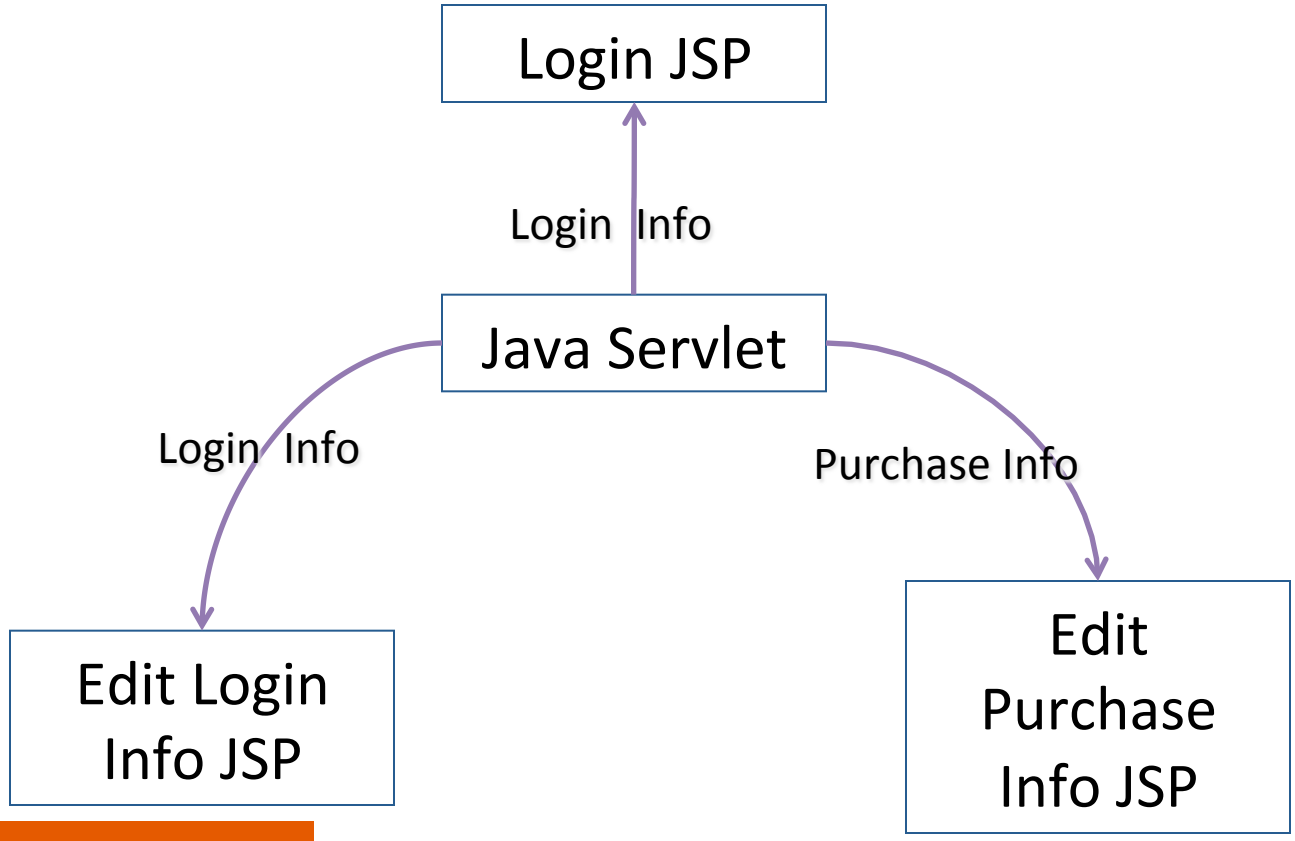
Software Decomposition

http://www.flickr.com/photos/bramkfarabraun/2704013961/

Decomposition: providing a detailed view of a component

Decomposition of the "website" component

Typical J2EE system: Servlet passes data to JSP, which displays it; browser posts back to servlet



Approaches for decomposing an architecture

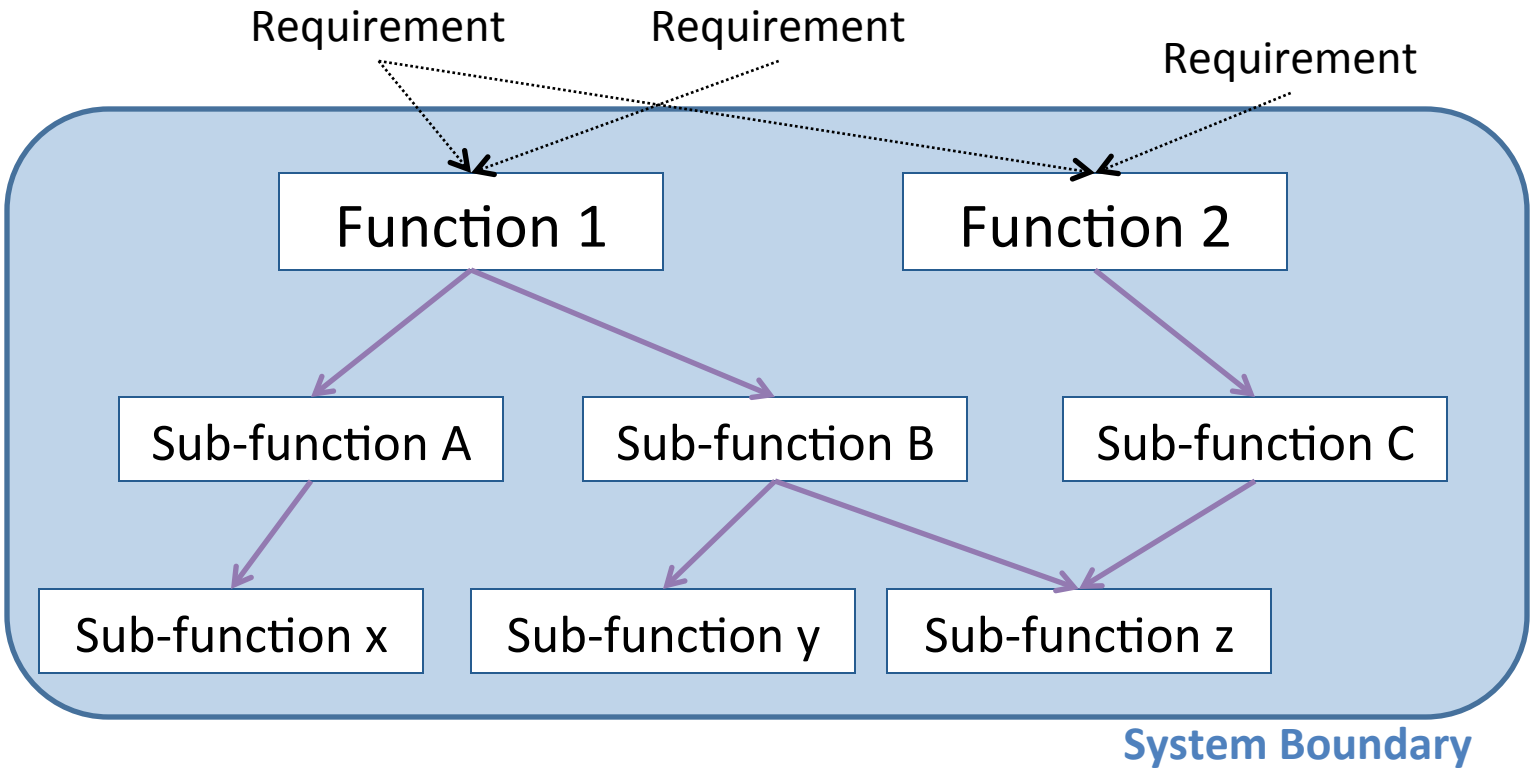
- Functional decomposition
- Data-oriented decomposition
- Object-oriented decomposition
- Process-oriented decomposition
- Feature-oriented decomposition
- Event-oriented decomposition

Functional decomposition

- Break each requirement into functions, then break functions recursively into sub-functions
  - One component per function or sub-function

- Each function computationally combines the output of sub-functions
  - E.g.: ticket_price = fee(station$_1$) + fee(station$_2$)
    + distance_fee(station$_1$, station$_2$)
    + fuel_surcharge(station$_1$, station$_2$)

Functional decomposition

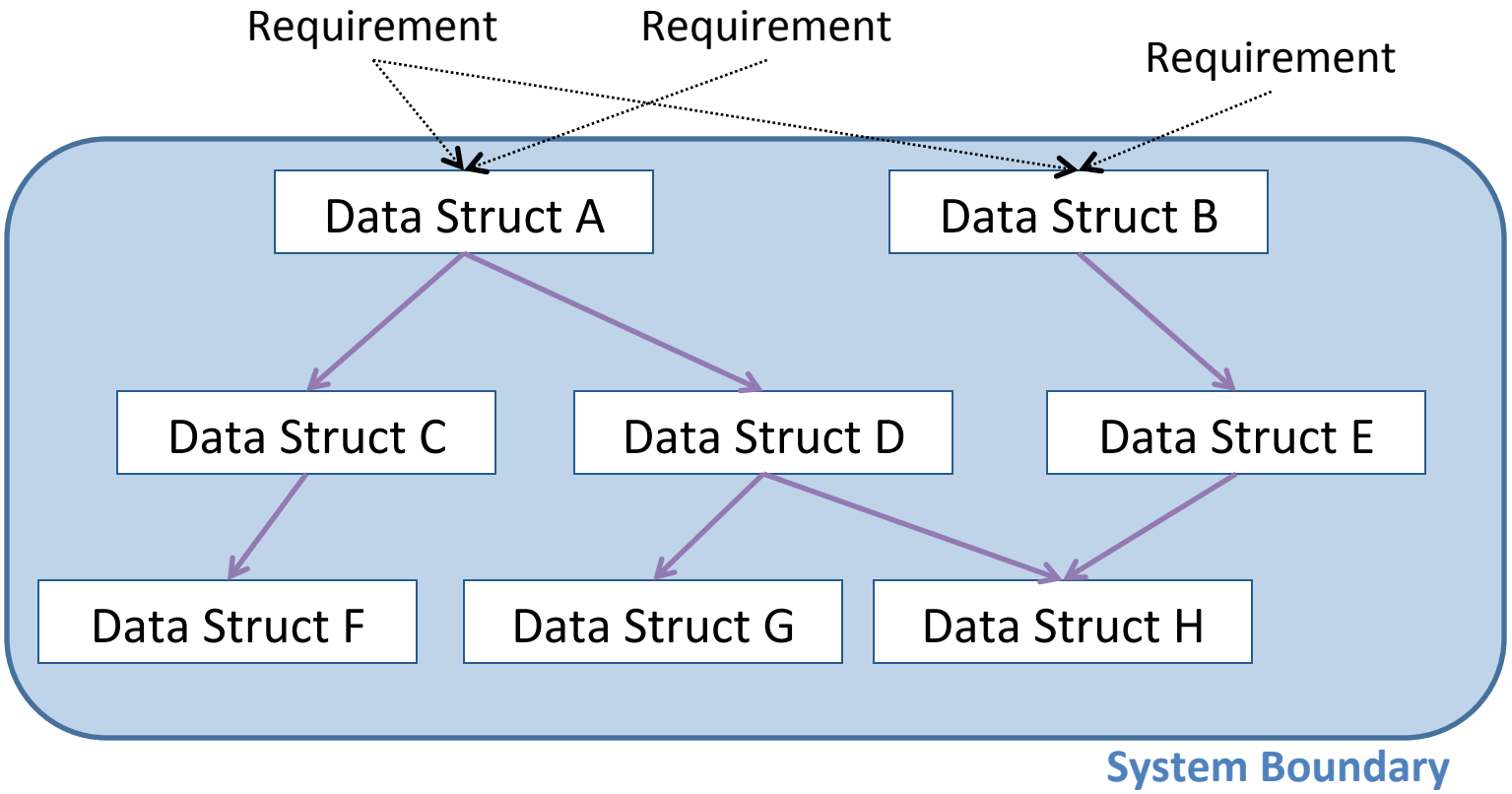


Data-oriented decomposition

- Identify data structures in requirements, break data structures down recursively
  - One component per data structure

- Each data structure contains part of the data
  - E.g.: Purchase info = Ticket info and billing info;
    ticket info = two stations and a ticket type;
    billing info = contact info and credit card info;
    contact info = name, address, phone, …;
    credit card info = type, number, expiration date

Data-oriented decomposition

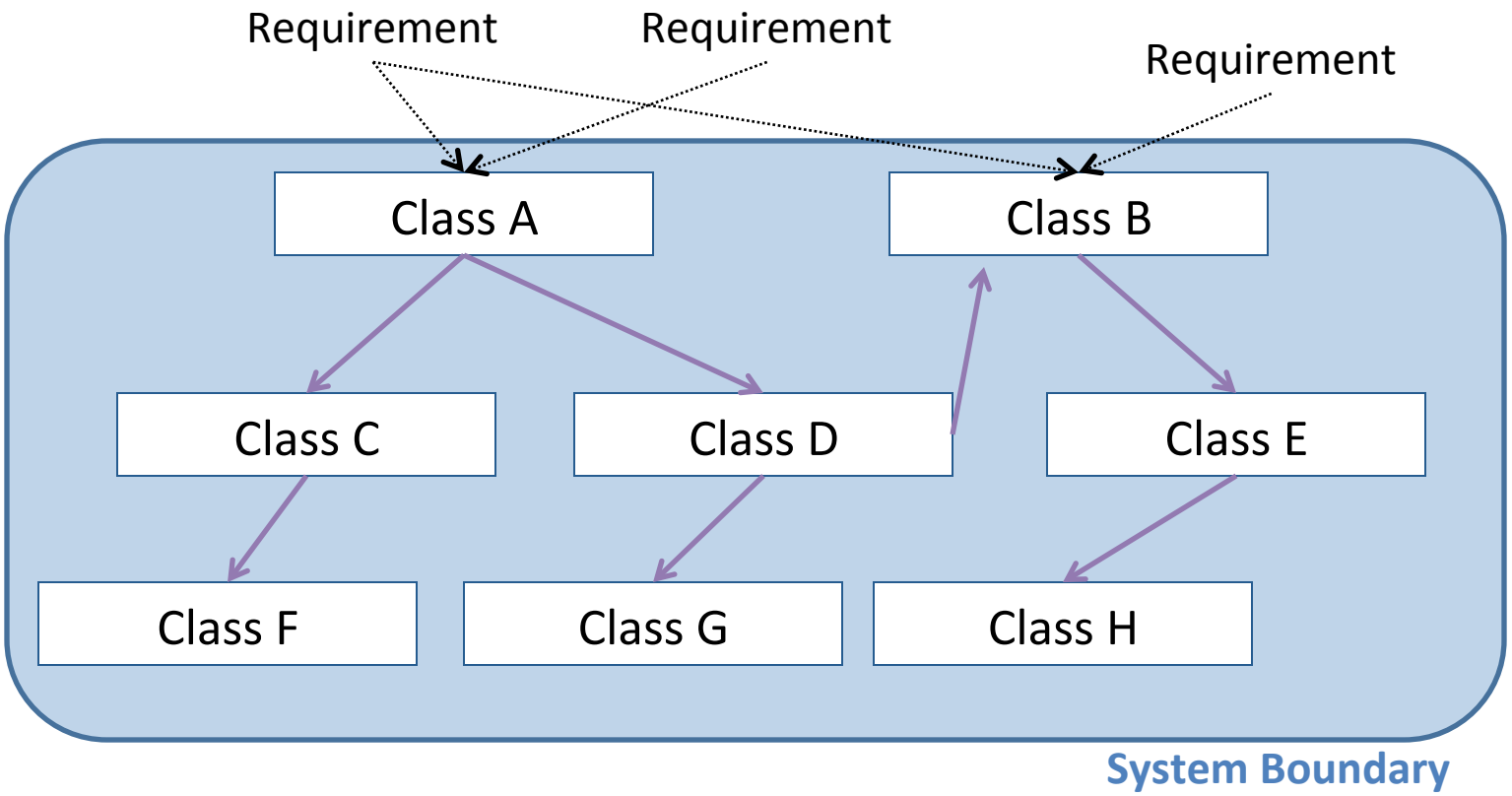


Object-oriented decomposition

- Identify data structures aligned with functions in requirements, break down recursively
  - One class component per data+function package

- Each component contains part of the data+fns
  - OO decomposition essentially is the same as functional decomposition aligned with data decomposition

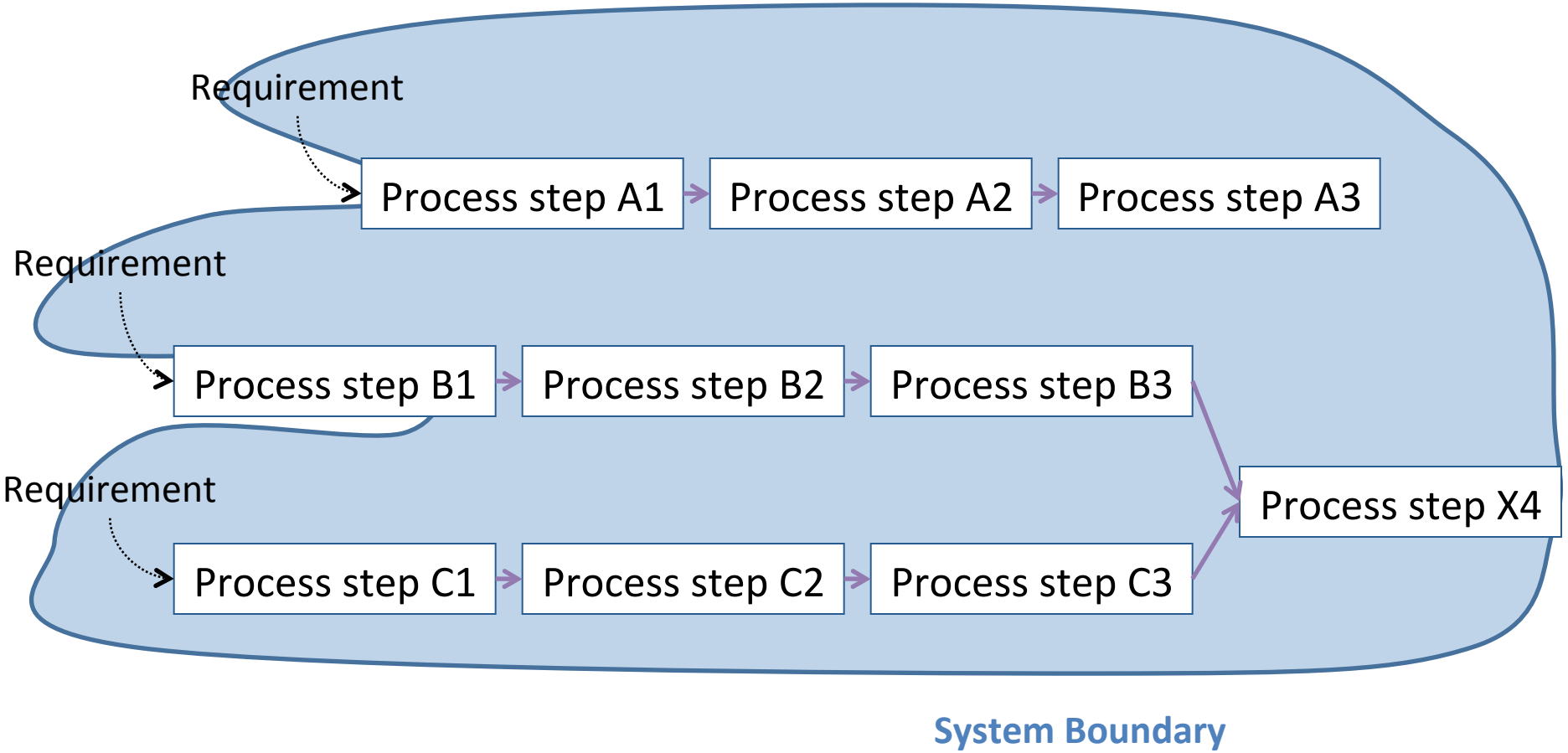
Object-oriented decomposition



Process-oriented decomposition

- Break requirements into steps, break steps into sub-steps recursively
  - One component per sub-step
- Each sub-step completes one part of a task
  - E.g.: one component to authenticate the user, another to display purchase info for editing, another to store the results away

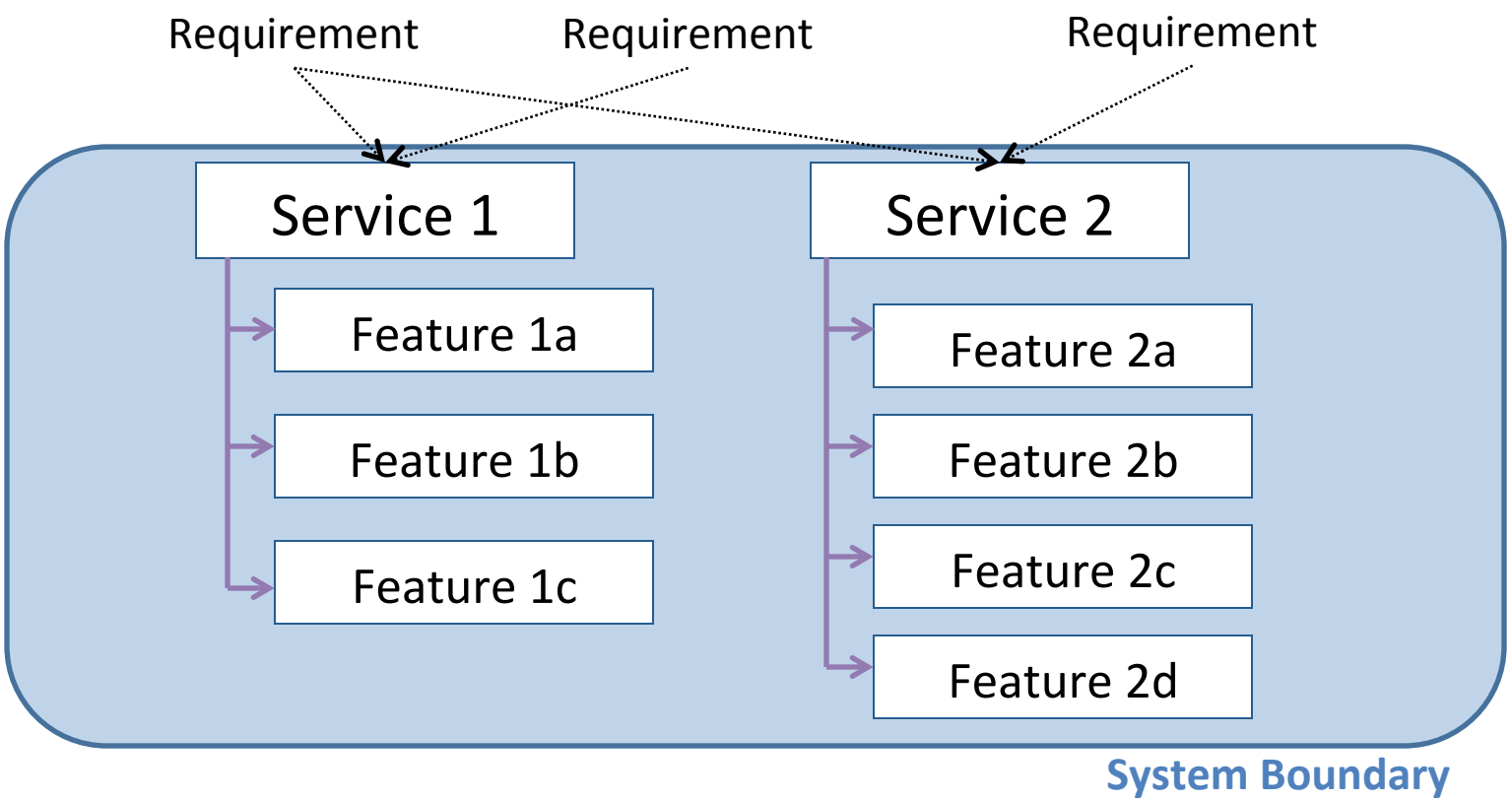
Process-oriented decomposition



Feature-oriented decomposition

- Break each requirement into services, then break services into features
  - One component per service or feature
- Each feature makes the service "a little better"
  - E.g.: service does basic authentication, but one feature gives it a user interface, another feature gives it an OpenID programmatic interface, another feature gives it input validation, and another feature does logging

Feature-oriented decomposition

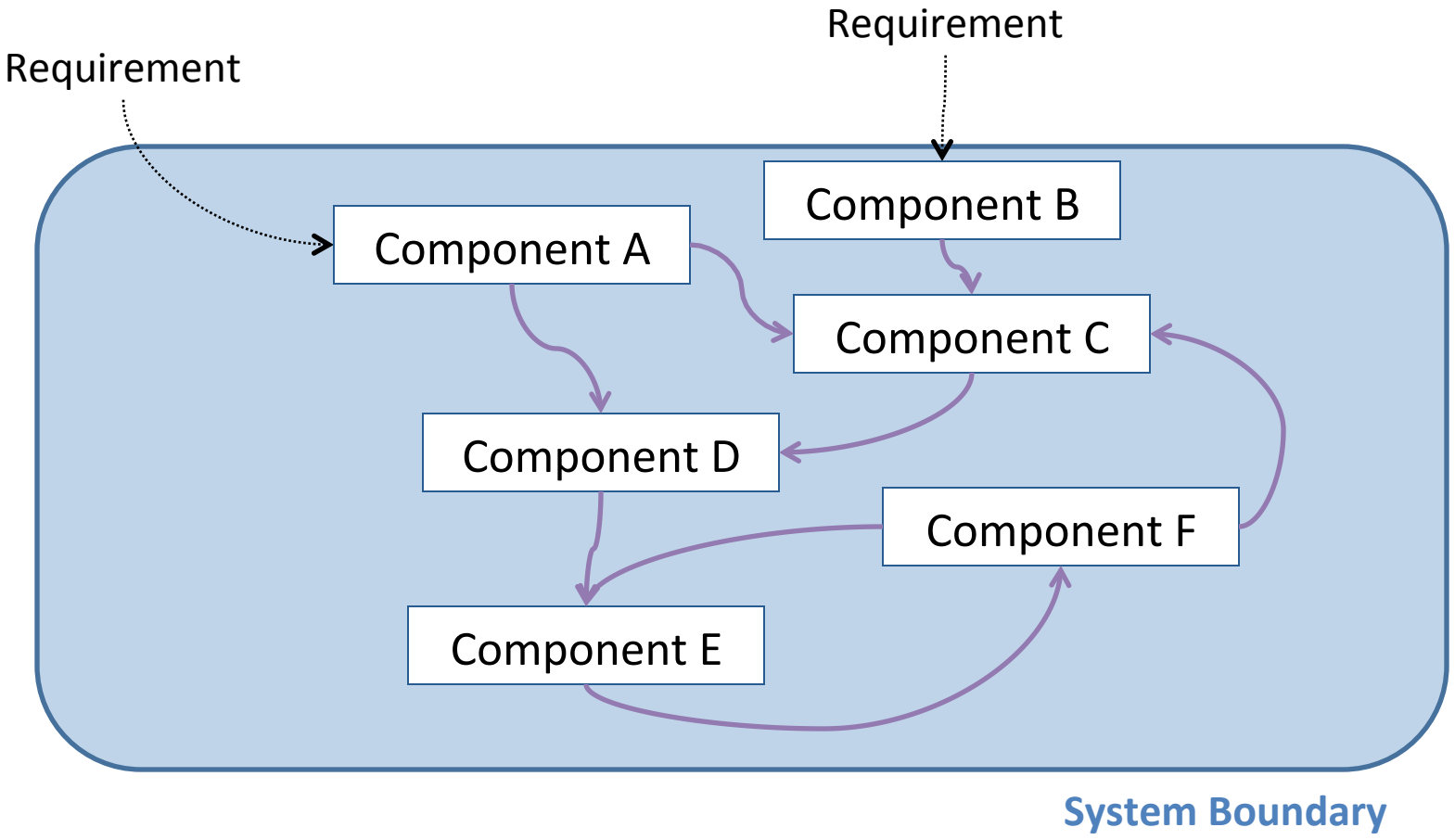


Event-oriented decomposition

- Break requirements into systems of events, recursively break events into sub-events and state changes
  - Each component receives and sends certain events, and manages certain state changes

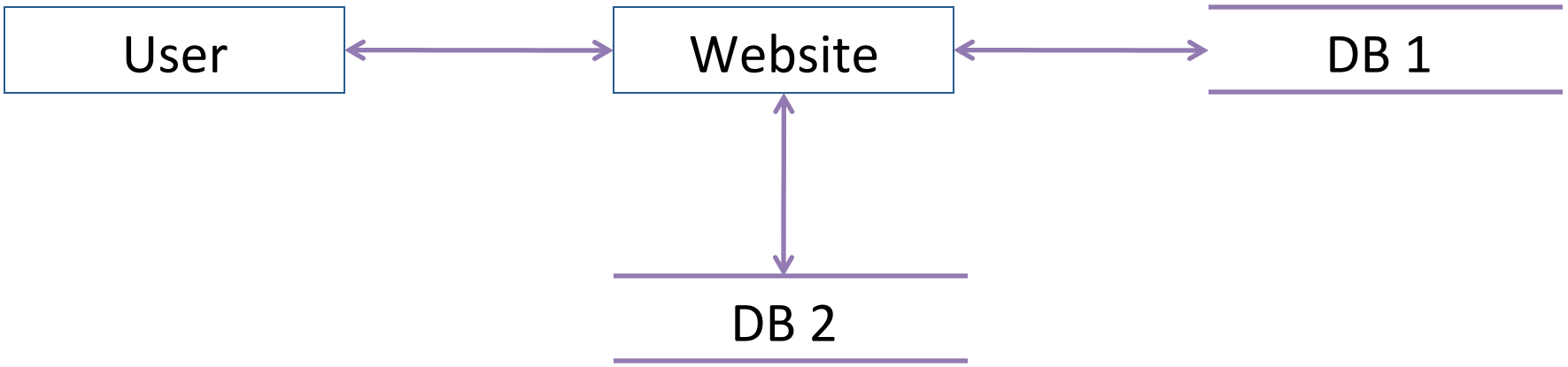
- Each component is like a stateful agent
  - E.g.: in the larger ticketing system, the mainframe signals the ticket printing system and the credit card company; the ticket printer notifies mainframe when it mails ticket to user

Event-oriented decomposition



Architectural style = a common kind of architecture

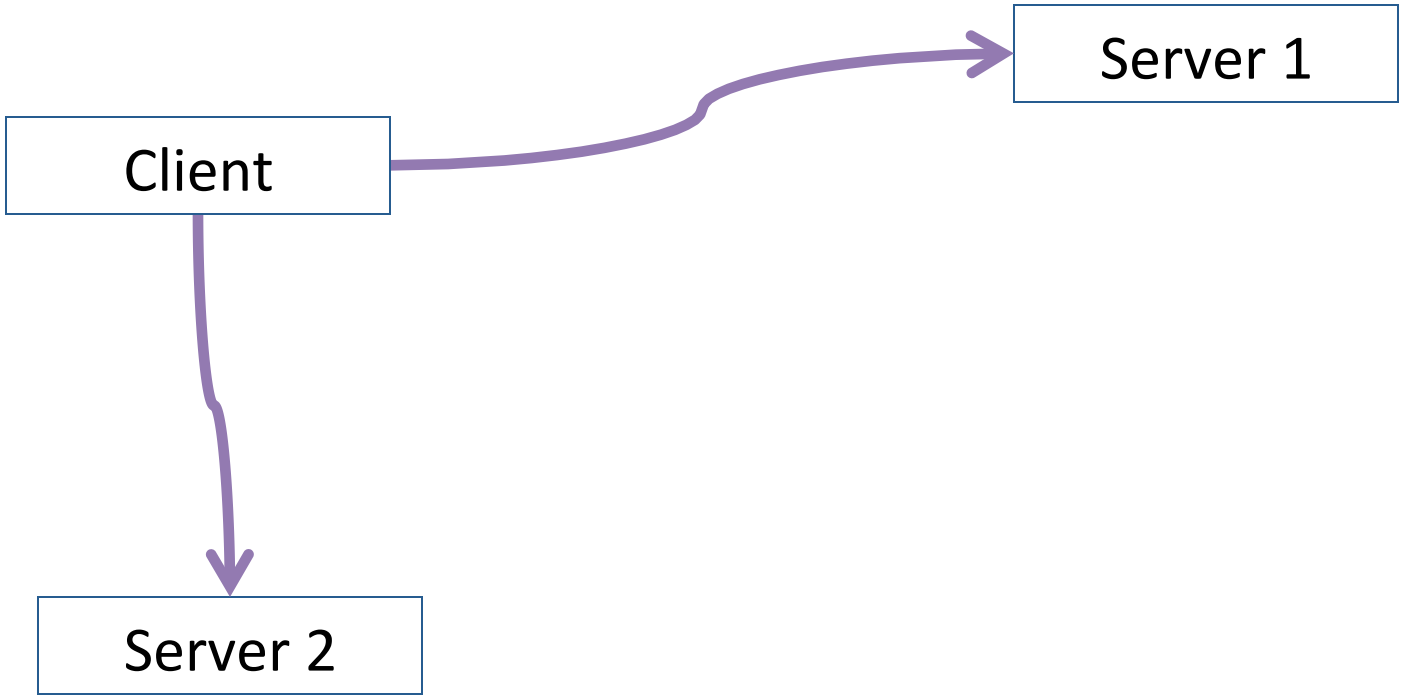
- Certain kinds of decomposition often occur
  - Certain kinds of components & connectors
  - Certain typical arrangements
- Example: which web app is shown below?



Could be just about any web app… they all look pretty similar at this level of abstraction.

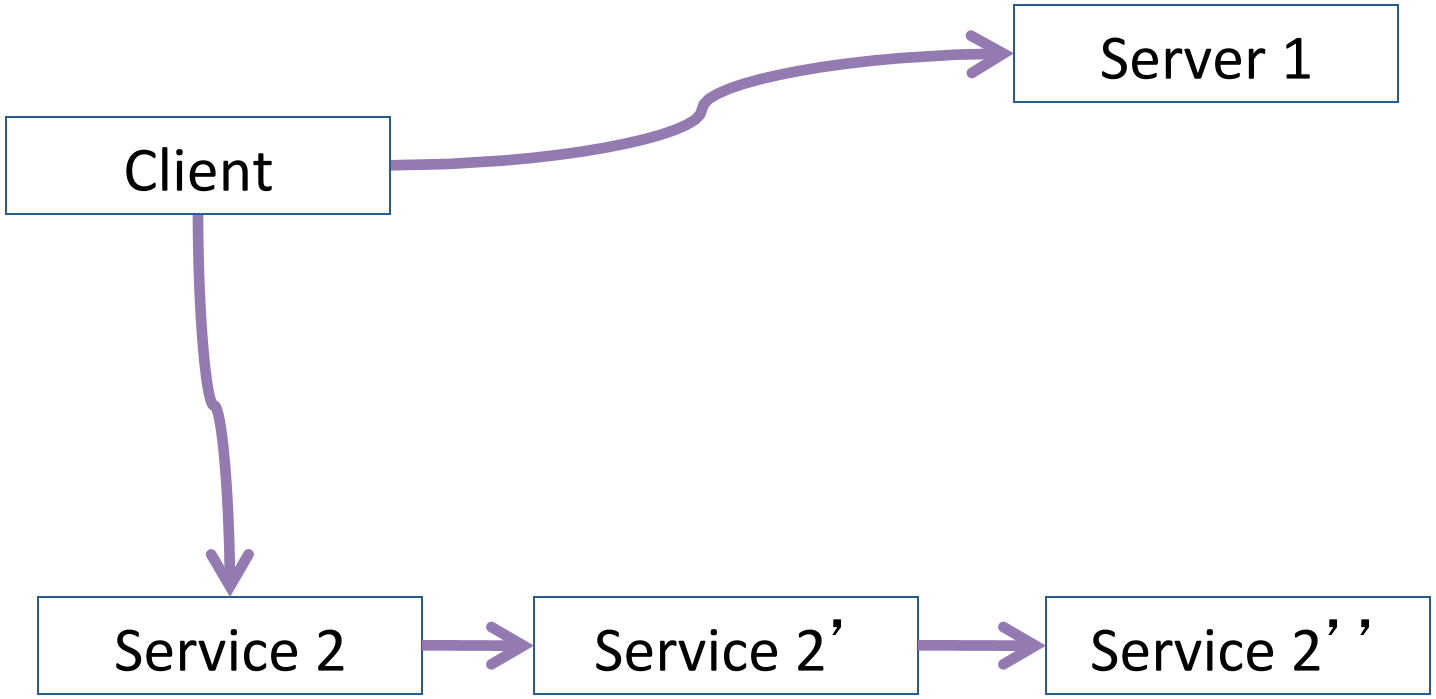
Mixing and matching is sometimes necessary

Simple client-server architecture



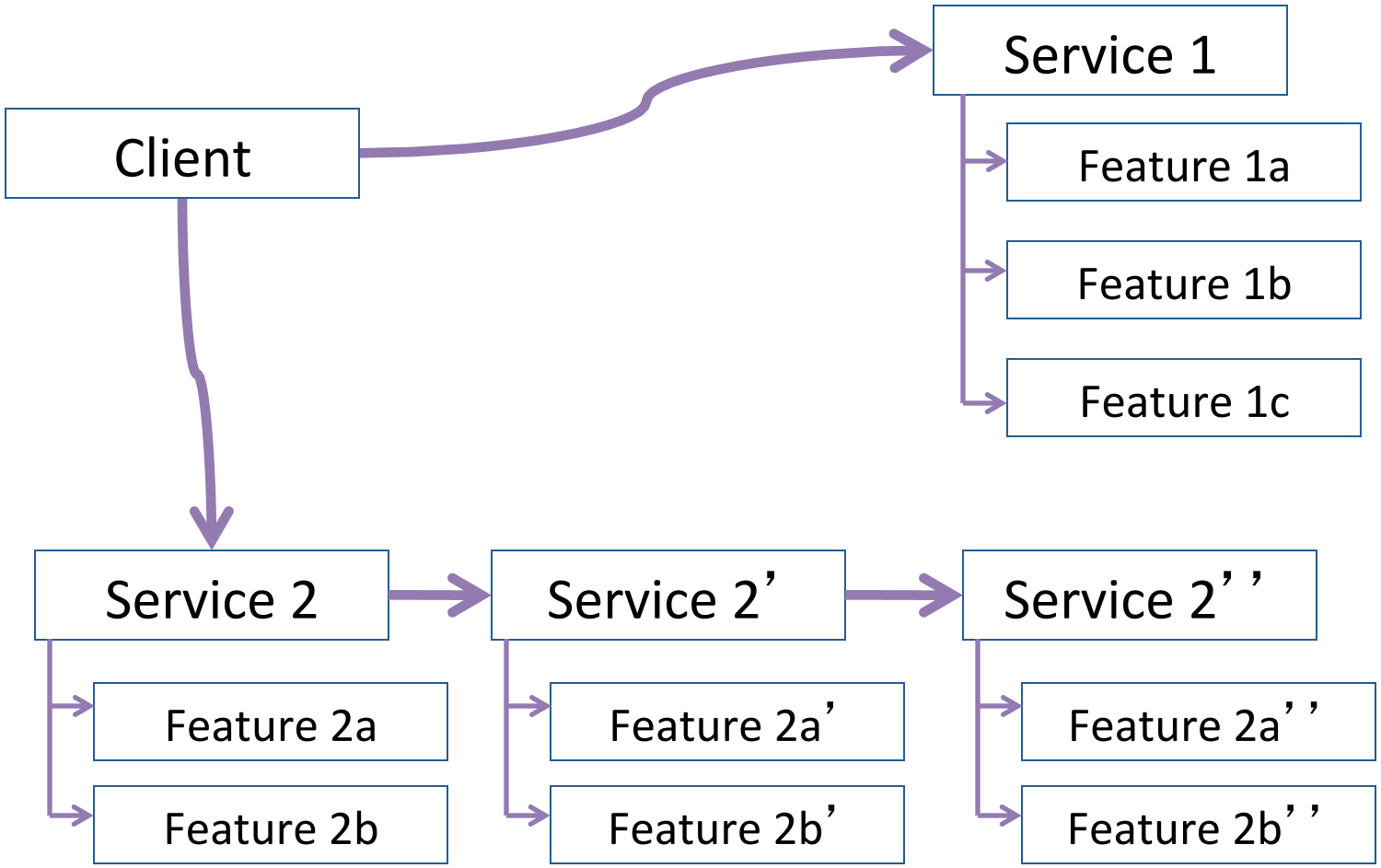
Mixing and matching is sometimes necessary

Decomposing one server may reveal a process-oriented design.



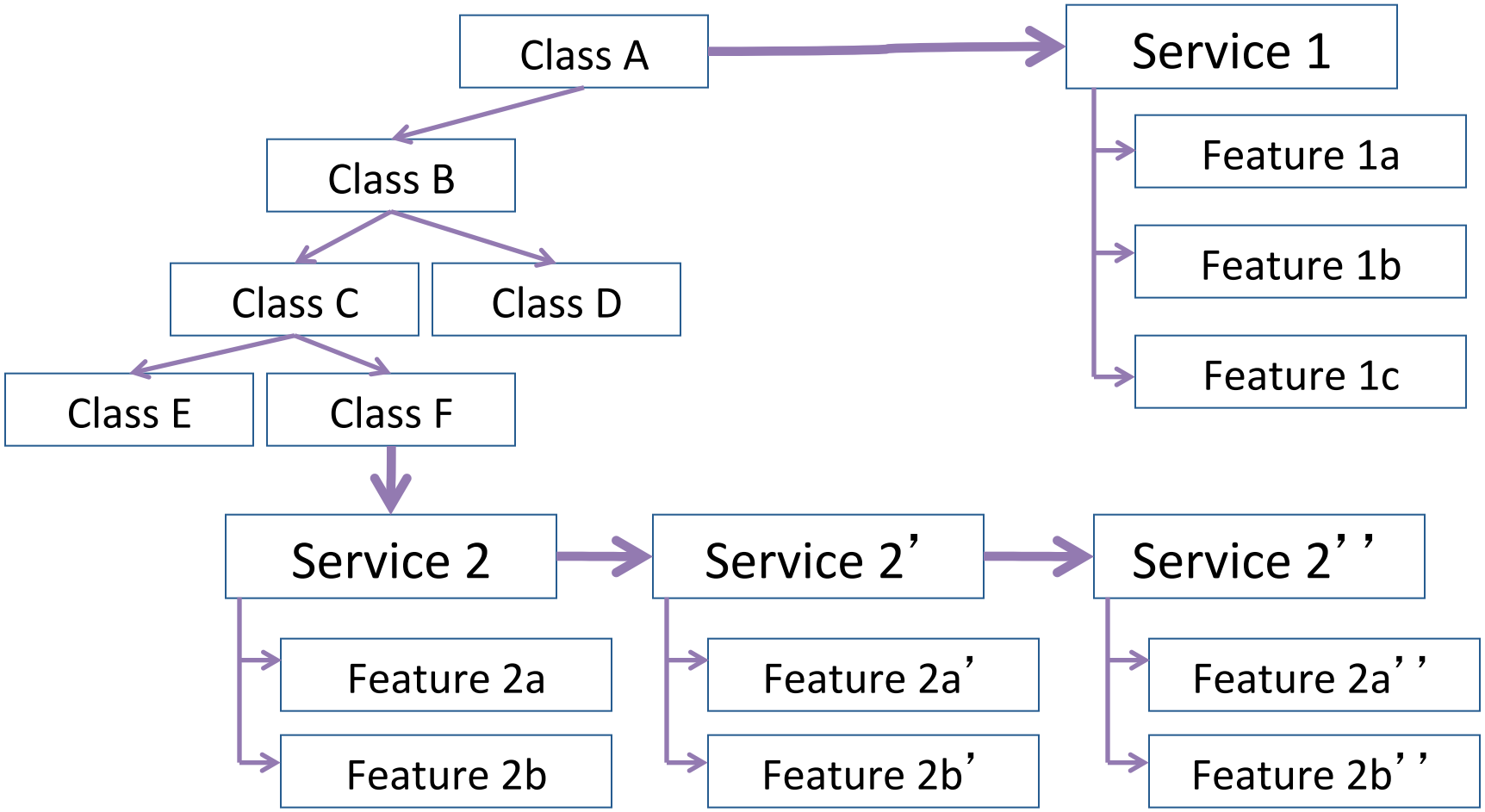
Mixing and matching is sometimes necessary

Decomposing the servers further may reveal a feature-oriented design.



Mixing and matching is sometimes necessary

Decomposing the client might reveal an object-oriented design.



Mixing and matching is sometimes necessary