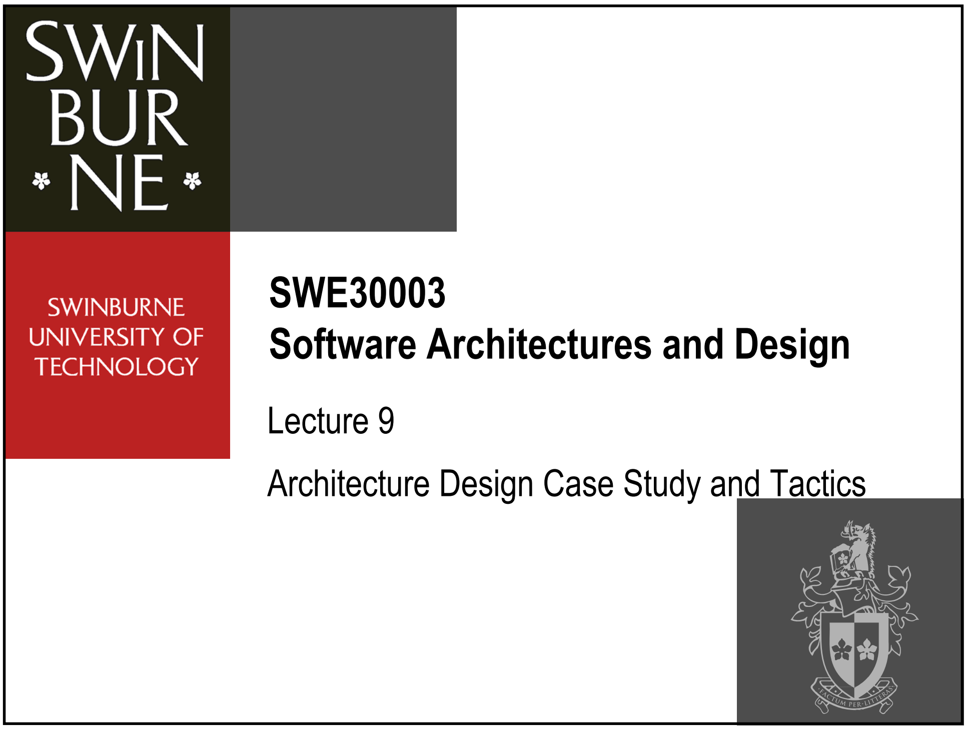


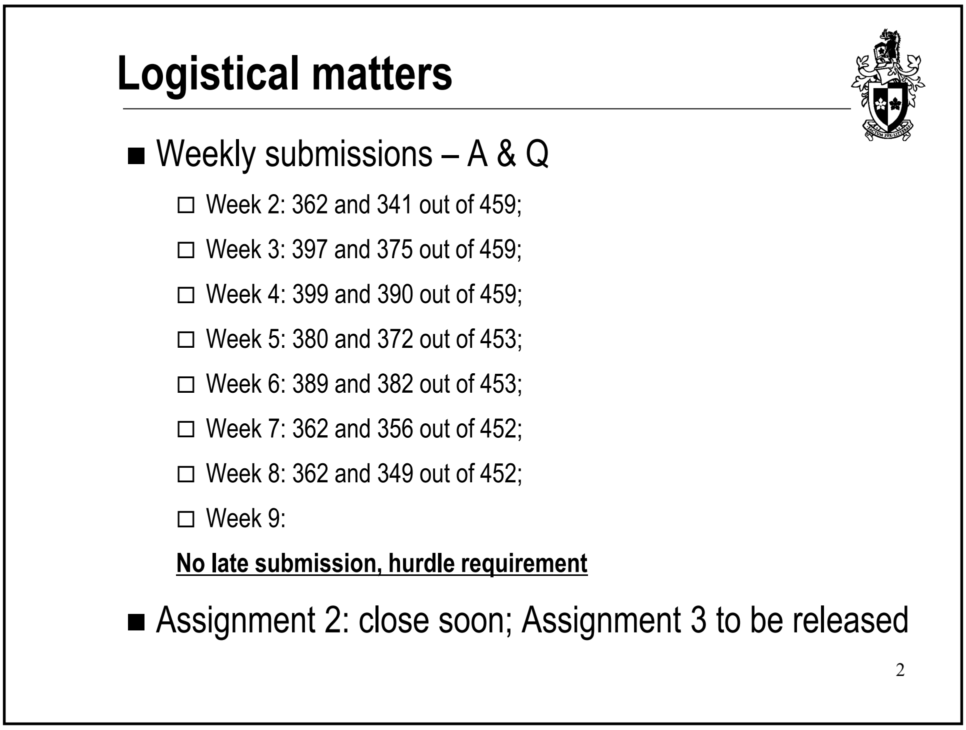
SWINBURNE UNIVERSITY OF TECHNOLOGY

SWE30003 Software Architectures and Design

Lecture 9

Architecture Design Case Study and Tactics

1



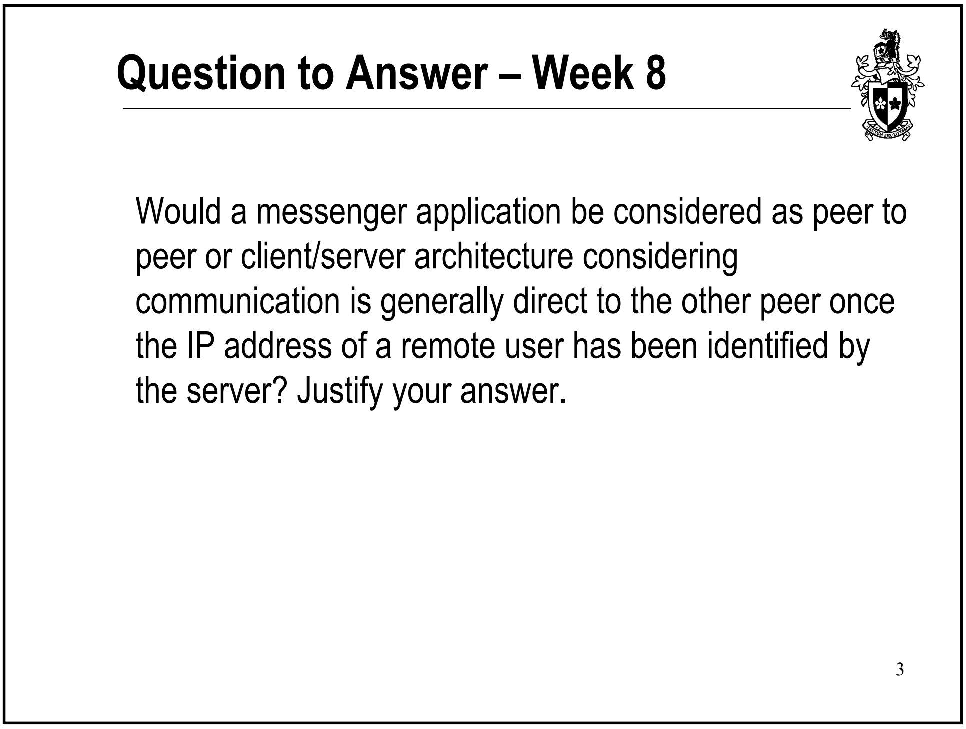
Logistical matters

- Weekly submissions – A & Q
  - □ Week 2: 362 and 341 out of 459;
  - □ Week 3: 397 and 375 out of 459;
  - □ Week 4: 399 and 390 out of 459;
  - □ Week 5: 380 and 372 out of 453;
  - □ Week 6: 389 and 382 out of 453;
  - □ Week 7: 362 and 356 out of 452;
  - □ Week 8: 362 and 349 out of 452;
  - □ Week 9:

  No late submission, hurdle requirement
- Assignment 2: close soon; Assignment 3 to be released

2

2

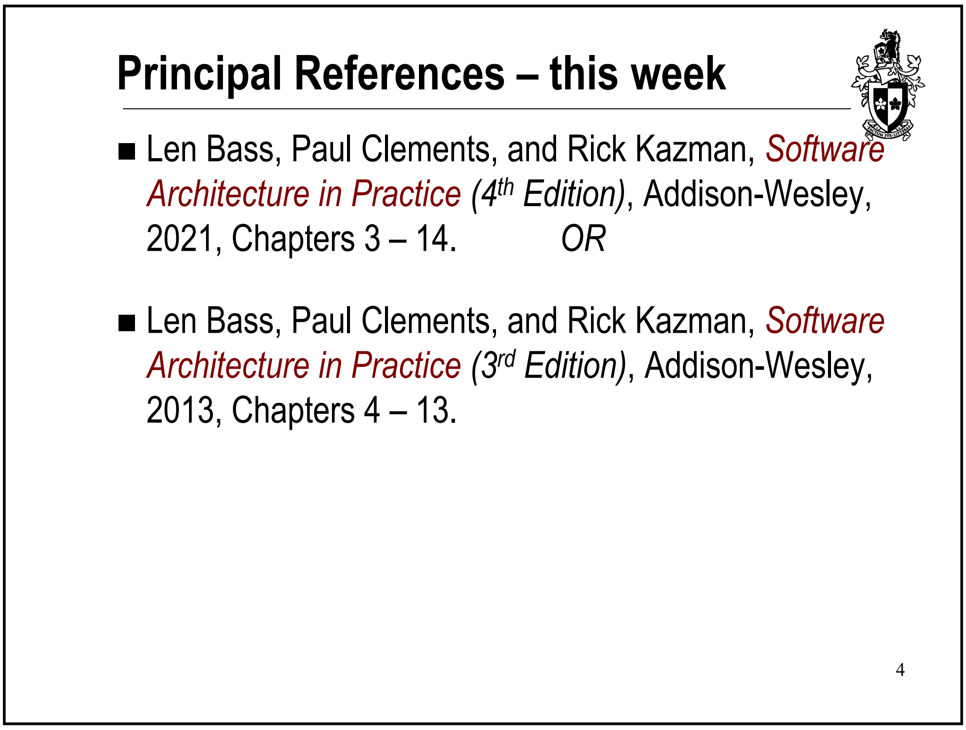


Question to Answer – Week 8

Would a messenger application be considered as peer to peer or client/server architecture considering communication is generally direct to the other peer once the IP address of a remote user has been identified by the server? Justify your answer.

3

3

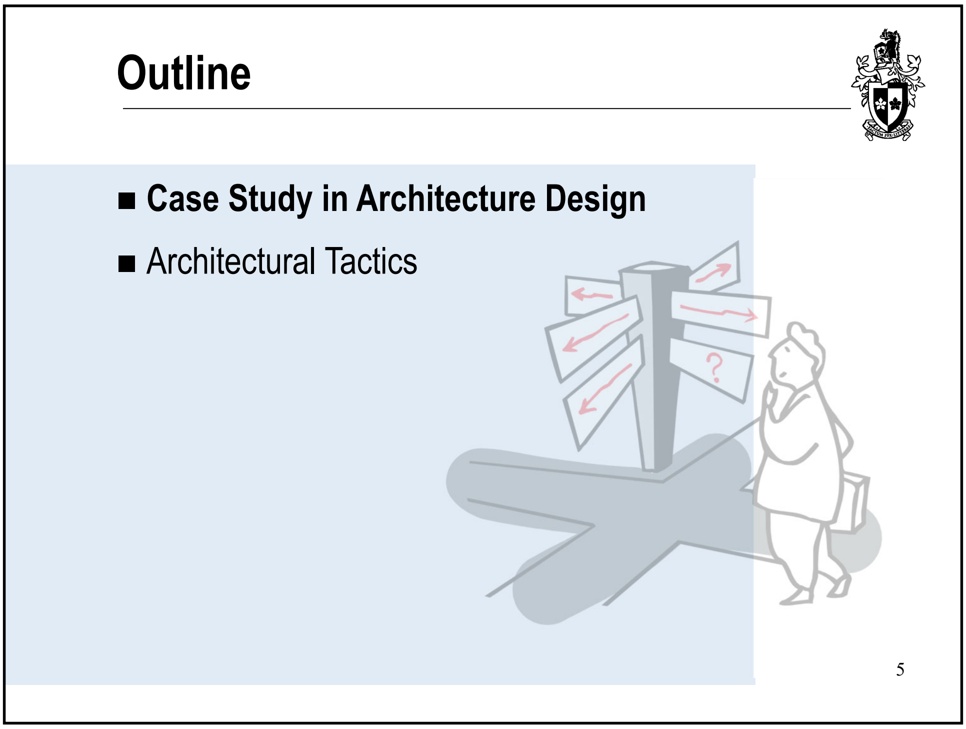


Principal References – this week

- Len Bass, Paul Clements, and Rick Kazman, *Software Architecture in Practice (4th Edition)*, Addison-Wesley, 2021, Chapters 3 – 14. *OR*
- Len Bass, Paul Clements, and Rick Kazman, *Software Architecture in Practice (3rd Edition)*, Addison-Wesley, 2013, Chapters 4 – 13.

4

4

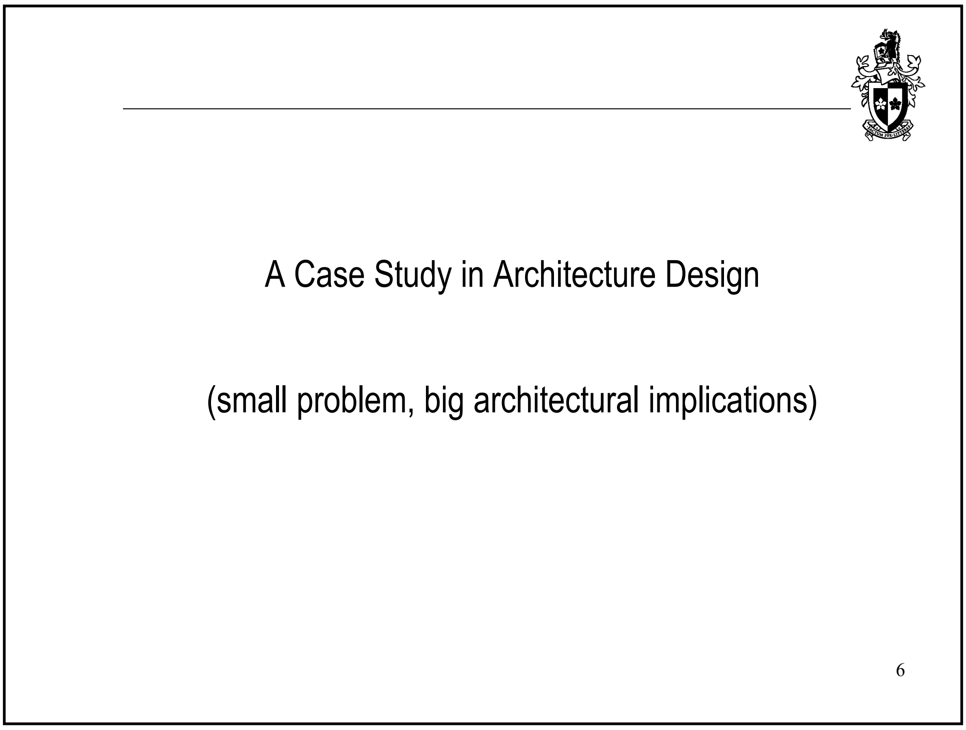


Outline

- ■ **Case Study in Architecture Design**
- ■ Architectural Tactics

5

5

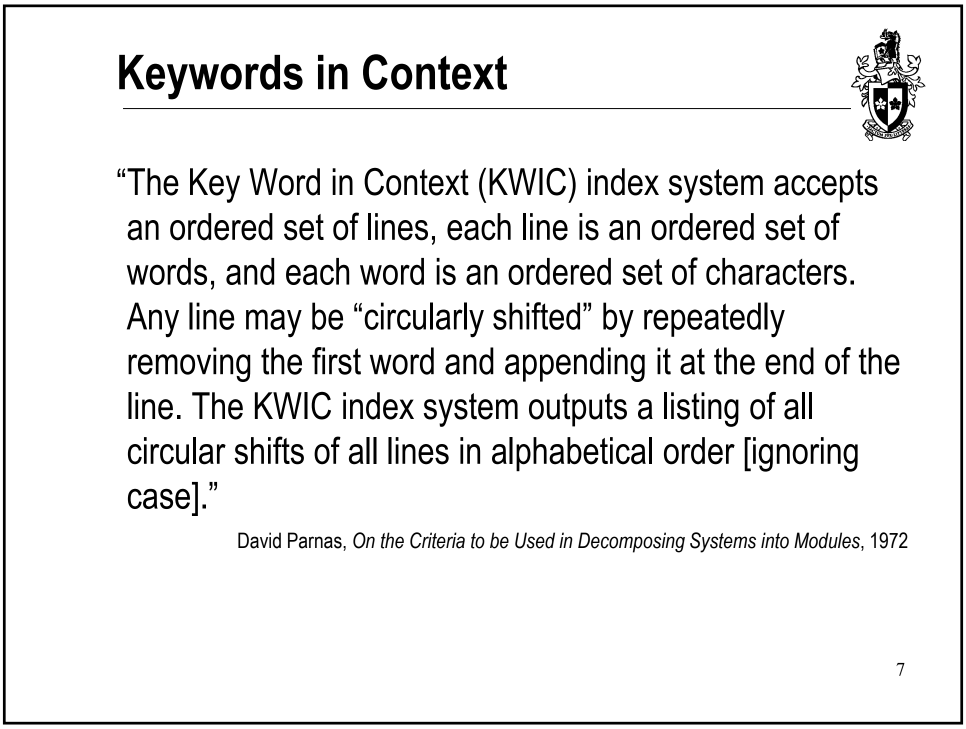


A Case Study in Architecture Design

(small problem, big architectural implications)

6

6

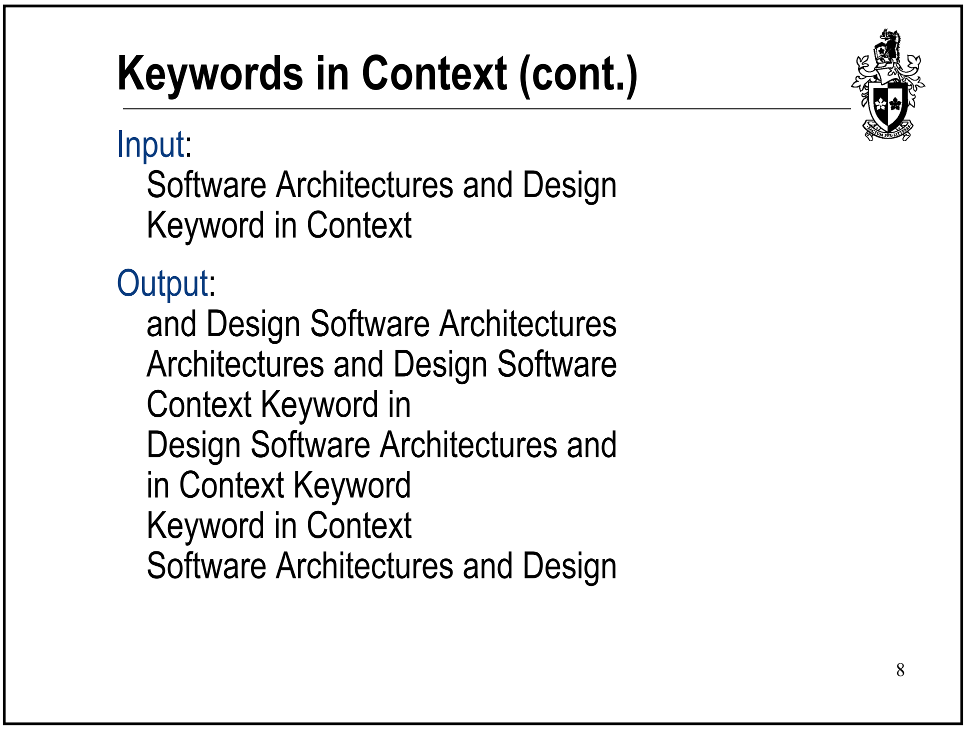


Keywords in Context

"The Key Word in Context (KWIC) index system accepts an ordered set of lines, each line is an ordered set of words, and each word is an ordered set of characters. Any line may be "circularly shifted" by repeatedly removing the first word and appending it at the end of the line. The KWIC index system outputs a listing of all circular shifts of all lines in alphabetical order [ignoring case]."

David Parnas, *On the Criteria to be Used in Decomposing Systems into Modules*, 1972

7



Keywords in Context (cont.)

Input:

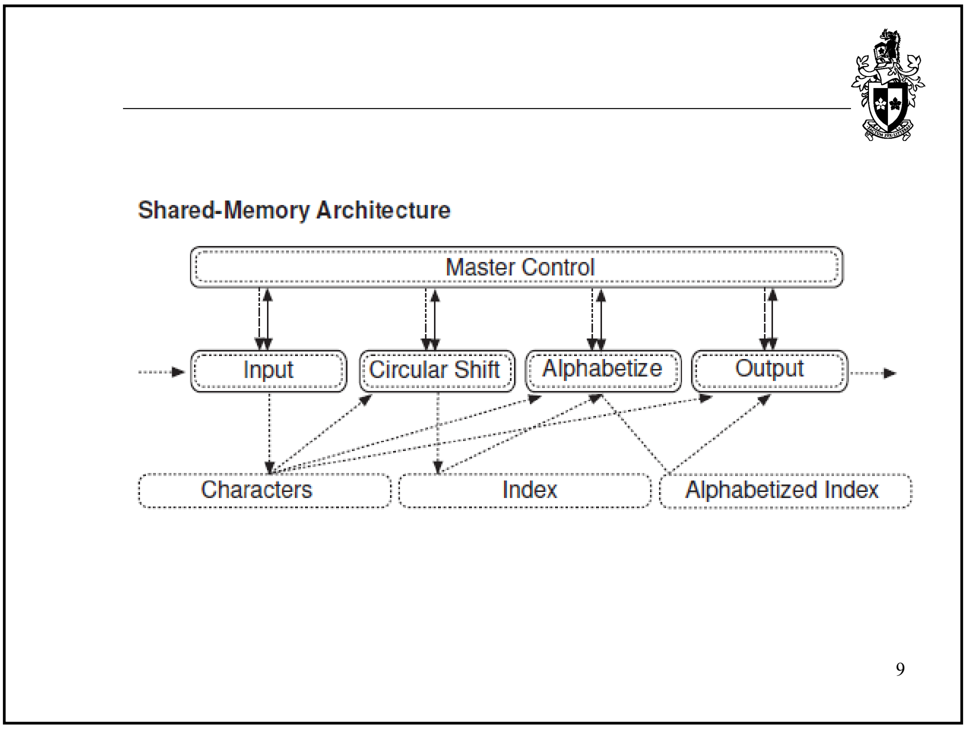
Software Architectures and Design
Keyword in Context

Output:

and Design Software Architectures
Architectures and Design Software
Context Keyword in
Design Software Architectures and
in Context Keyword
Keyword in Context
Software Architectures and Design

8

© Swinburne University of Technology



Shared-Memory Architecture

Master Control

Input

Circular Shift

Alphabetize

Output

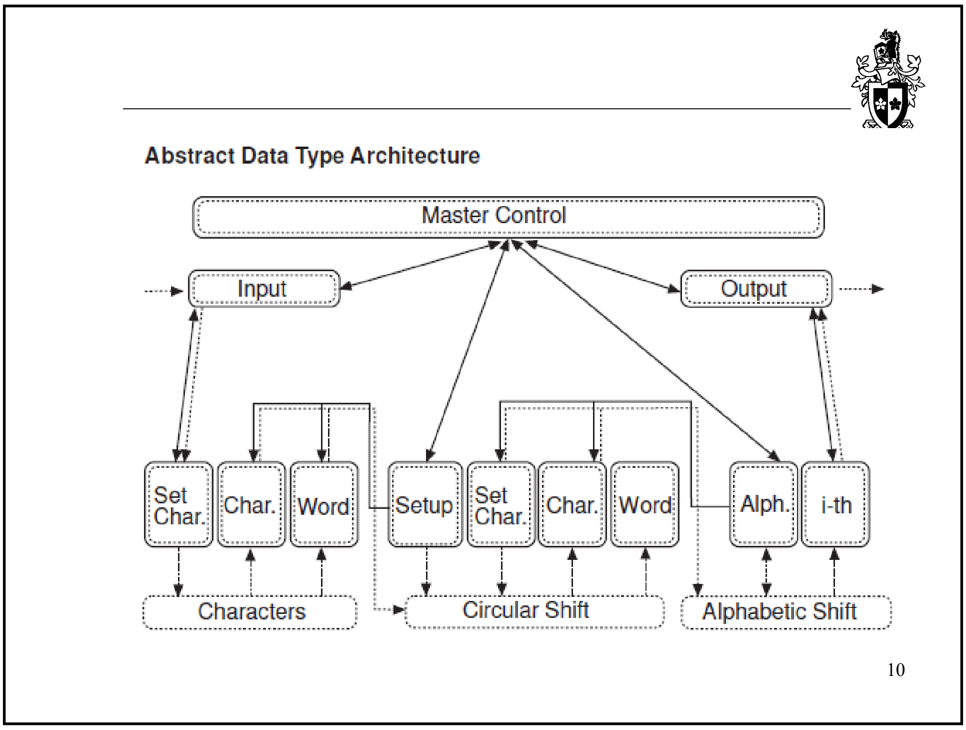
Characters

Index

Alphabetized Index

9

9



Abstract Data Type Architecture

Master Control

Input

Output

Set Char.

Char.

Word

Setup

Set Char.

Char.

Word

Alph.

i-th

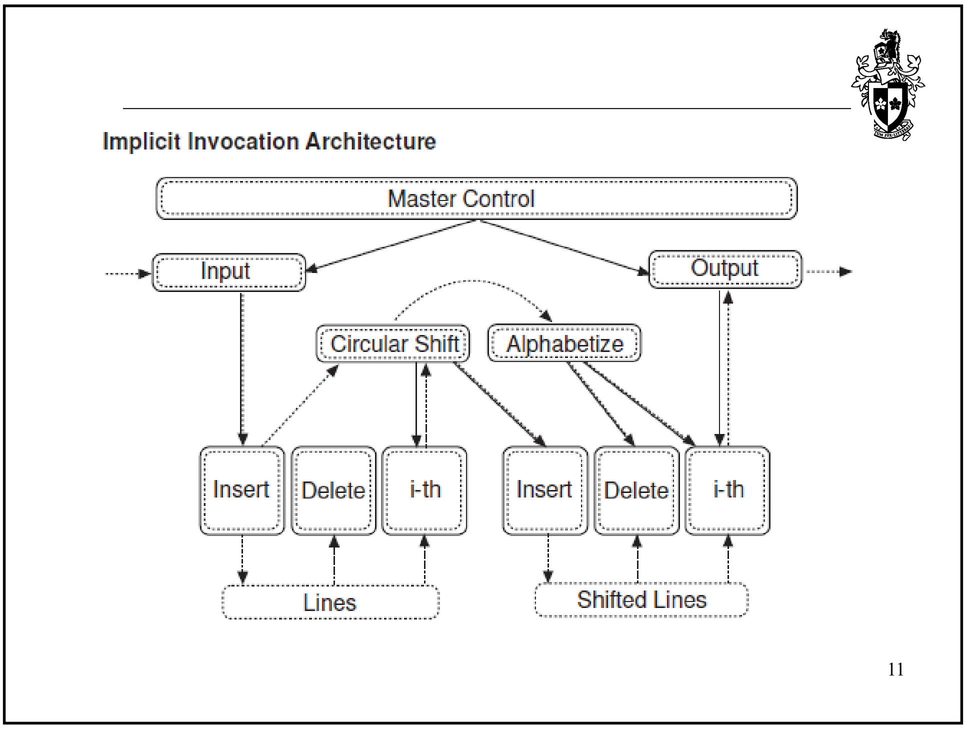
Characters

Circular Shift

Alphabetic Shift

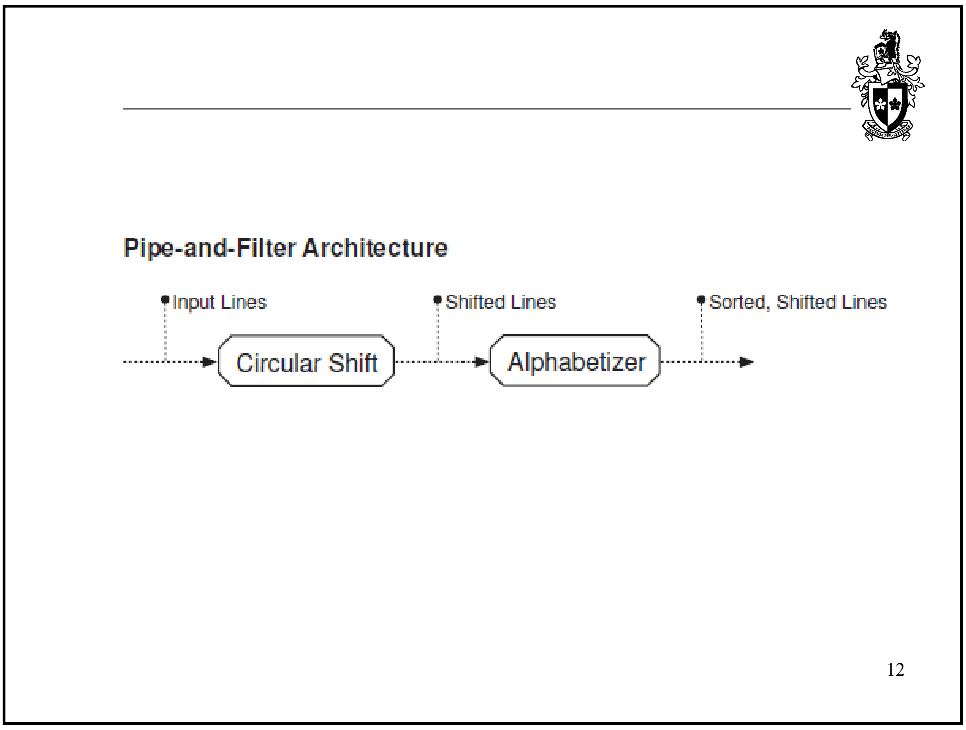
10

10



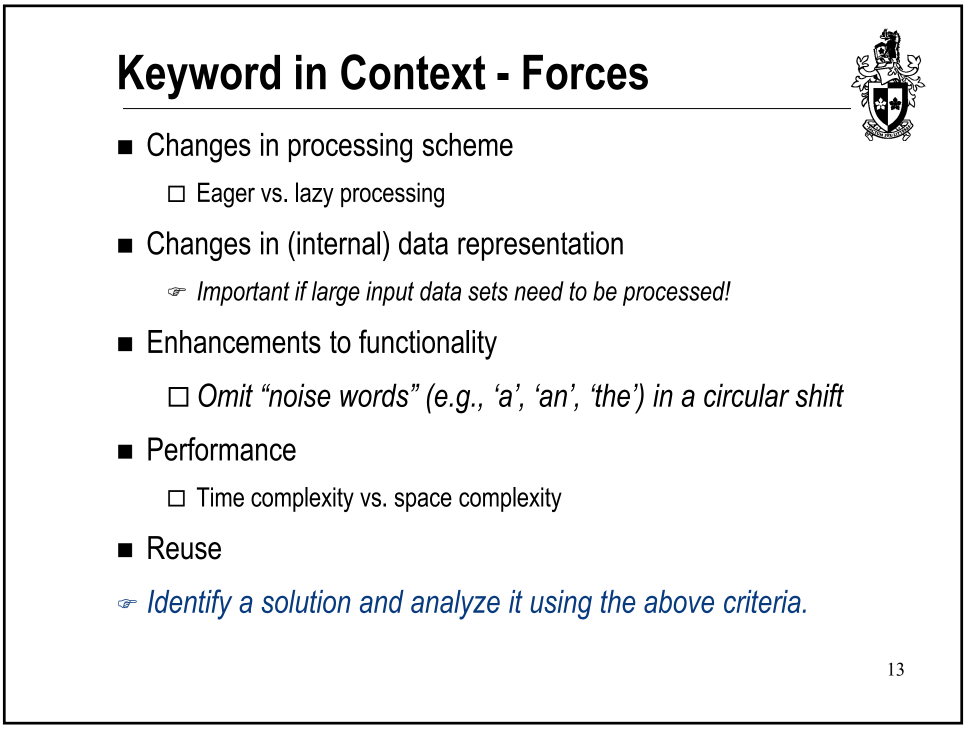
Implicit Invocation Architecture

11



Pipe-and-Filter Architecture

12



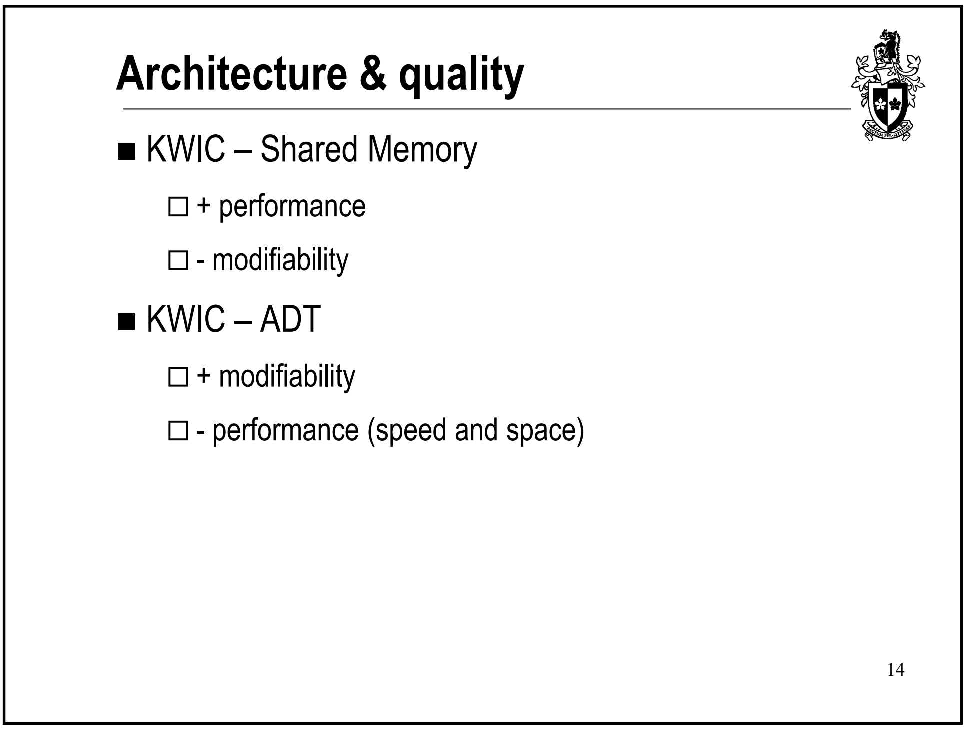
Keyword in Context - Forces

- Changes in processing scheme
  - □ Eager vs. lazy processing
- Changes in (internal) data representation
  - ☞ *Important if large input data sets need to be processed!*
- Enhancements to functionality
  - □ *Omit "noise words" (e.g., 'a', 'an', 'the') in a circular shift*
- Performance
  - □ Time complexity vs. space complexity
- Reuse

☞ *Identify a solution and analyze it using the above criteria.*

13

13

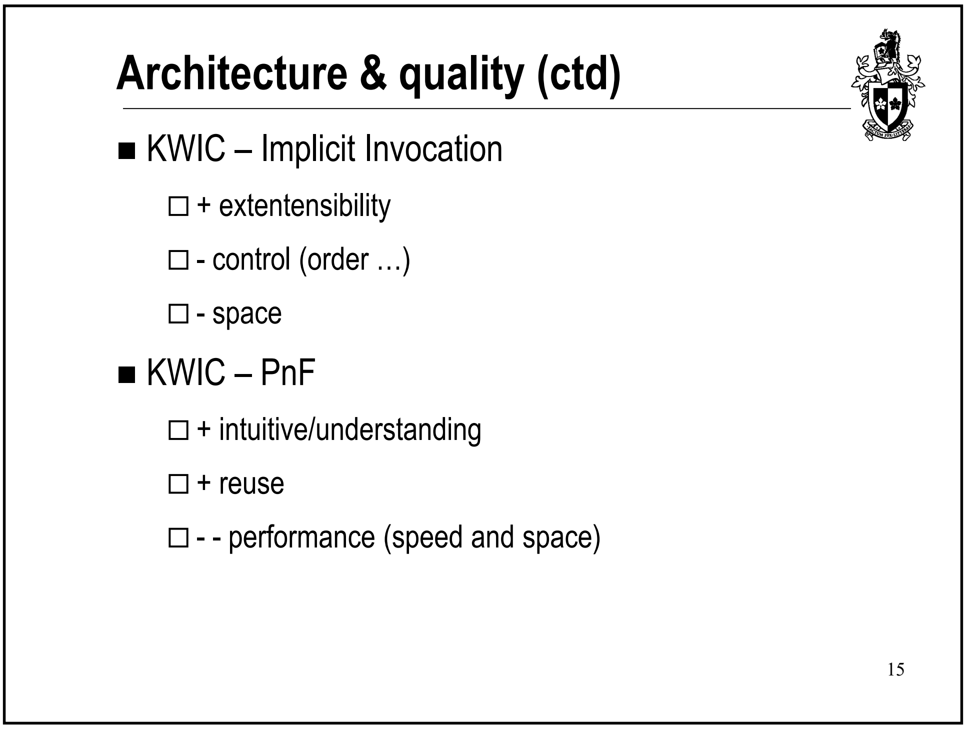


Architecture & quality

- KWIC – Shared Memory
  - □ + performance
  - □ - modifiability
- KWIC – ADT
  - □ + modifiability
  - □ - performance (speed and space)

14

14

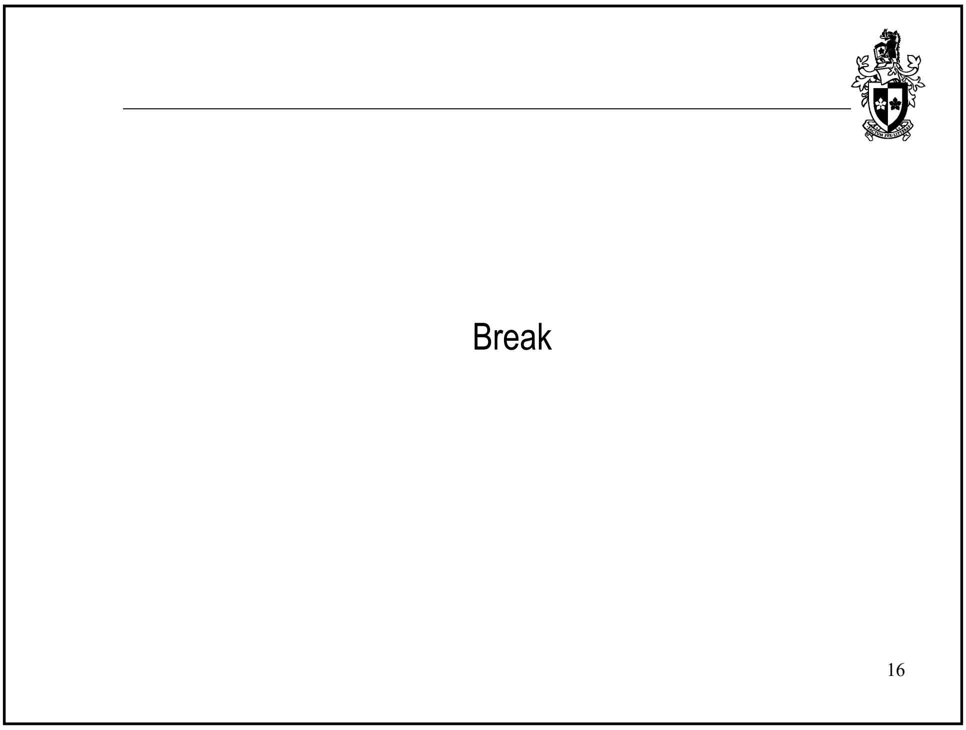


Architecture & quality (ctd)

- KWIC – Implicit Invocation
  - □ + extentensibility
  - □ - control (order …)
  - □ - space
- KWIC – PnF
  - □ + intuitive/understanding
  - □ + reuse
  - □ - - performance (speed and space)

15

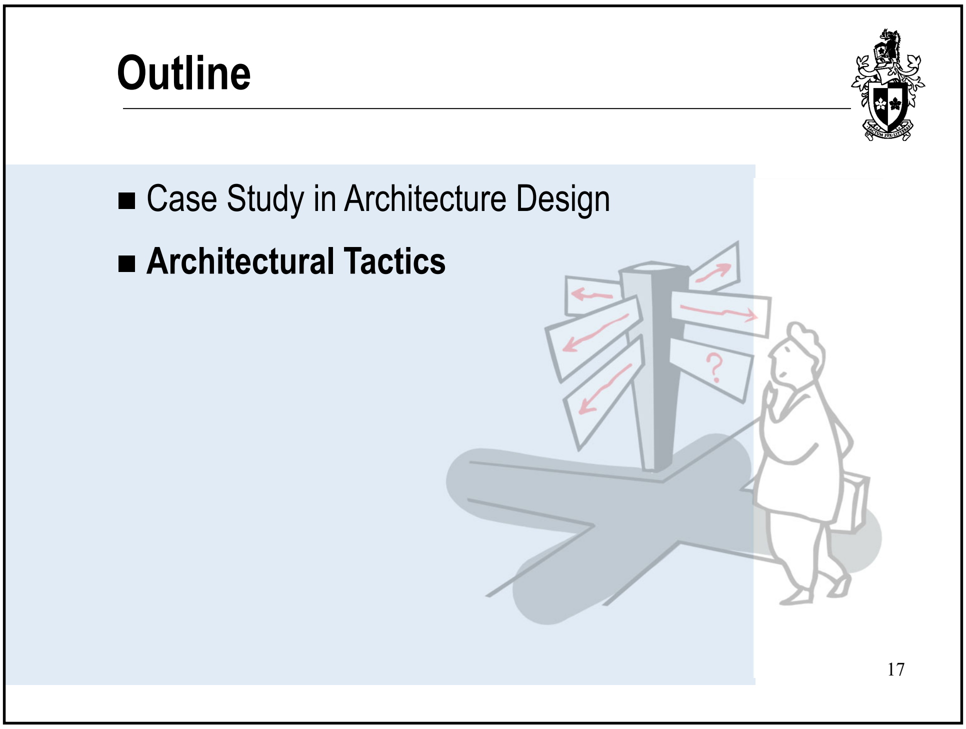
15



Break

16

16

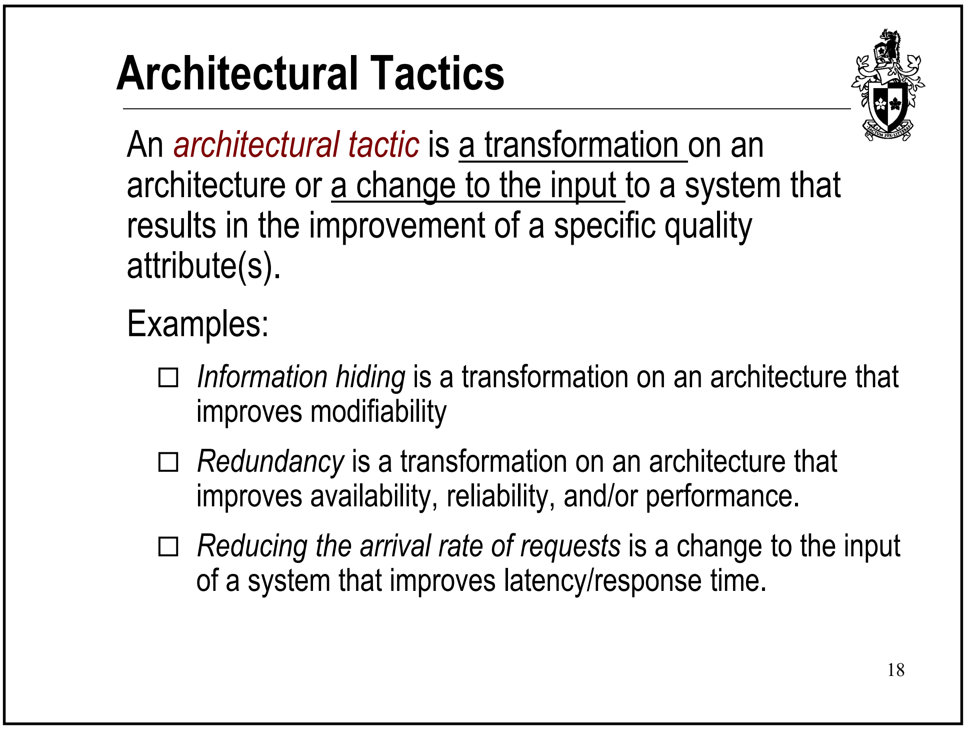


Outline

- Case Study in Architecture Design
- **Architectural Tactics**

17

17



Architectural Tactics

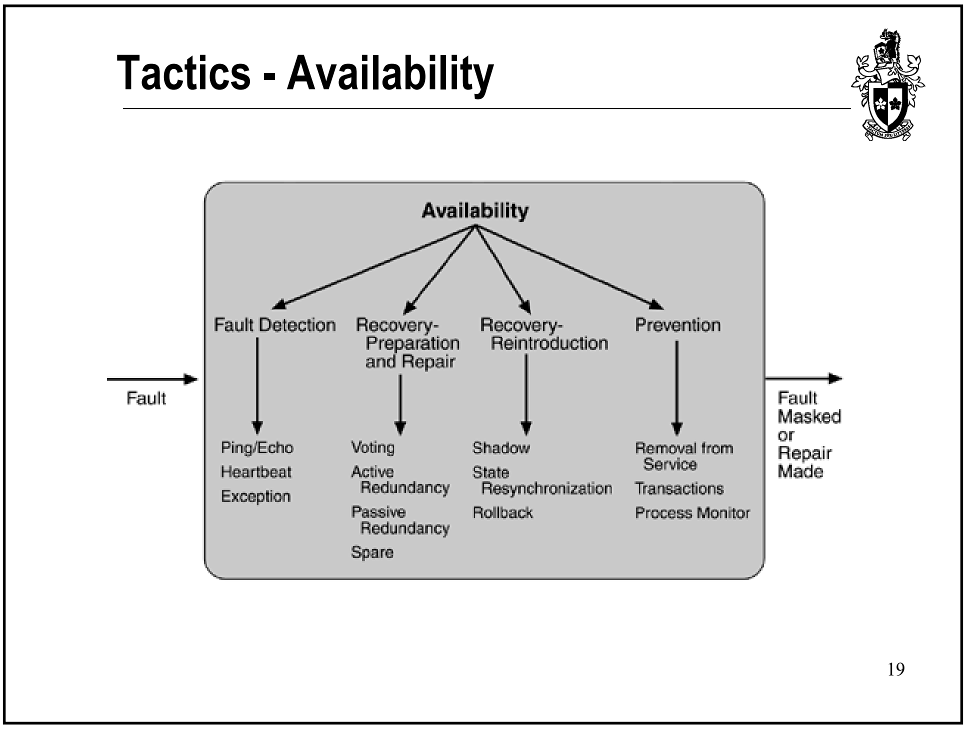
An *architectural tactic* is a transformation on an architecture or a change to the input to a system that results in the improvement of a specific quality attribute(s).

Examples:

- □ *Information hiding* is a transformation on an architecture that improves modifiability
- □ *Redundancy* is a transformation on an architecture that improves availability, reliability, and/or performance.
- □ *Reducing the arrival rate of requests* is a change to the input of a system that improves latency/response time.

18

18



Tactics - Availability

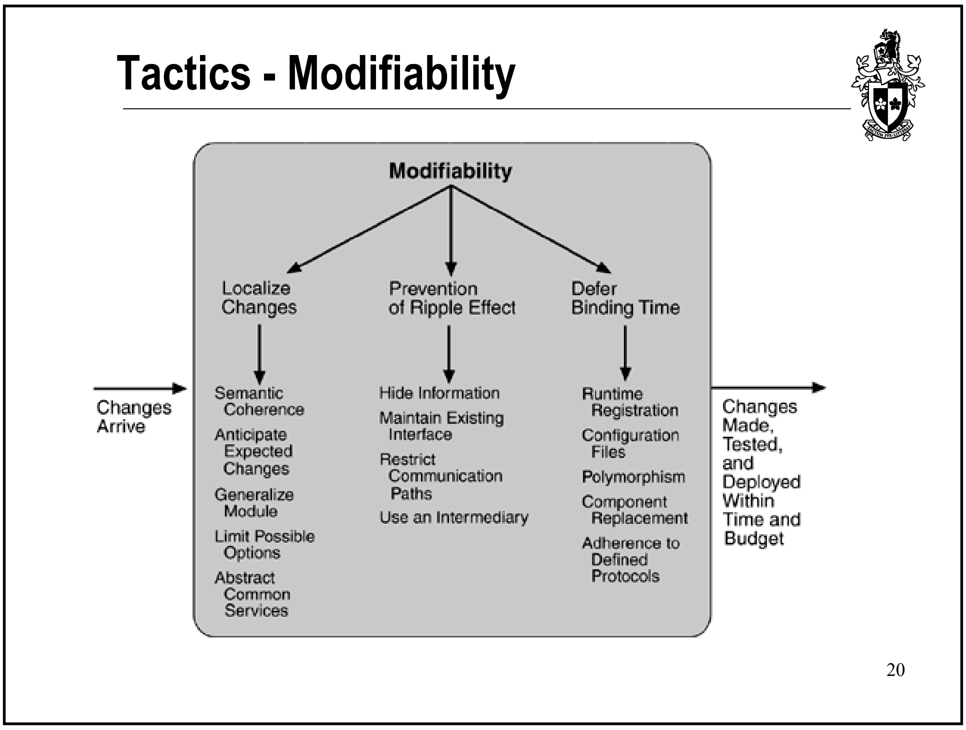
Availability

- Fault Detection
  - Ping/Echo
  - Heartbeat
  - Exception
- Recovery-Preparation and Repair
  - Voting
  - Active Redundancy
  - Passive Redundancy
  - Spare
- Recovery-Reintroduction
  - Shadow
  - State Resynchronization
  - Rollback
- Prevention
  - Removal from Service
  - Transactions
  - Process Monitor

Fault → Availability → Fault Masked or Repair Made

19

19



Tactics - Modifiability

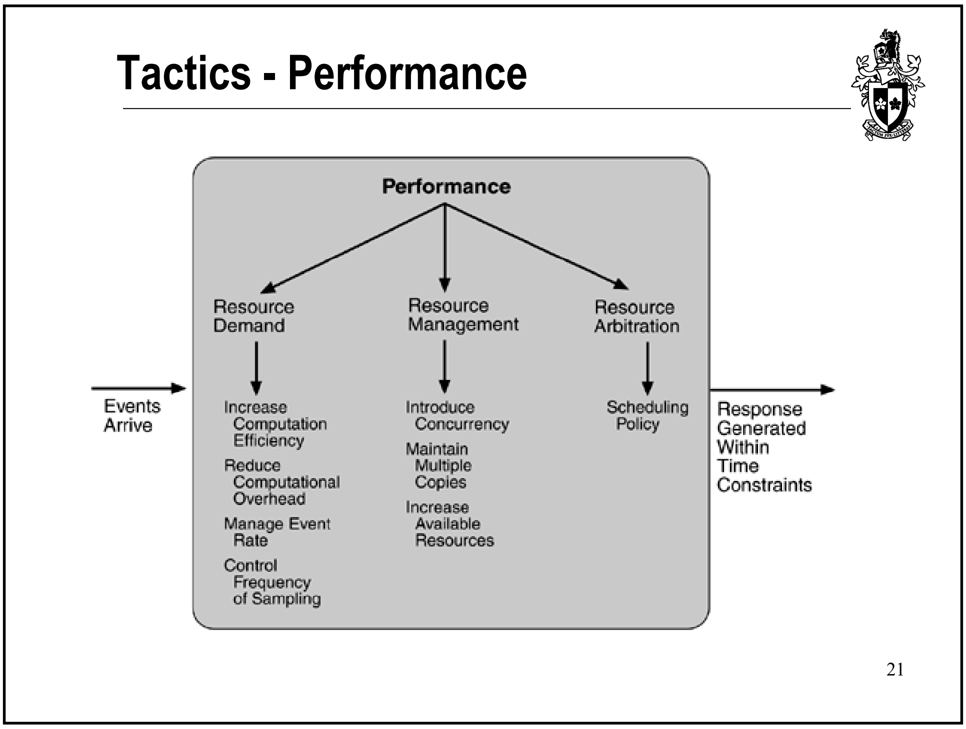
Modifiability

- Localize Changes
  - Semantic Coherence
  - Anticipate Expected Changes
  - Generalize Module
  - Limit Possible Options
  - Abstract Common Services
- Prevention of Ripple Effect
  - Hide Information
  - Maintain Existing Interface
  - Restrict Communication Paths
  - Use an Intermediary
- Defer Binding Time
  - Runtime Registration
  - Configuration Files
  - Polymorphism
  - Component Replacement
  - Adherence to Defined Protocols

Changes Arrive → Modifiability → Changes Made, Tested, and Deployed Within Time and Budget

20

20



Tactics - Performance

Performance

Resource Demand

Resource Management

Resource Arbitration

Events Arrive

Increase Computation Efficiency

Reduce Computational Overhead

Manage Event Rate

Control Frequency of Sampling

Introduce Concurrency

Maintain Multiple Copies

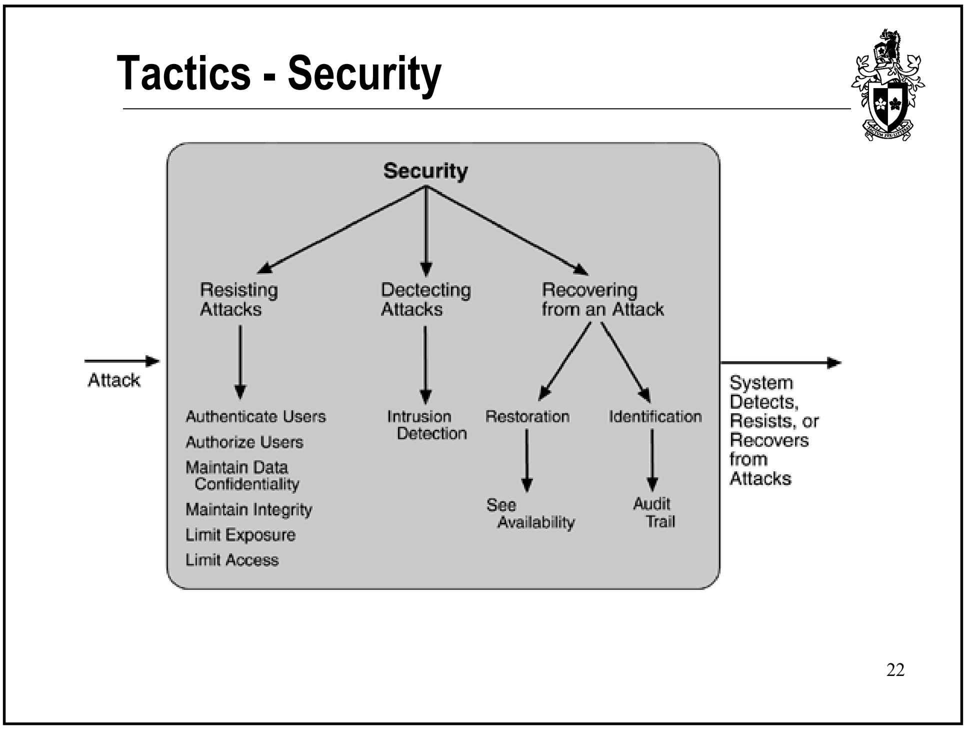
Increase Available Resources

Scheduling Policy

Response Generated Within Time Constraints

21

21



Tactics - Security

Security

Resisting Attacks

Dectecting Attacks

Recovering from an Attack

Attack

Authenticate Users

Authorize Users

Maintain Data Confidentiality

Maintain Integrity

Limit Exposure

Limit Access

Intrusion Detection

Restoration

Identification

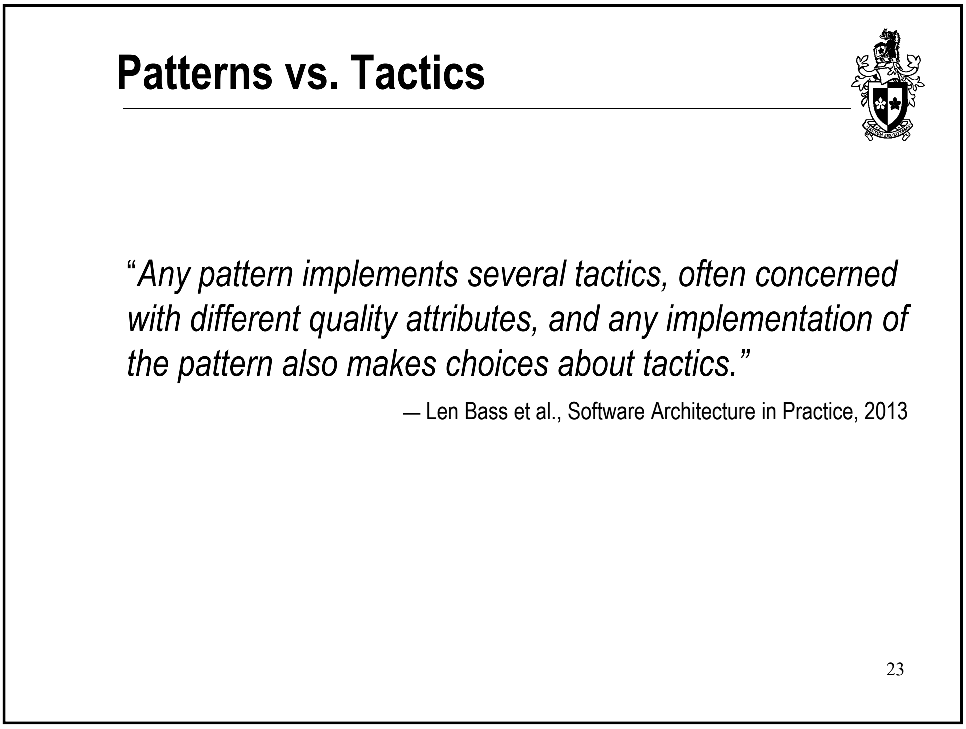
See Availability

Audit Trail

System Detects, Resists, or Recovers from Attacks

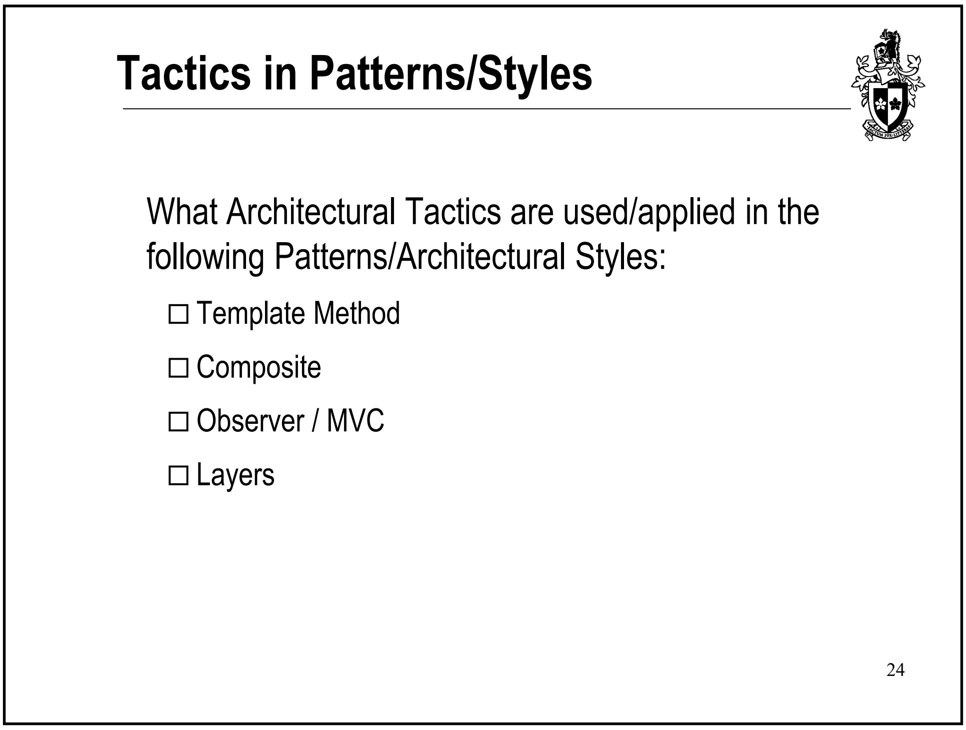
22

22



Patterns vs. Tactics

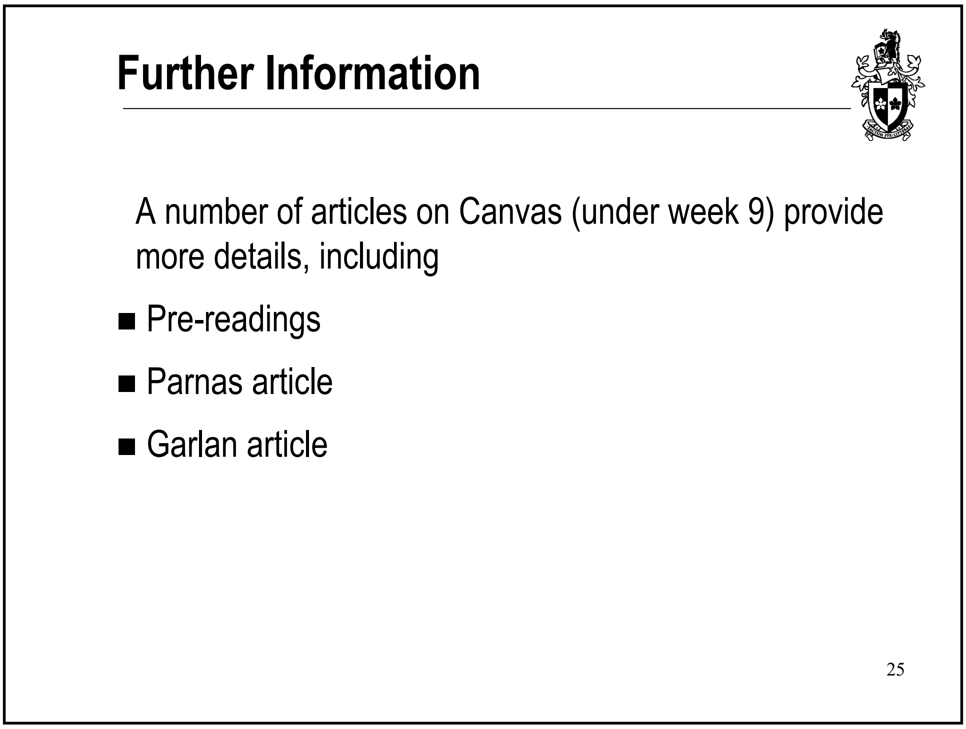
> *“Any pattern implements several tactics, often concerned with different quality attributes, and any implementation of the pattern also makes choices about tactics.”*
>
> — Len Bass et al., Software Architecture in Practice, 2013



Tactics in Patterns/Styles

What Architectural Tactics are used/applied in the following Patterns/Architectural Styles:

- ☐ Template Method
- ☐ Composite
- ☐ Observer / MVC
- ☐ Layers



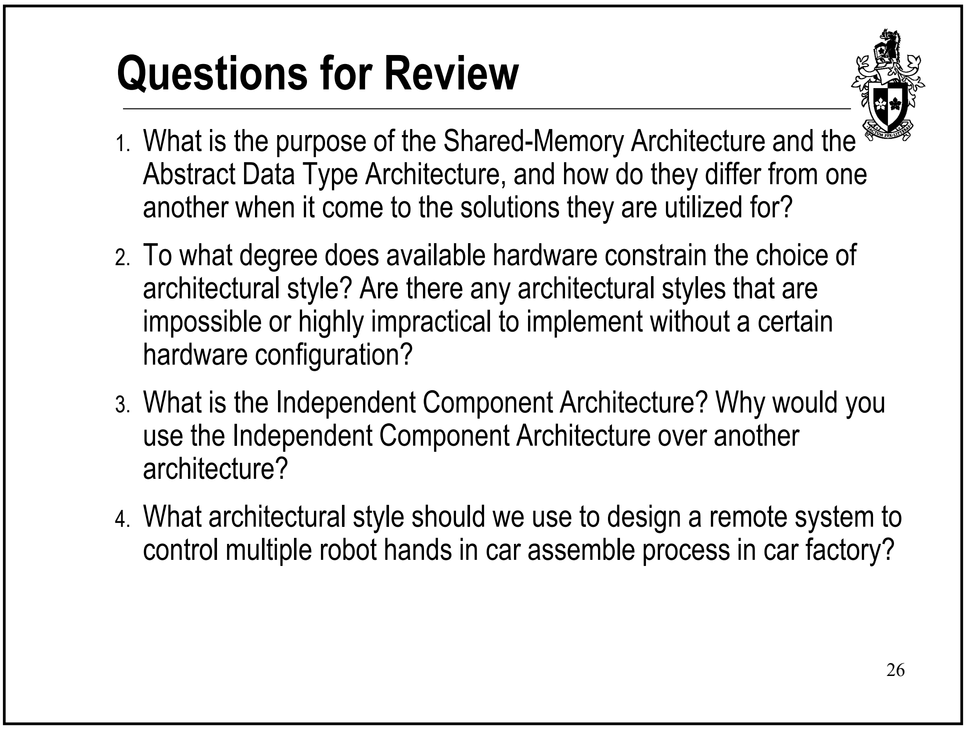
Further Information

A number of articles on Canvas (under week 9) provide more details, including

- Pre-readings
- Parnas article
- Garlan article

25

25

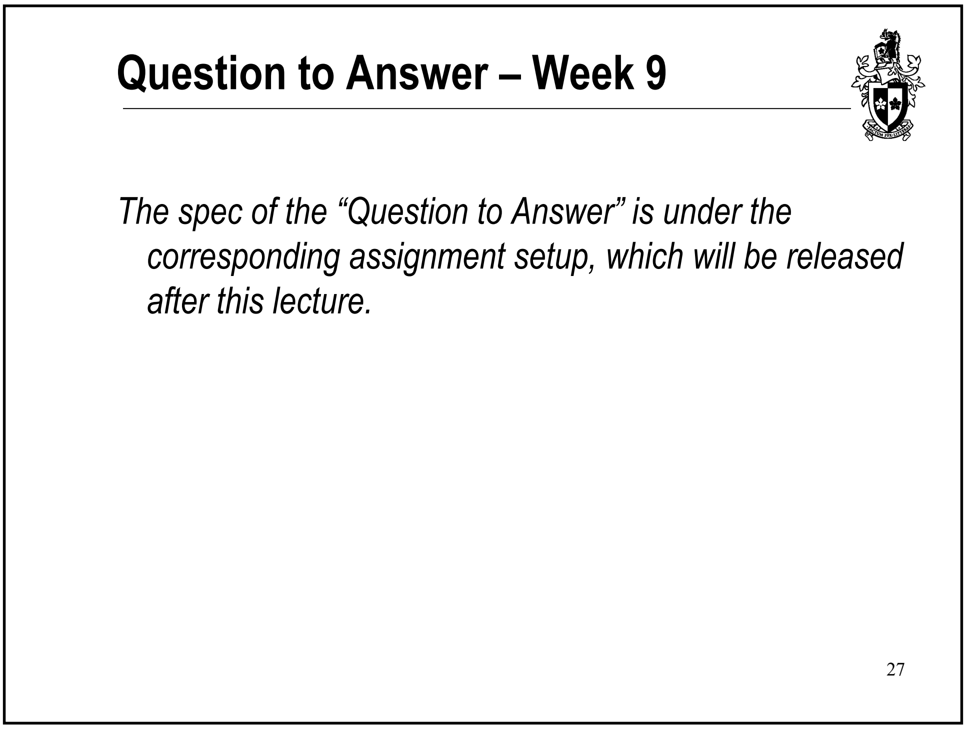


Questions for Review

1. What is the purpose of the Shared-Memory Architecture and the Abstract Data Type Architecture, and how do they differ from one another when it come to the solutions they are utilized for?
2. To what degree does available hardware constrain the choice of architectural style? Are there any architectural styles that are impossible or highly impractical to implement without a certain hardware configuration?
3. What is the Independent Component Architecture? Why would you use the Independent Component Architecture over another architecture?
4. What architectural style should we use to design a remote system to control multiple robot hands in car assemble process in car factory?

26

26

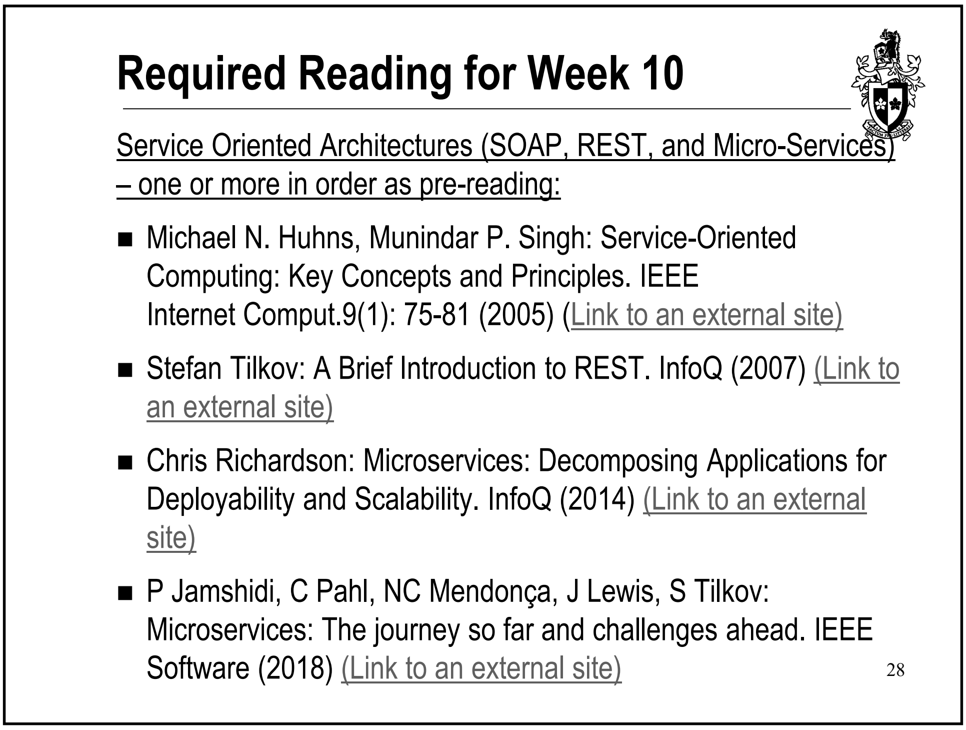


Question to Answer – Week 9

The spec of the “Question to Answer” is under the corresponding assignment setup, which will be released after this lecture.

27

27



Required Reading for Week 10

Service Oriented Architectures (SOAP, REST, and Micro-Services) – one or more in order as pre-reading:

- Michael N. Huhns, Munindar P. Singh: Service-Oriented Computing: Key Concepts and Principles. IEEE Internet Comput.9(1): 75-81 (2005) (Link to an external site)
- Stefan Tilkov: A Brief Introduction to REST. InfoQ (2007) (Link to an external site)
- Chris Richardson: Microservices: Decomposing Applications for Deployability and Scalability. InfoQ (2014) (Link to an external site)
- P Jamshidi, C Pahl, NC Mendonça, J Lewis, S Tilkov: Microservices: The journey so far and challenges ahead. IEEE Software (2018) (Link to an external site)

28

28