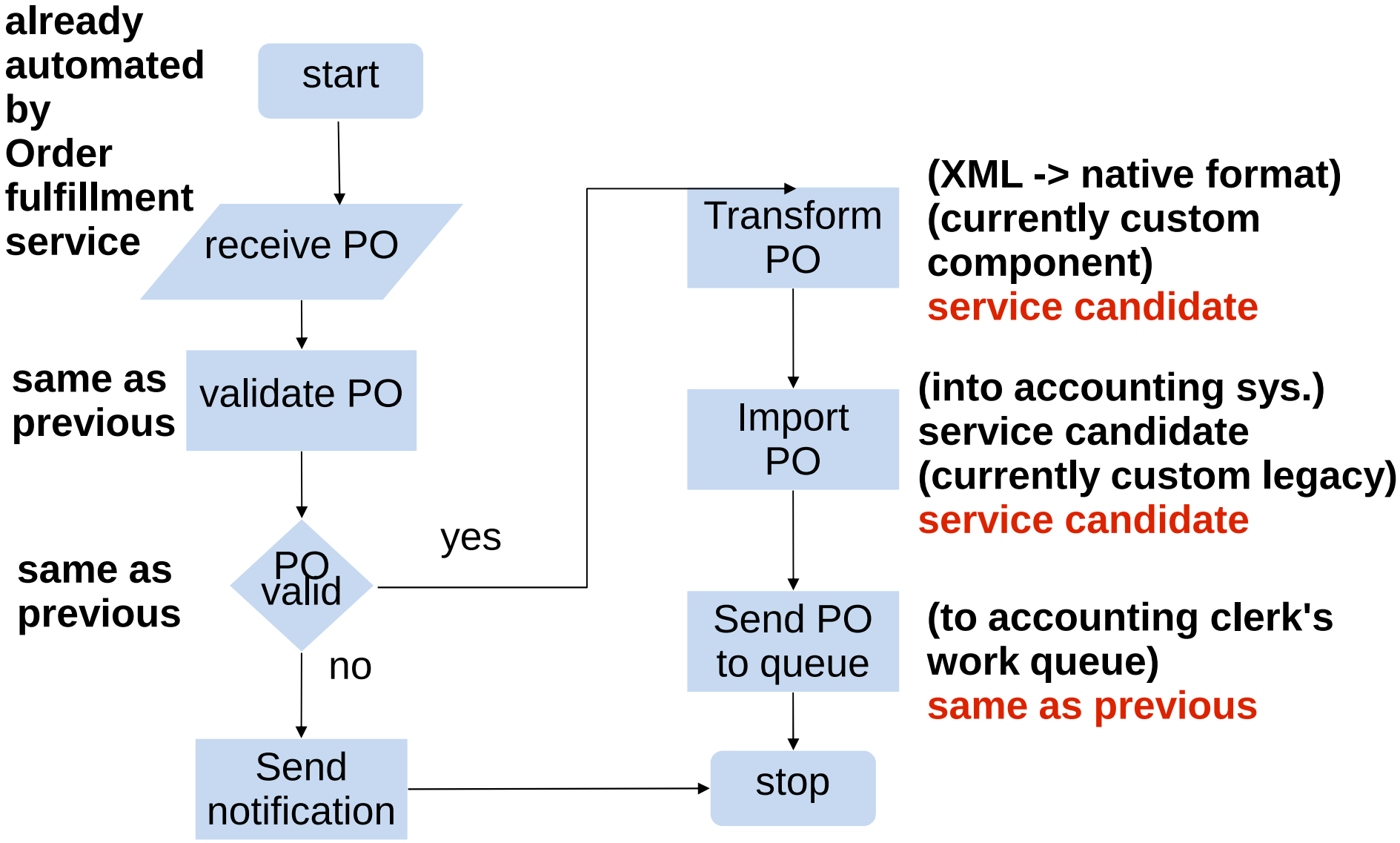
Order Fulfillment Process



Step 2: Identify automation systems

- What is already implemented?
  - encapsulate
  - replace
- Models: UML deployment diagram, mapping tables

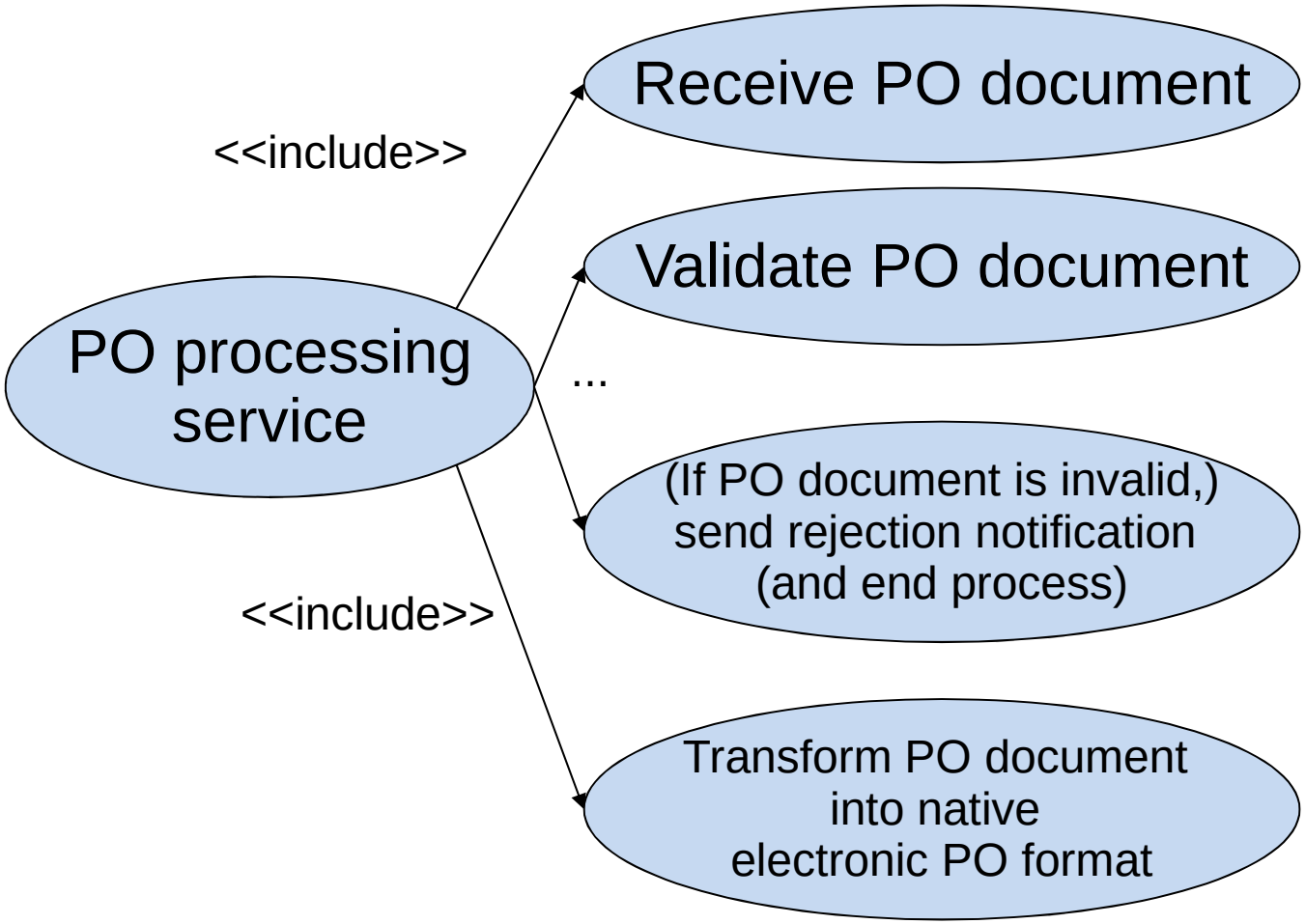
Order Fulfillment Process



Step 3: Model candidate services

- How to compose services?
- Service (candidates) conceptual model
  - operations + service contexts
  - SO principles
- Focus on task- and entity-centred services
- Models: BPM, UML use case or class diag.

Example service operation candidates



Example business process logic

- Not service operation candidates
  - if PO document is valid, proceed with the transform PO document step
  - if the PO document is invalid, end process

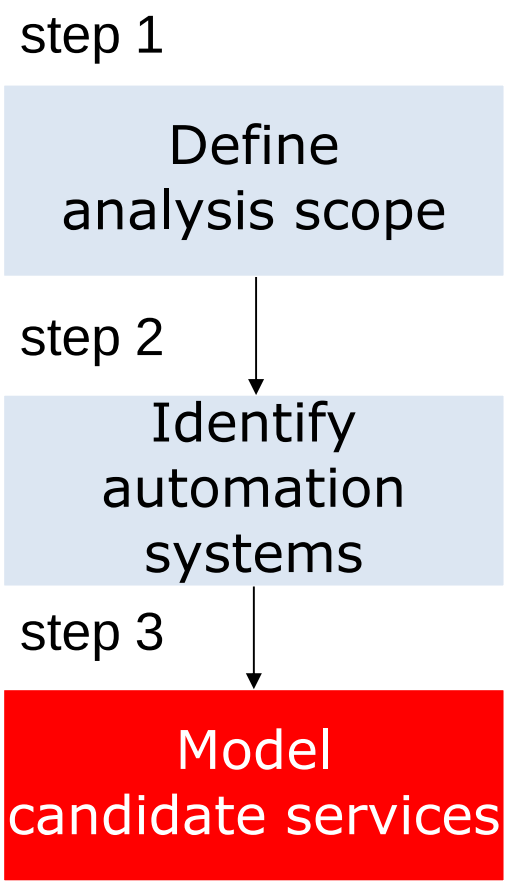
Task- versus entity-centred services

- Task-centred
  - (+) direct mapping of business requirements
  - (-) dependent on specific process
- Entity-centred
  - (+) agility
  - (-) upfront analysis
  - (-) dependent on controllers

Benefits of business-centric SOA

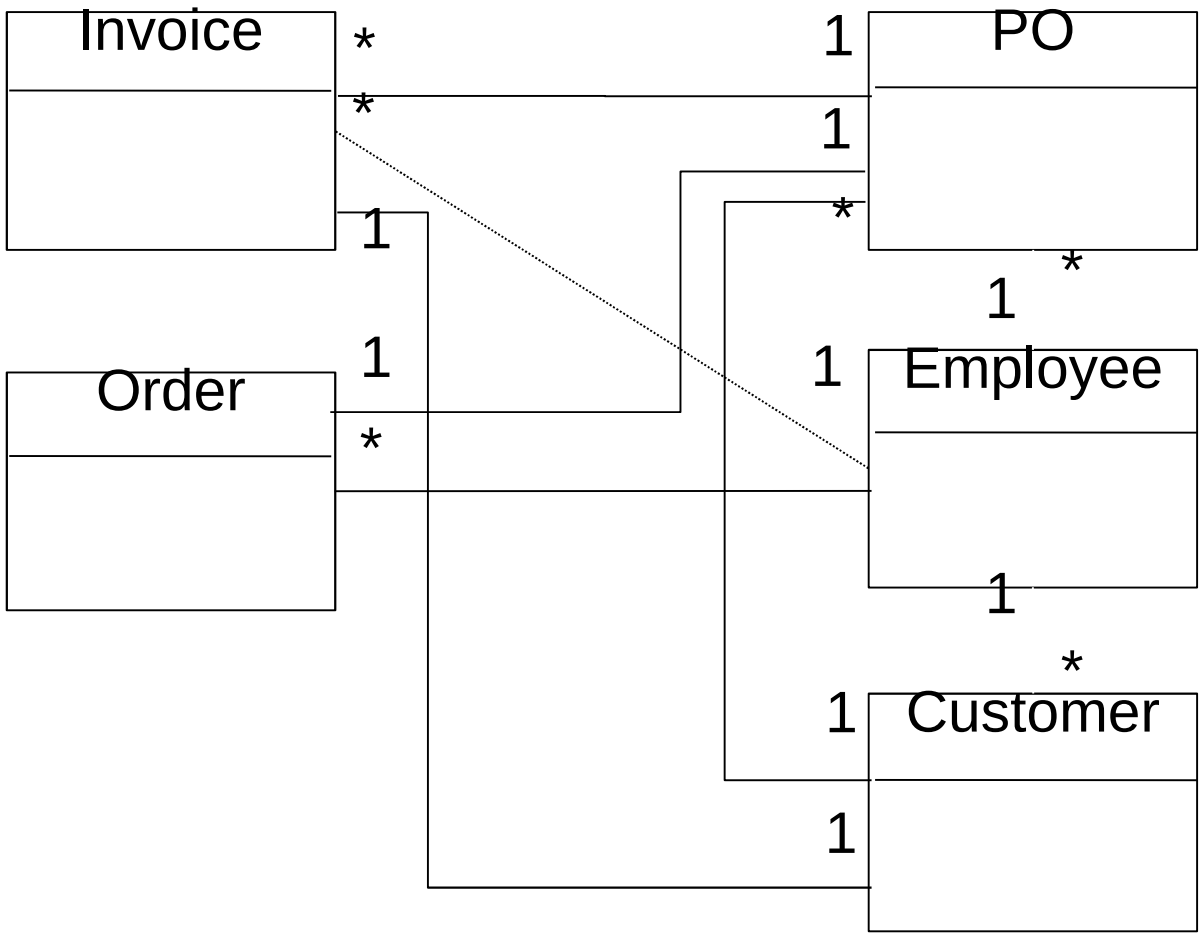
- introduce agility
- prepare for orchestration
- enable reuse

Modeling



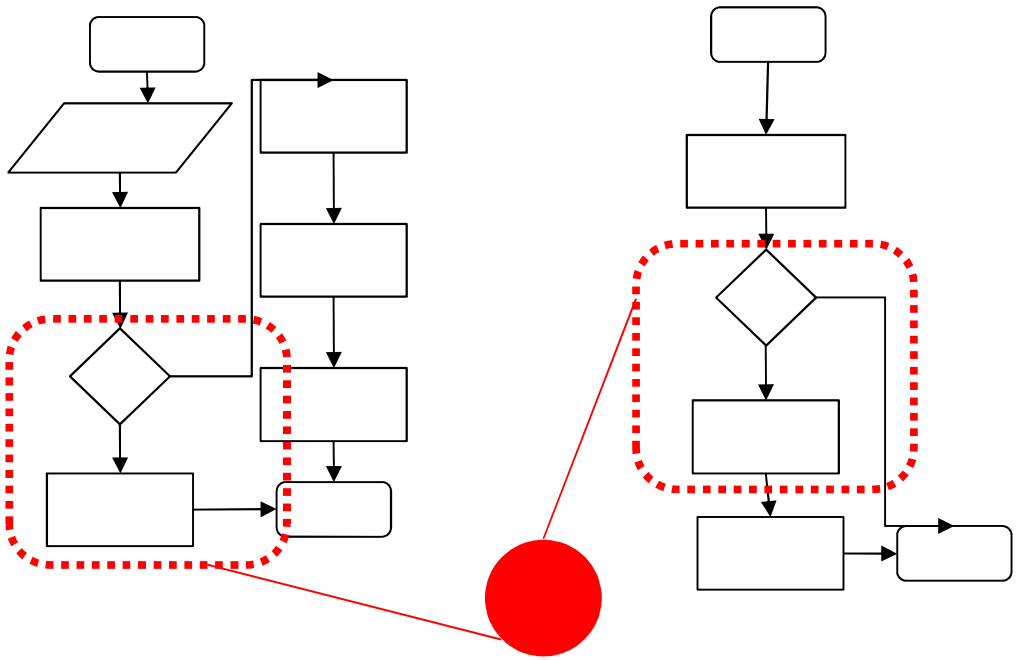
- identify agnostic service candidates
- filter out process-specific logic
- apply SO principles
- identify candidate service compositions
- identify application service operation/service candidates
- apply SO principles
- revise operation candidates grouping

Entity Models



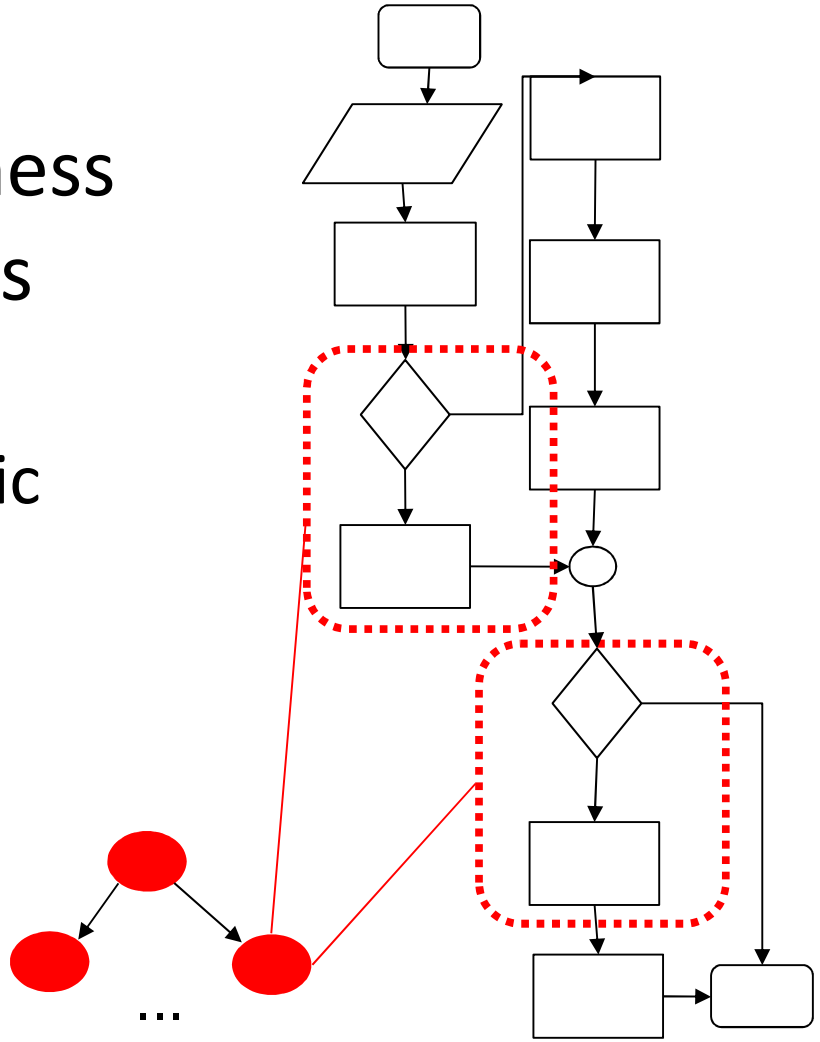
Service modeling guidelines

- task-centric business service candidates
  - reusability of encapsulated logic across processes



Service modeling guidelines

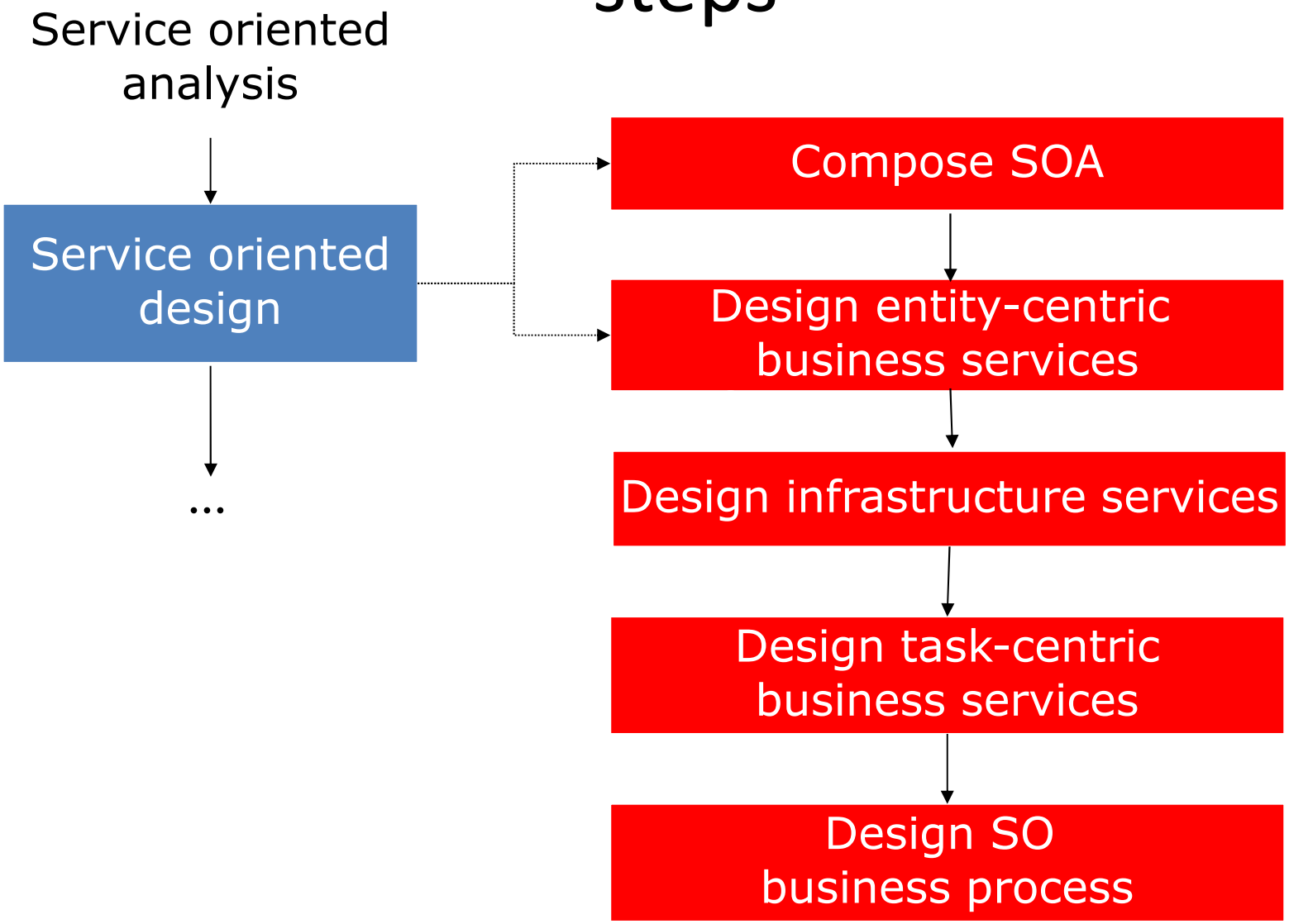
- task-centric business service candidates
  - reusability of encapsulated logic within a process



Service modeling guidelines

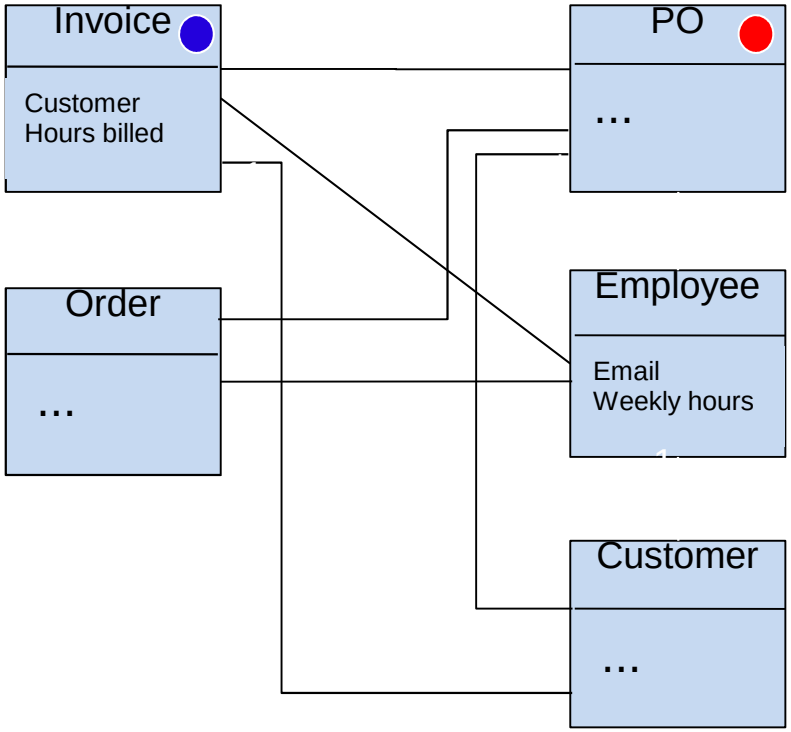
- identify logical units of work with explicit boundaries
  - SO principle about autonomous services (hiding logic)
- ...

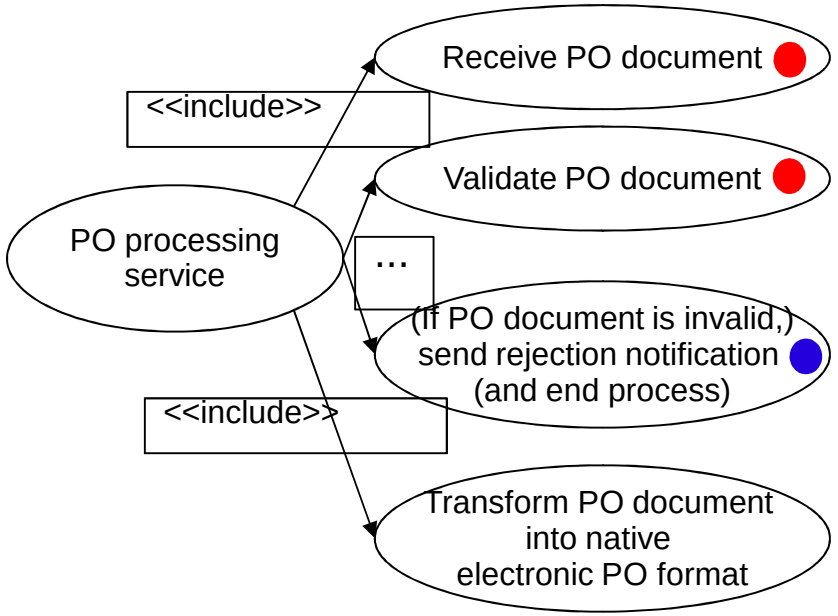
Service-oriented design: design sub-steps



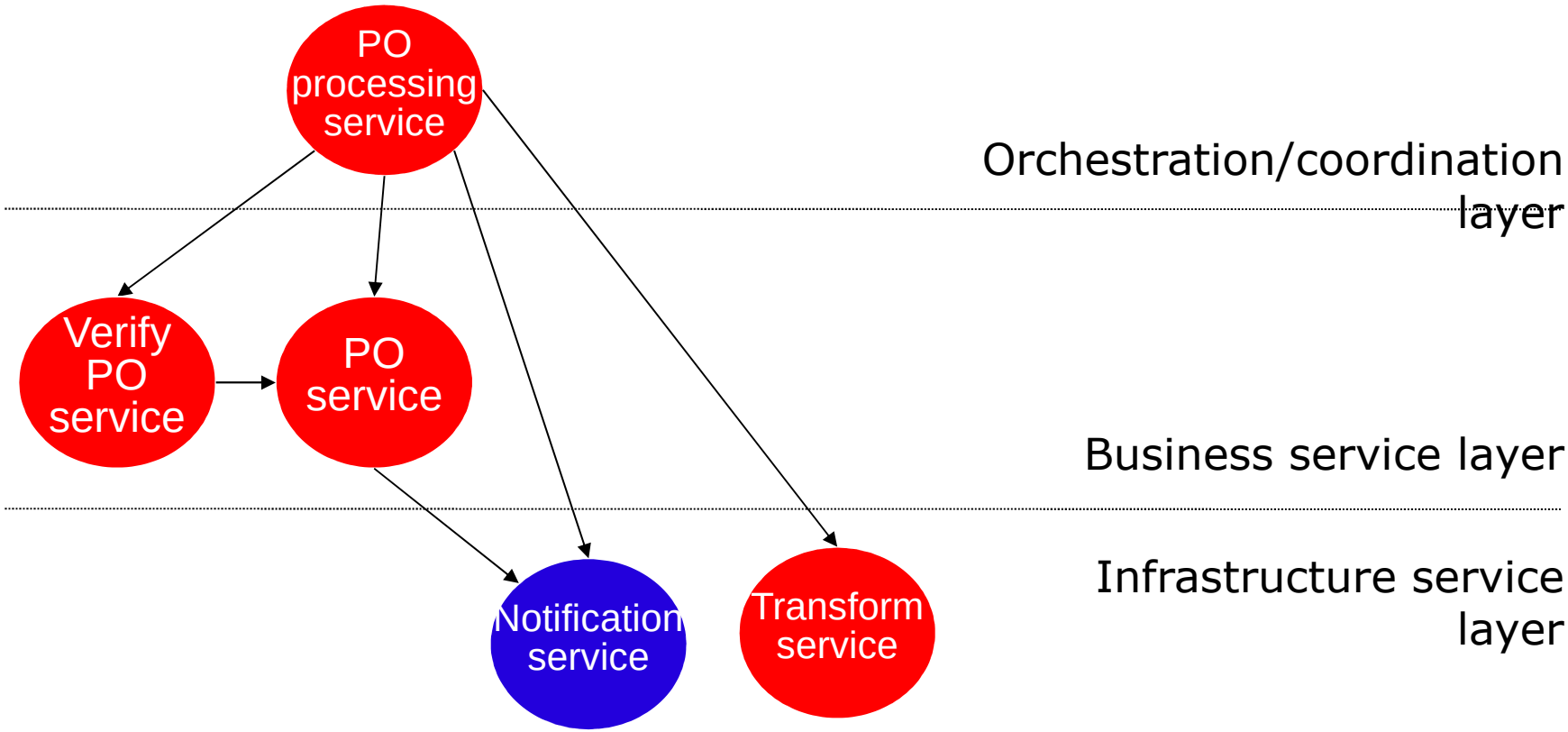
Entity-centric business services

- Goal: entity-centric business service layer + parent orchestration layer



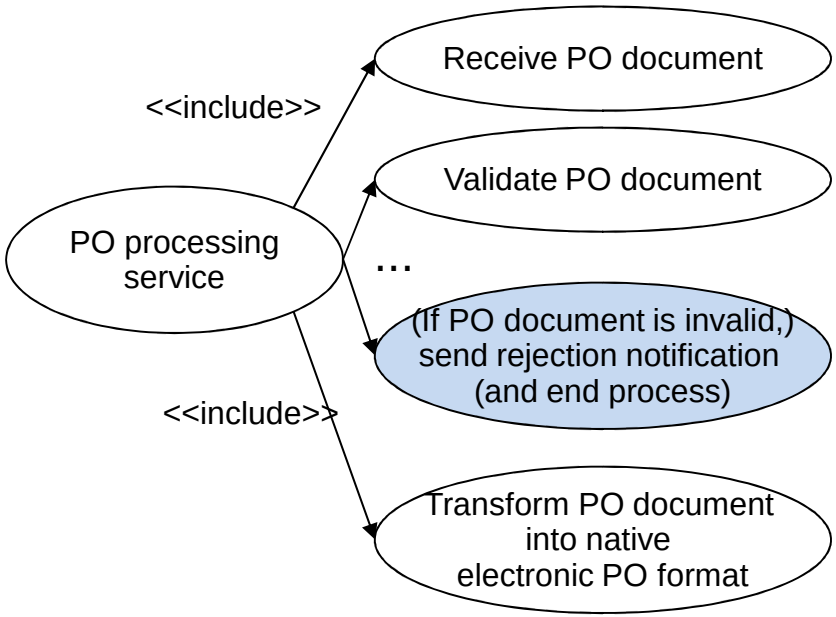


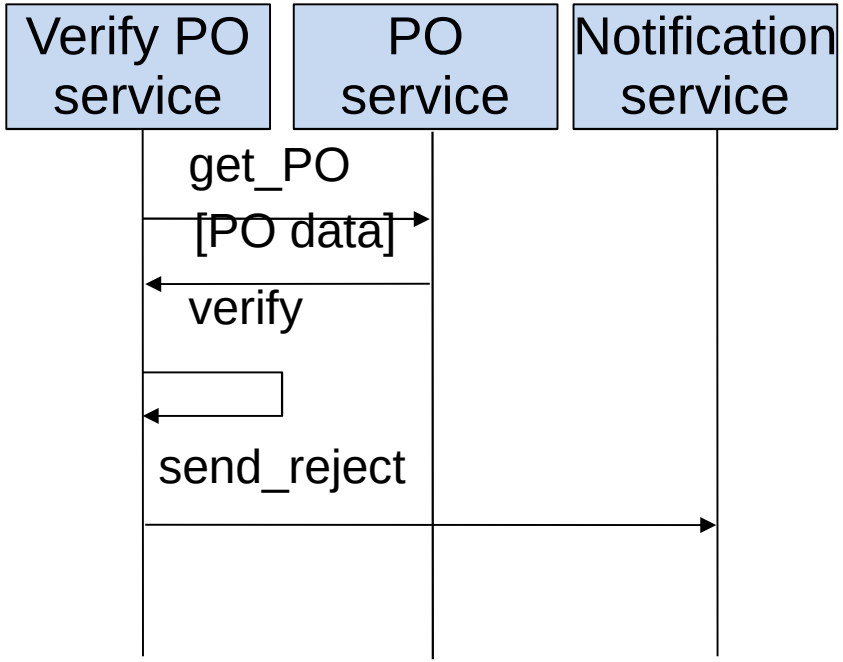
Infrastructure services



Task-centric business services

- UML sequence diagram
  - express and refine order of invocations implicit in the UML use case diagram





Summary

- Services have a long history (telephony)
- Most important characteristic: dynamic discovery of services
- SOA as architectural style
- Today's Web services mostly syntax-based
- Key design decisions in SOSE concern service layering, industry standards, and relevant SO principles
- SOSE differentiates from traditional life cycles mainly in the analysis and design phases