CS5030

Software Architecture Design

Learning objectives

- On completing this lecture and associated reading, you should
  - Understand the responsibilities and benefits of software architectures
  - Be aware of factors that affect architecture design decisions
  - Know the activities involved in creating software architectures
  - Understand the concept and benefits of architecture styles
  - Be aware of a few well-known architecture styles and their characteristics
  - Understand the criteria for selecting an architecture style for a given context

Software architecture

- Every software system has an architecture
  - Planned or not
  - Documented or not
  - Fit for purpose or not

- If planned and designed appropriately, it can help guide development and maintenance

Architecture for software systems

“Architecture is about the important stuff. Whatever that is”

“The decisions you wish you could get right early in a project”

- Ralph Johnson (one of the Gang of Four)

"Architecture represents the significant design decisions that shape a system, where significant is measured by cost of change."

- Grady Booch

Architectural responsibilities

- Understanding context
- Making decisions
- Modelling
- Validating
- Delivery

[Eltjo Poort, 2020]

Uses of architecture - organisational

- Communicating high-level decisions to stakeholders
- Providing context for the system
- Planning work allocation
- Estimating cost

Uses of architecture - technical

- Focusing on meeting significant functional and non-functional requirements
- Analysing system properties
- Acting as blueprint for development and maintenance
- Improving large scale reuse
- Facilitating integration with other systems

Software architecture design

- Typically a stage between requirements engineering and software design
  - Iterative or plan-driven
- Software architecture is the basis for detailed design
- Sometimes boundary between architecture and detailed design is blurred
- Focus on quality attributes

Scale

- Architecture in the small
  - Architecture of individual programs
- Architecture in the large
  - Architecture of complex enterprise systems
  - Can include other systems, programs and components

Architecture design decisions

- Is there a generic application architecture that can be used?
- How will the system be distributed?
- What architectural styles are appropriate?
- What approach will be used to structure the system?
- How will the system be decomposed into modules?
- What control strategy should be used?
- How will the architectural design be evaluated?
- How should the architecture be documented?

Software architecture design

- Inputs
  - Prioritised requirements and constraints
    - Domain, functional and non-functional
- Process
  - Iterative process of decomposing system
  - Choice of architecture style to address requirements at each decision point
  - Different perspectives
- Output
  - Documented (possibly partial) architecture design

Design decisions

- Mapping
  - Characteristics of system as specified by significant requirements
  - structure and behaviour (including interactions) as defined by architecture decisions

- Tradeoffs may be required

Documenting software architecture

- Software architecture will be used by many stakeholders
  - Technical and non-technical
- Needs to be documented in an appropriate way so it can be shared and understood by all stakeholders
- Notations
  - Natural language
  - Boxes and lines
  - UML
  - Formal architecture languages

Software architecture views

- Difficult to capture all the architecture details of a system in a single specification / diagram
- We use a set of viewpoints or perspectives to model the details of software architecture
- A number of different viewpoints are used
  - For eg, logical, process, development, physical, deployment

Validating software architecture

- Software architectures should be validated against requirements

- Possible strategies
  - Architecture reviews
  - Architecture evaluation techniques
  - Proving it by code
  - Formal methods

Software architecture and agility

- Software architecture and agile development are compatible
- We don’t need to produce a complete architecture at the outset but we do need an overall blueprint
  - Refined / updated as development progresses
  - ‘Just enough’ architecture [George Fairbanks]
- Future needs and changes should be considered but system should not be over-built

Architecture styles / patterns

- Named collections of architectural design decisions
  - Applicable to a recurring context
  - Constrain decisions that are specific to a particular system in that context
  - Elicit beneficial qualities in the target system

[Taylor, Medvidovic & Dashofy]

- Determine structure and interactions
- Can be customised to each system

Design patterns

- In contrast to architecture styles ….
  - Provide templates for refining subsystems or components of a software system
  - Do not influence the fundamental structure of a software system
  - Help implement architectural patterns
- Each pattern typically only affects a single subsystem

Choosing styles

- Applicable context
  - Components
  - Interaction patterns – control and data
  - Constraints
- Advantages and disadvantages for
  - Structure and interaction
  - Non-functional or quality requirements
- Variations and refinements may be possible

Architecture styles - benefits

- Existing, proven architecture solutions to recurring system contexts
- Shared vocabulary and understanding
- Reuse of architecture design and code
- Guide to development and maintenance
- Partial documentation of system

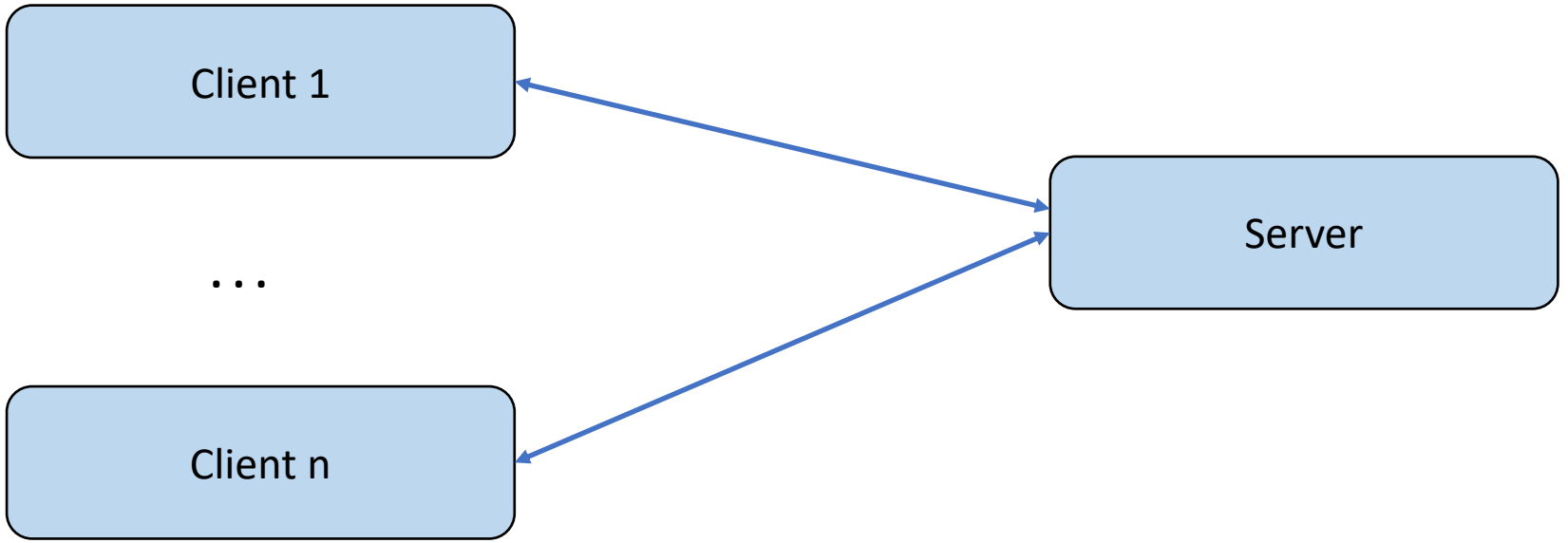
Architecture styles – considerations

- A style will not entirely define an architecture
- Multiple styles will need to be combined appropriately to design the system
- Choosing well-known styles might stifle innovation

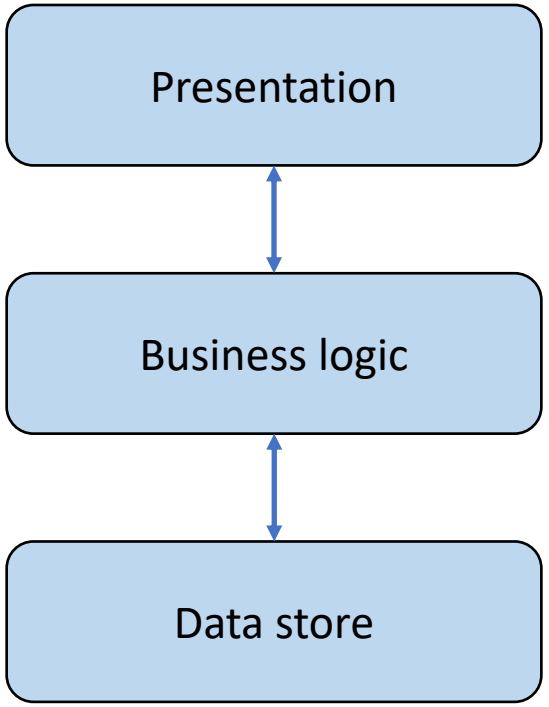
Popular architecture styles

- Client-server
- Tiered / layered
- Peer to peer
- Pipe and filter
- Microservices
- Model-view-controller

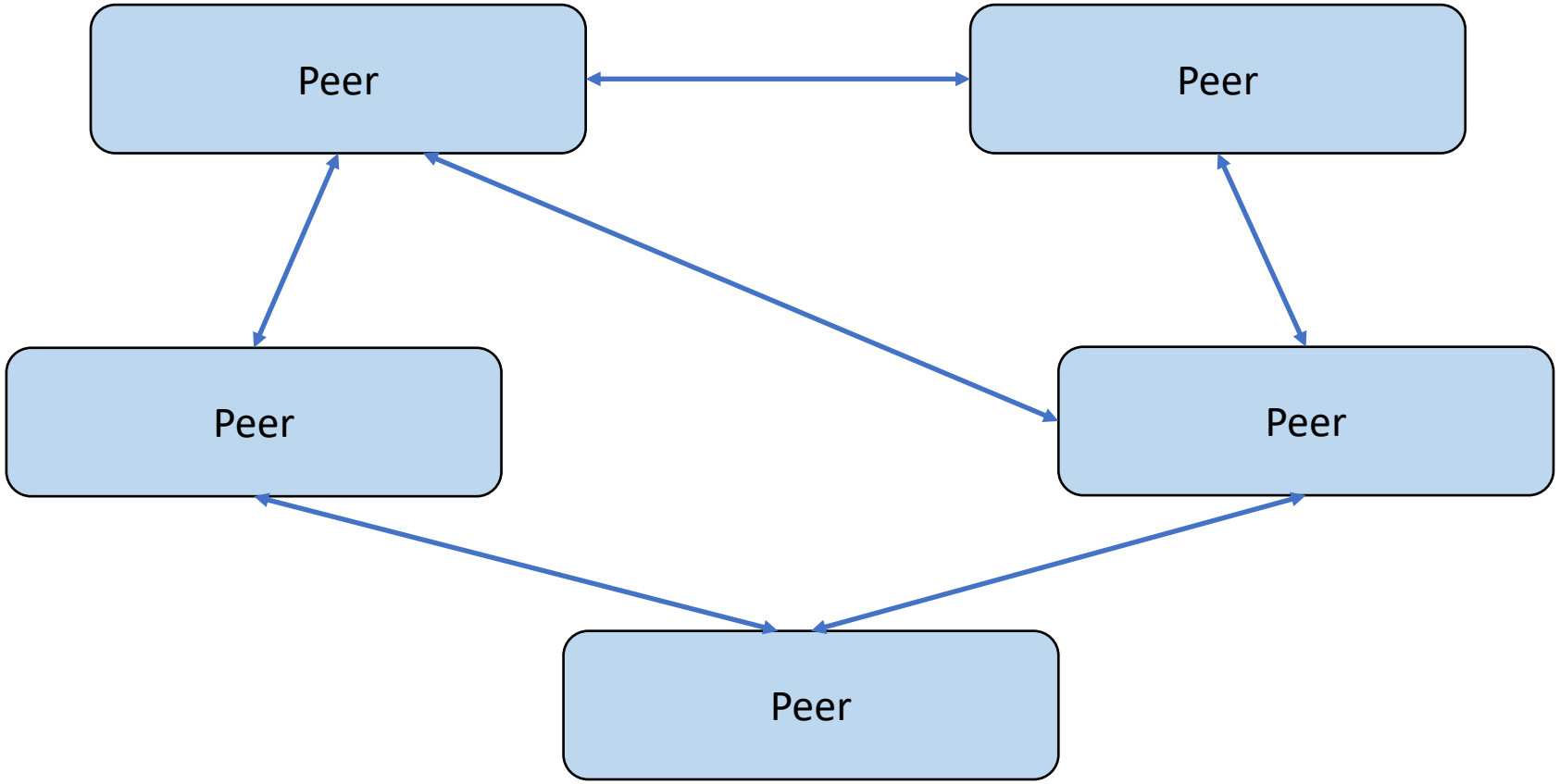
Client-server architecture



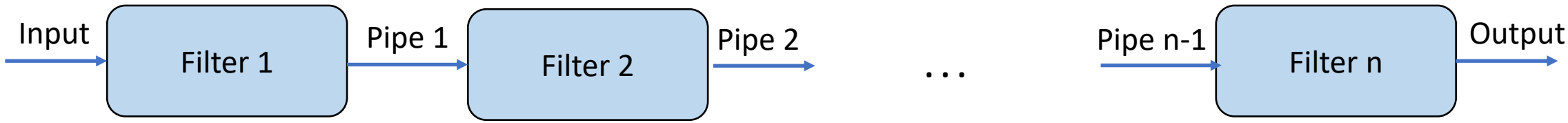
Layered / tiered architecture



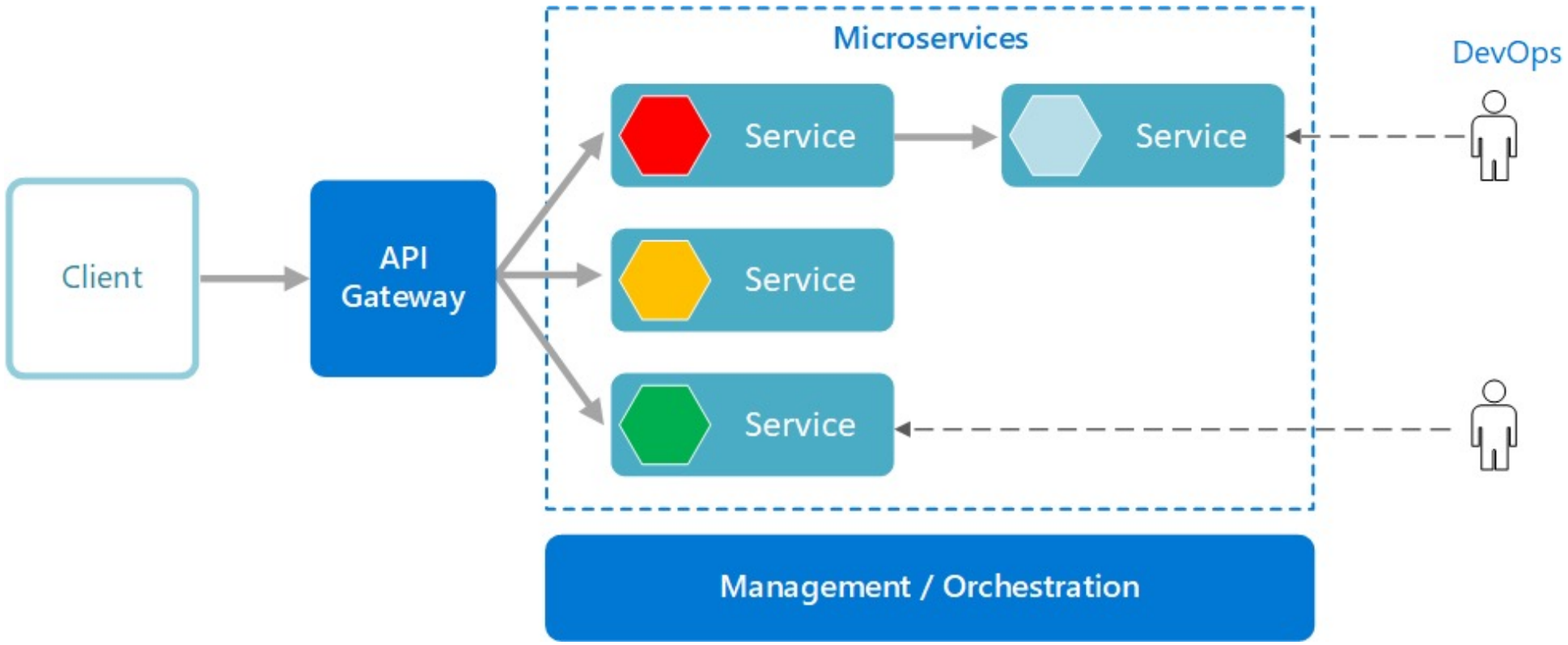
Peer to peer architecture



Pipe and filter architecture

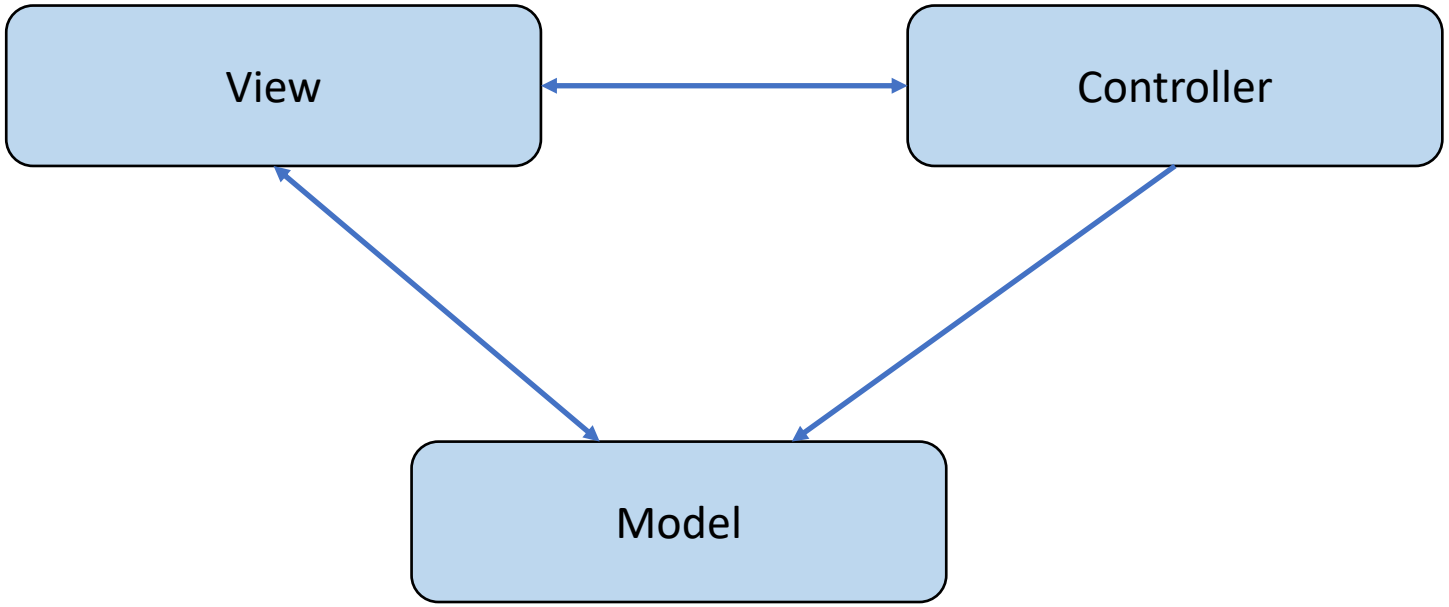


Microservices architecture



[https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices]

Model-view-controller



Key points

- Software architectures are designed taking into account functional and non-functional requirements of the system
- Architecture design decisions should factor in type of system, distribution and interaction of its elements, appropriate styles, etc
- Software architectures must be documented from different perspectives using notations appropriate for relevant stakeholders
- Software architectures can and should be used in agile as well as plan-driven development scenarios
- Architecture styles provide proven solutions to system contexts
- Styles have to be chosen to address architecturally significant requirements at each stage
- Typically each system will require a combination of styles to address all the requirements