Material and some slide content from:
- Emerson Murphy-Hill, Reid Holmes
- Software Architecture: Foundations, Theory, and Practice
- Essential Software Architecture

Architectural Style Intro & Early Feedback

Mei Nagappan

Pitch Survey

- Your group name
- Most interesting
- Most useful

Early Feedback

- What can be improved/done differently?
- What do you dislike the most?
- What do you like?

Attribute Driven Design

- Choose module to decompose
  - Initially whole system is one module
- Refine the module
  - Choose arch drivers from NFR and FR
  - Choose arch style that satisfies them
  - Create modules based on style
  - Allocate functionality to each module
  - Define interfaces for modules
  - Verify and evaluate against NFR and FR
- Repeat until you cannot decompose

Architectural styles

- Some design choices are better than others
  - Experience can guide us towards beneficial sets of choices (patterns) that have positive properties
- An architectural style is a named collection of architectural design decisions that:
  - Are applicable to a given context
  - Constrain design decisions
  - Elicit beneficial qualities in resulting systems

[TAILOR ET AL.]

Architectural styles

A set of architectural design decisions that are applicable to a recurring design problem, and parameterized to account for different software development contexts in which that problem appears.

e.g., Three-tier architectural pattern:

Architectural styles

- Defines a family of architectures that are constrained by:
  - Component/connector vocabulary
  - Topology
  - Semantic constraints
- When describing styles diagrammatically:
  - Nodes == components (e.g., procedures, modules, processes, databases, …)
  - Edges == connectors (e.g., procedure calls, events, db queries, pipes, …)

[CZARNECKI]

Good properties of an architecture

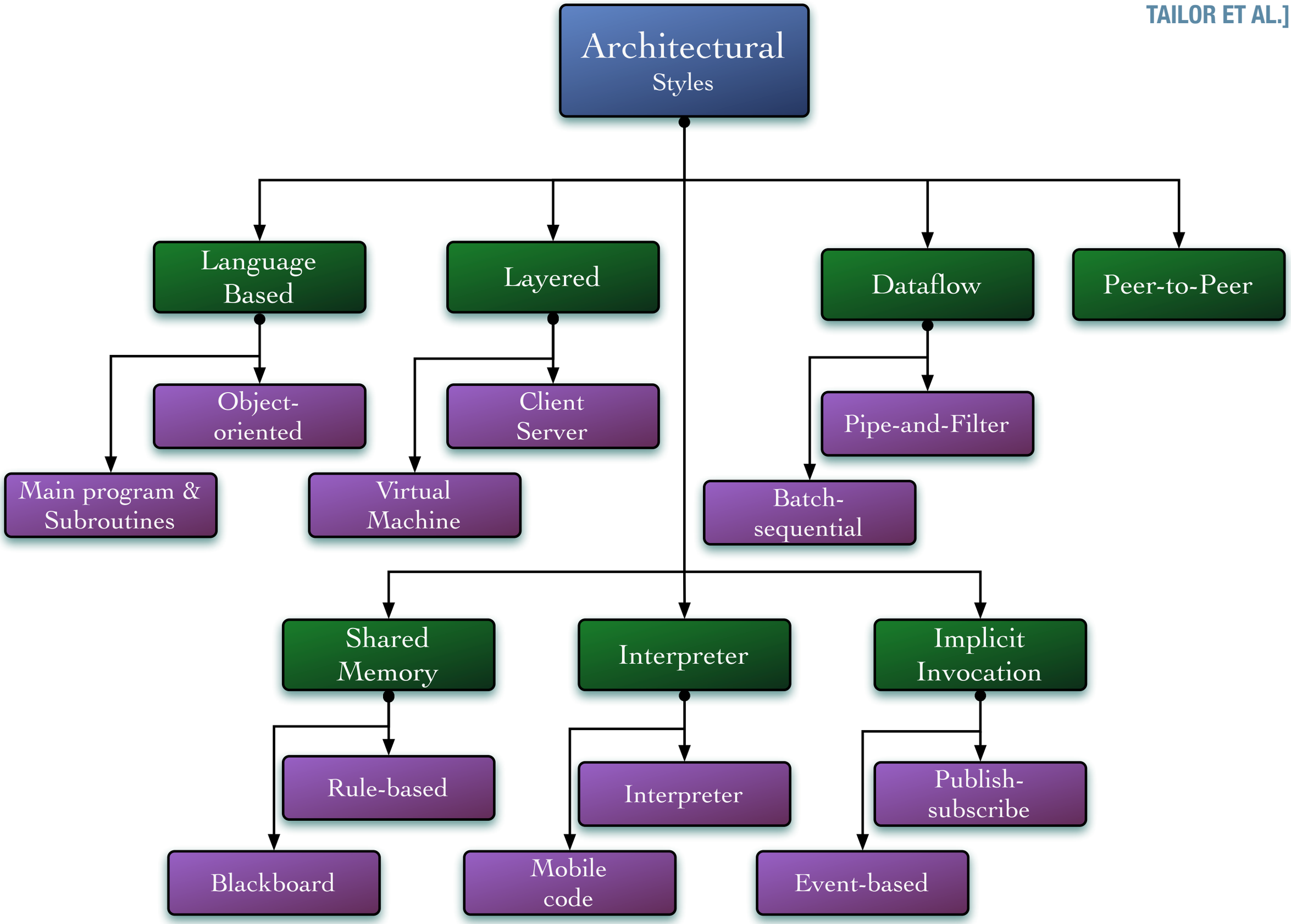
- Result in a consistent set of principled techniques
- Resilient in the face of (inevitable) changes
- Source of guidance through product lifetime
- Reuse of established engineering knowledge

[CZARNECKI]

“Pure” architectural styles

- Pure architectural styles are rarely used in practice
- Systems in practice:
  - Regularly deviate from pure styles.
  - Typically feature many architectural styles.
- Architects must understand the “pure” styles to understand the strength and weaknesses of the style as well as the consequences of deviating from the style.

[TOPOLOGY FROM TAILOR ET AL.]



Arch Activity

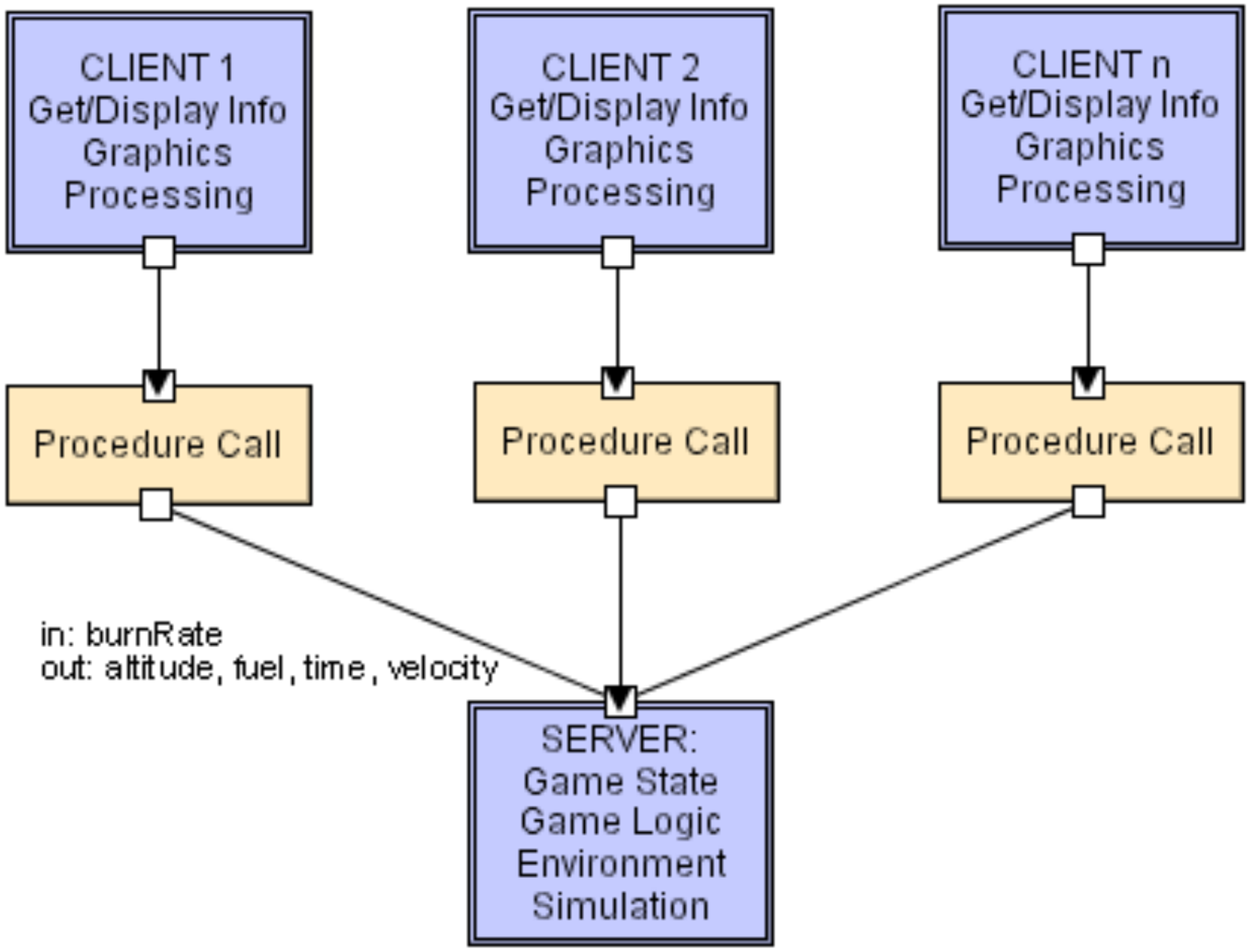
Presentation

- 20 minute maximum
- Two primary components:
  - Static description of the style / pattern
  - Dynamic description of how the style / pattern is useful over time
- Comprehensive example / tutorial that demonstrates the style / pattern
- No slides; be creative. Make it memorable.

Arch Style

- Have its own vocabulary for its components and connectors? (define)
- Impose specific topological constraints? (diagram)
- Most applicable to specific kinds of problems?
- Engender specific kinds of change resilience?
- Have any specific negative behaviours?
- Support/inhibit specific NFPs?

Style: Client-server



Style: Client-server

- Clients communicate with server which performs actions and returns data. Client initiates communication.
- Components:
  - Clients and server.
- Connections:
  - Protocols, RPC.
- Data elements:
  - Parameters and return values sent / received by connectors.
- Topology:
  - Two level. Typically many clients.

[TAILOR ET AL.]

Style: Client-server

- Additional constraints:
  - Clients cannot communicate with each other.
- Qualities:
  - Centralization of computation. Server can handle many clients.
- Typical uses:
  - Applications where: client is simple; data integrity important; computation expensive.
- Cautions:
  - Bandwidth and lag concerns.