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

Architectural Styles

Reid Holmes

Lecture 5 - Tuesday, September 28 2010.

Objectives

- What are the benefits / pitfalls of different architectural approaches?
- What are the phases of the design process?
- What are some alternative design strategies? When are they necessary?
- Define: abstraction, reification, and SoC.
- Differentiate arch styles, patterns, and DSSAs.
- Identify key architectural style categories.

Architectural approaches

- Creative
  - Engaging
  - Dangerous
  - Potentially unnecessary or best
- Methodical
  - Efficient when domain is familiar
  - Not always successful
  - Predictable outcome

Design process

1. Feasibility stage:
   - Identify set of feasible concepts.
2. Preliminary design stage:
   - Select and develop best concept.
3. Detailed design stage:
   - Develop engineering descriptions of concept.
4. Planning stage:
   - Evaluate / alter concept to fit requirements. Team allocation / budgeting.

Design strategies

- Standard
- Cyclic
  - Revisit earlier stages.
- Parallel
  - Split off #2 or #1 in parallel.
- Adaptive
  - Plan next stage with insights from current.
- Incremental
  - Update all stages as experience is gained.

Abstraction

Definition:

> “A concept or idea not associated with a specific instance.”

Top down

> ?

Bottom up

> ?

Reification:

> “The conversion of a concept into a thing.”

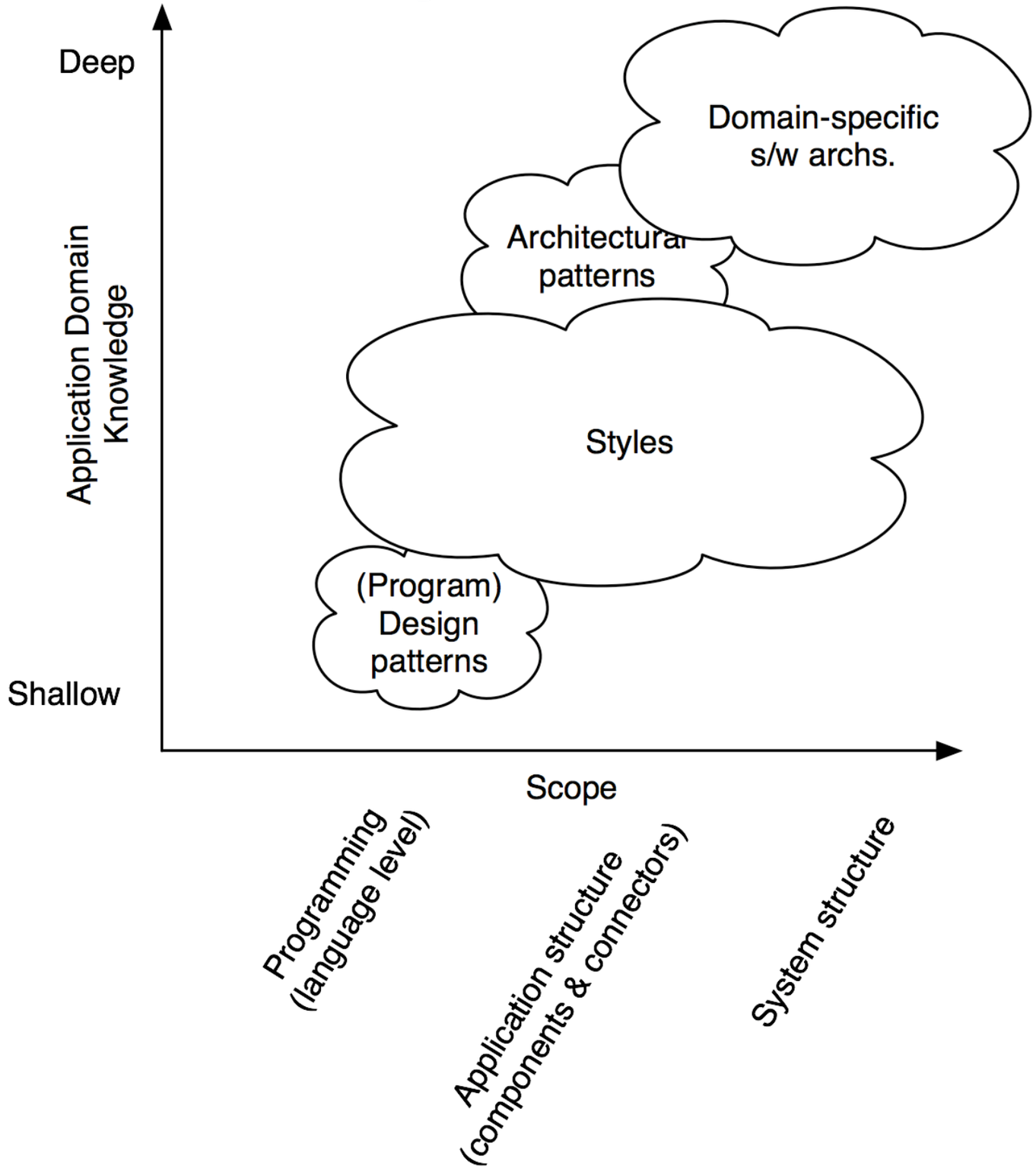
Level of discourse

- Consider application as a whole.
  - e.g., stepwise refinement.
- Start with sub-problems.
  - Combine solutions as they are ready.
- Start with level above desired application.
  - e.g., consider simple input as general parsing.

Separation of Concerns

- Decomposition of problem into independent parts.
- In arch, separating components and connectors.
- Complicated by:
  - Scattering:
    - Concern spread across many parts.
      - e.g., logging.
  - Tangling:
    - Concern interacts with many parts.
      - e.g., performance.

Patterns, Styles, and DSSAs



[TAILOR ET AL.]

DSSAs

- Domain-specific software architectures:
  - Specialized for a particular task (domain).
  - Generalized for effective use in that domain.
  - Composed using a standardized topology.
- Key benefit: maximal reuse of knowledge.
- Key drawback: only applicable in specific domain.

Architectural patterns

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.

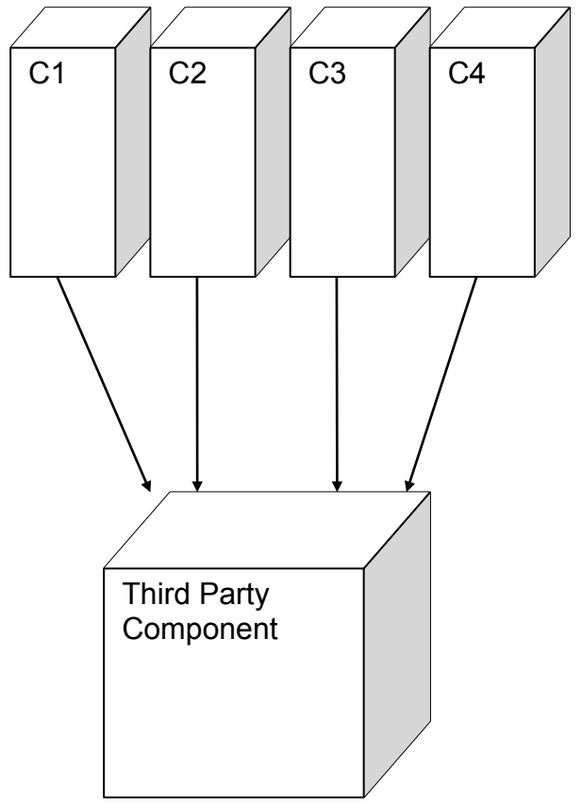
e.g., Three-tier architectural pattern:

Architectural styles

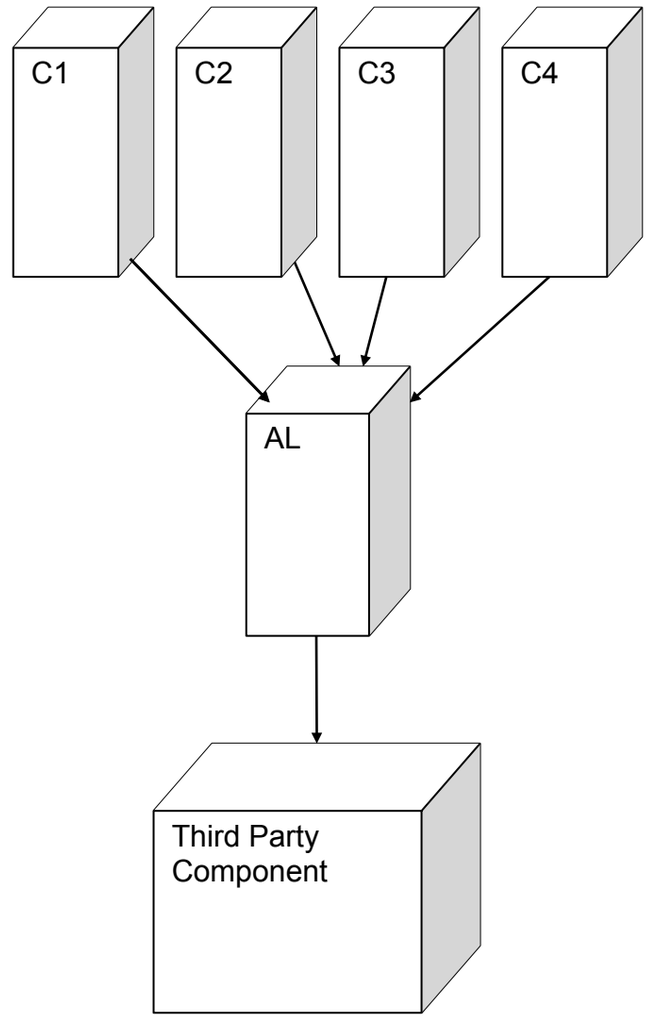
- Some design choices are better than others.
  - Experience can guide us towards beneficial sets of choices (patterns) that have positive properties.
    - Such as?
- An architectural style is a named collection of architectural design decisions that:
  - ?
  - ?
  - ?

Structure and Dependencies

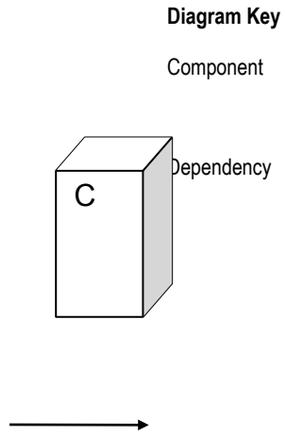
- Excessive dependencies are not a good idea.
- Key issue:
  - ?
- Reduce direct dependencies on these.



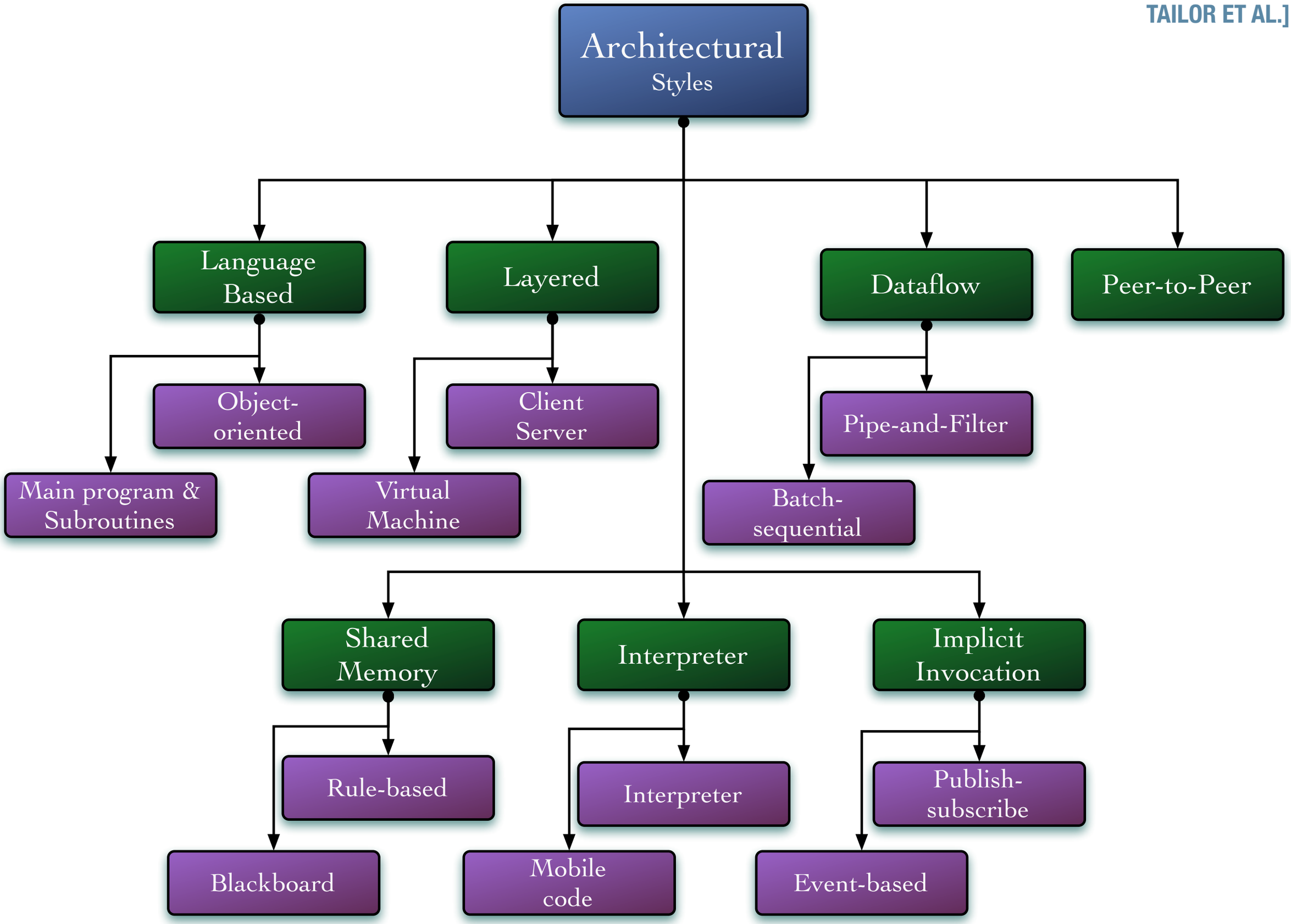
Four components are directly dependent on a third party component. If the third party component is replaced with a new component with a different interface, changes to each component are likely.



Only the AL (abstraction layer) component is directly dependent on the third party component. If the third party component is replaced, changes are restricted to the AL component only



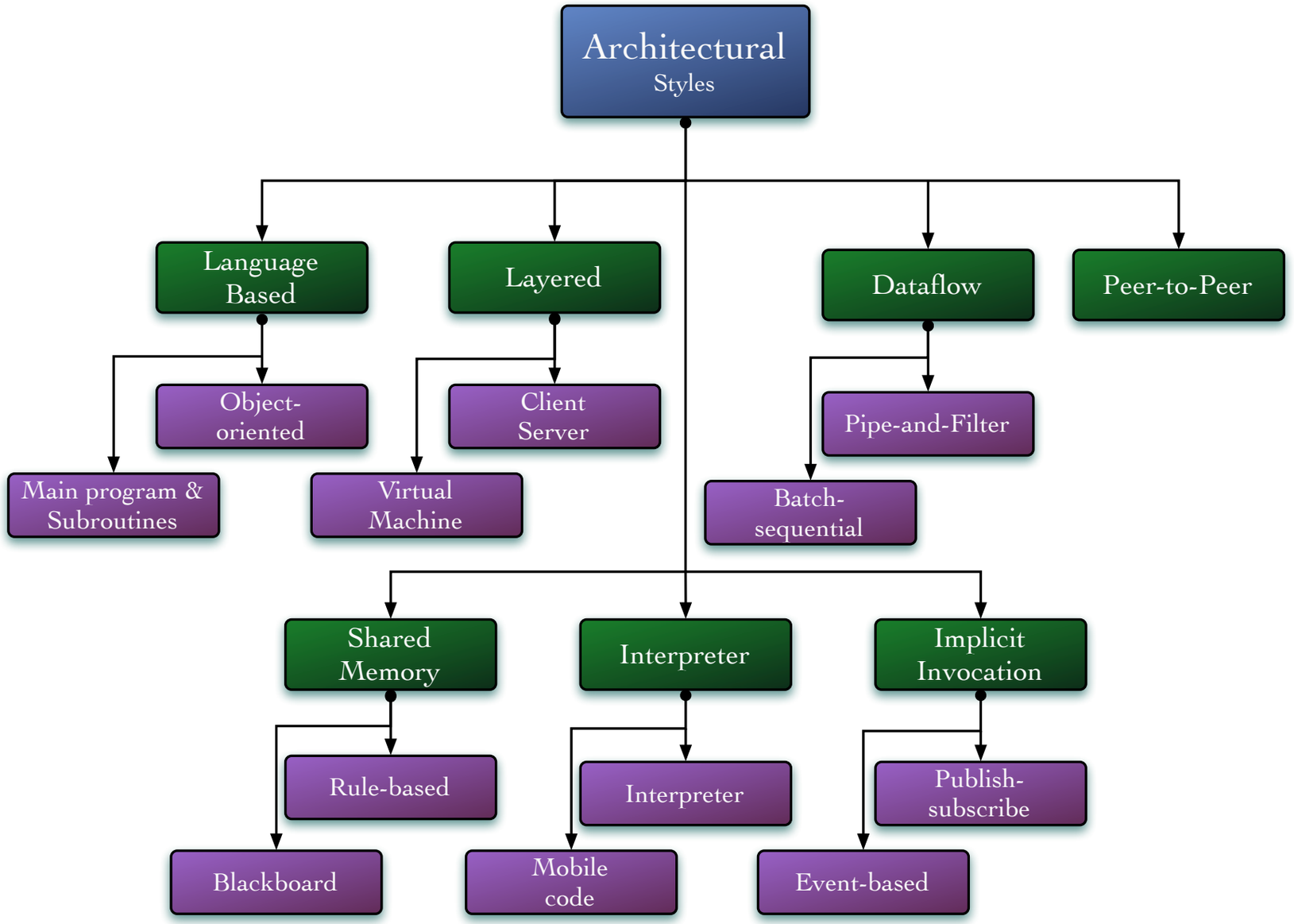
[TOPOLOGY FROM TAILOR ET AL.]



[TOPOLOGY FROM TAILOR ET AL.]

Activity

- Design Facebook using an assigned pattern.
  - What are the components, connectors, and topology?



Activity followup

- Discussion revealed that designing FB using:
  - P2P is not a good idea.
  - Straight OO would be awkward.
  - Layered could also work but not performant.
  - Blackboard and publish/subscribe fit well.
    - Likely a hybrid of these would work best.