# Benefits & Limitations of Patterns & Frameworks: Part 1

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software Integrated Systems

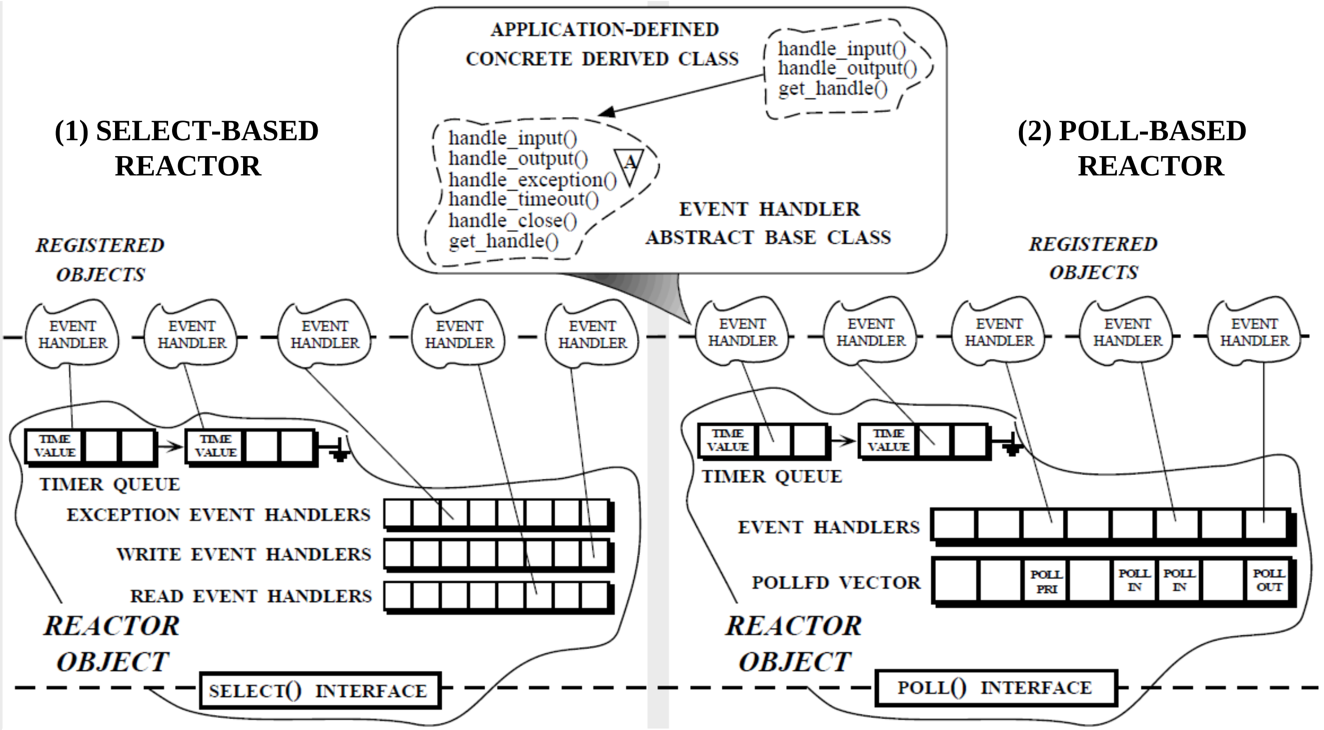
Vanderbilt University
Nashville, Tennessee, USA

# Topics Covered in this Part of the Module

- Summarize the benefits & limitations of patterns

# Benefits of Patterns

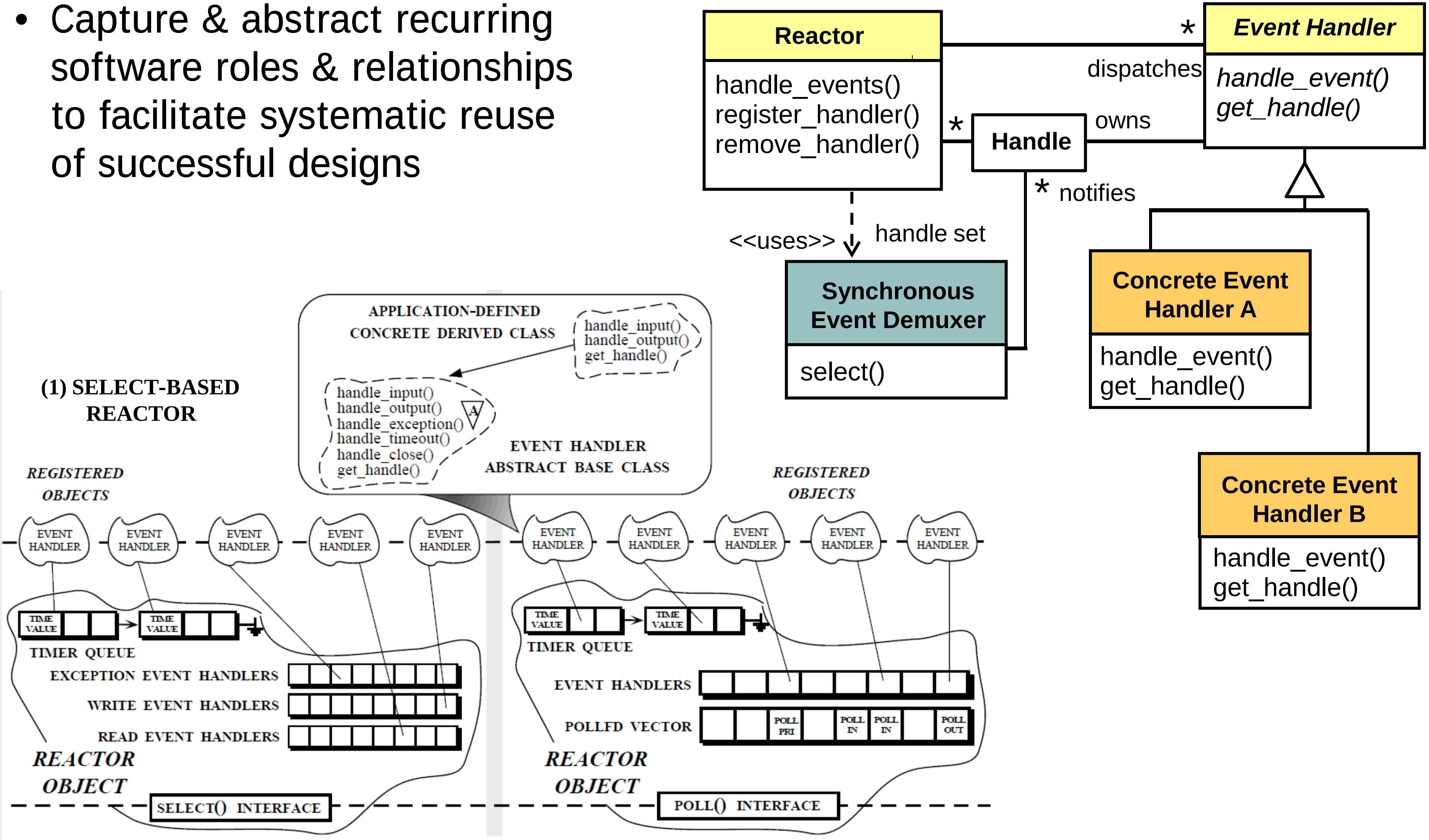
- Capture & abstract recurring software roles & relationships to facilitate systematic reuse of successful designs



See www.dre.vanderbilt.edu/~schmidt/PDF/Reactor2-93.pdf for more info

# Benefits of Patterns

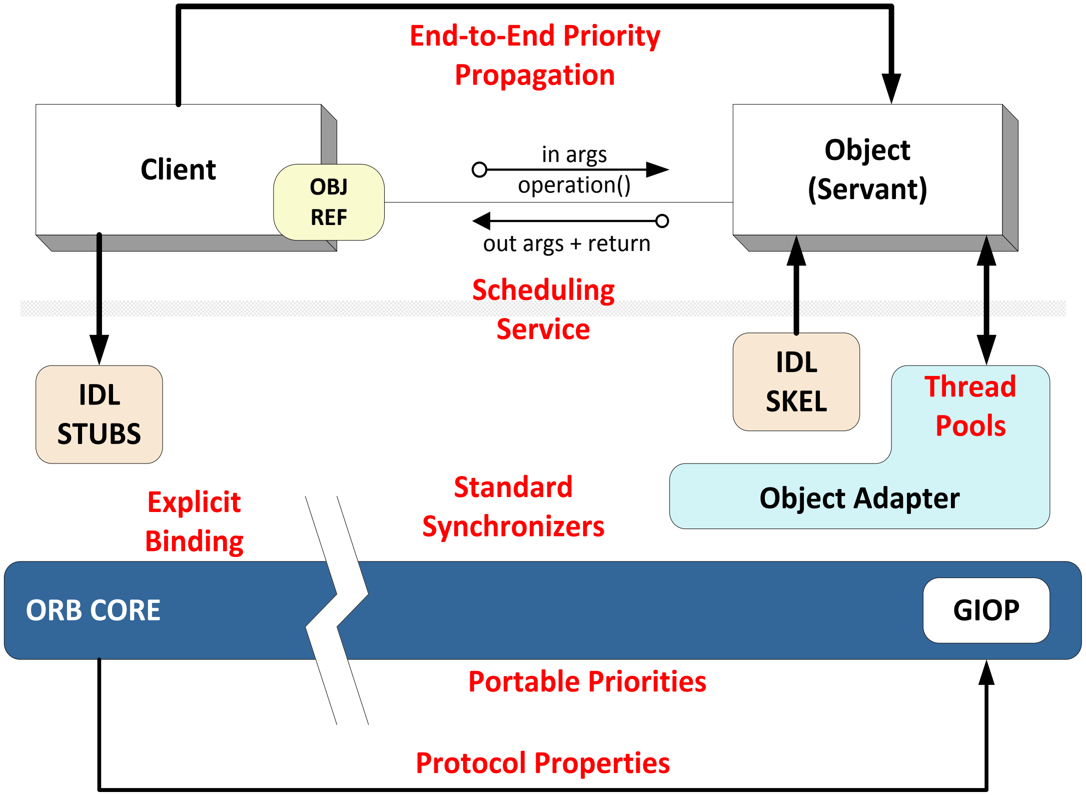
- Capture & abstract recurring software roles & relationships to facilitate systematic reuse of successful designs



See www.dre.vanderbilt.edu/~schmidt/PDF/Reactor.pdf for more info

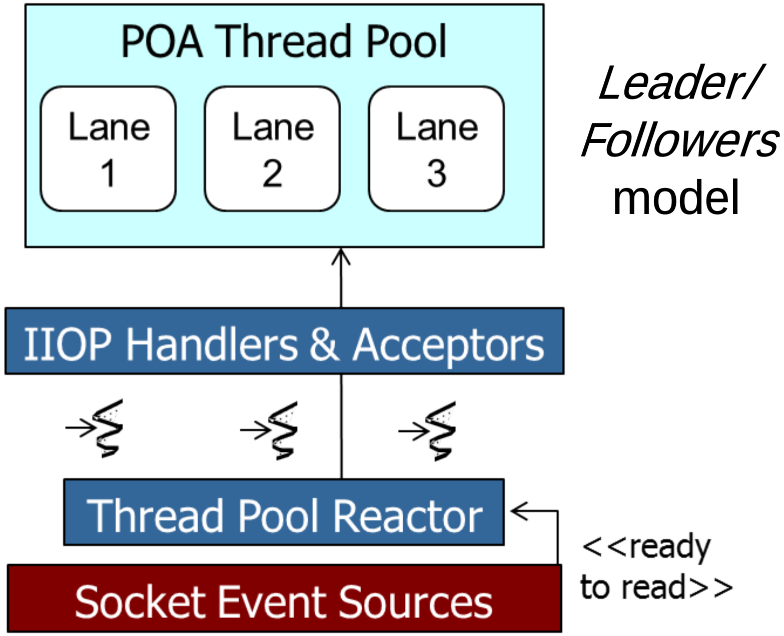
# Benefits of Patterns

- Capture & abstract recurring software roles & relationships to facilitate systematic reuse of successful designs
- Record engineering tradeoffs & design alternatives to enhance development & sustainment

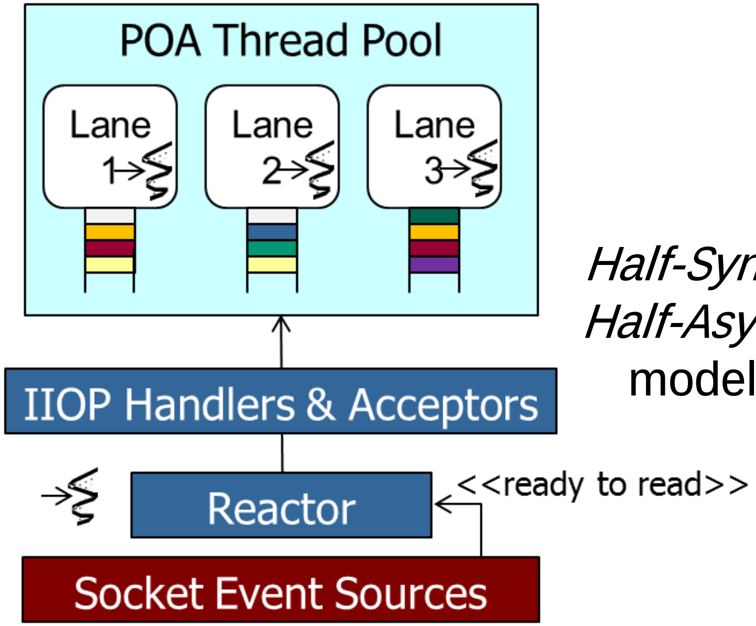


# Benefits of Patterns

- Capture & abstract recurring software roles & relationships to facilitate systematic reuse of successful designs
- Record engineering tradeoffs & design alternatives to enhance development & sustainment



*Leader/ Followers* model



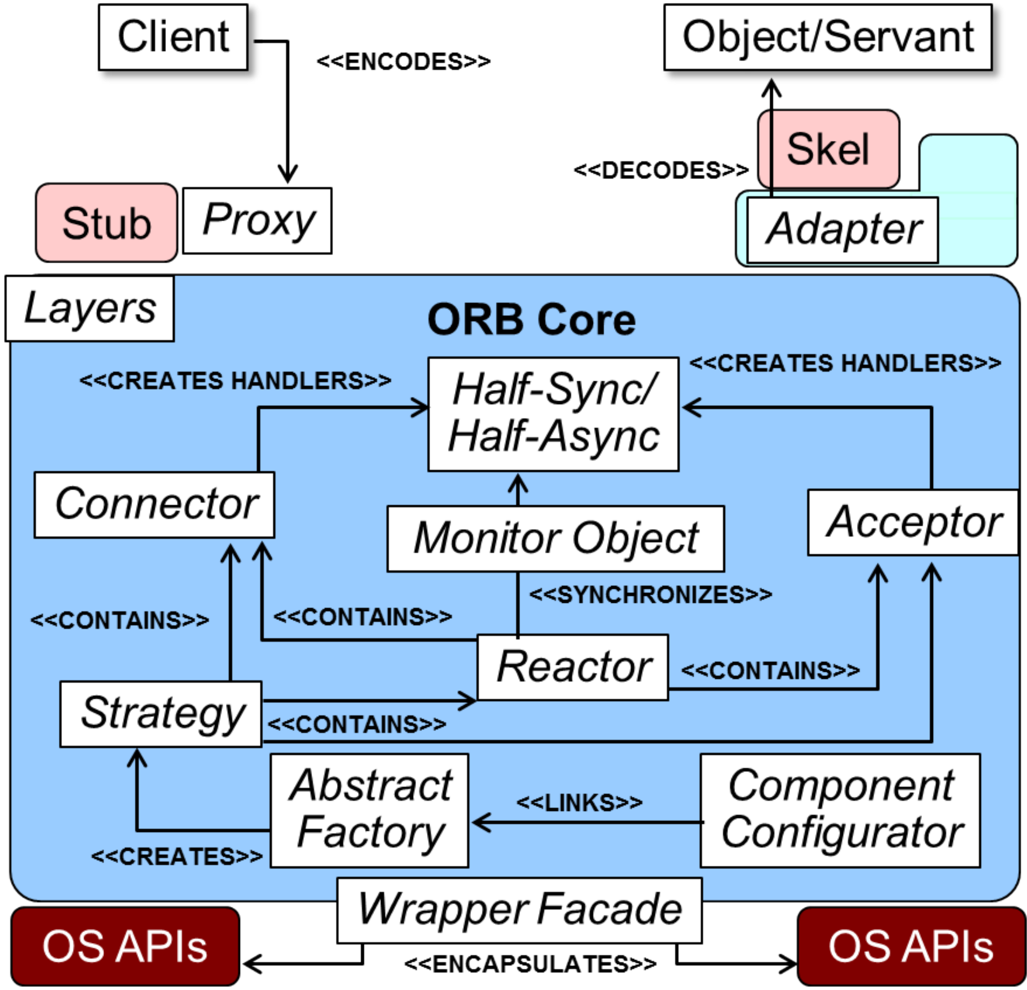
*Half-Sync/ Half-Async* model

| Tradeoff Analysis | Leader/ Followers | Half-Sync/ Half-Async |
|---|---|---|
| Features | Poor | Good |
| Scalability | Poor | Good |
| Efficiency | Good | Poor |
| Optimizations | Good | Poor |
| Priority inversion | Good | Poor |

See www.dre.vanderbilt.edu/~schmidt/PDF/OM-01.pdf for more info

# Benefits of Patterns

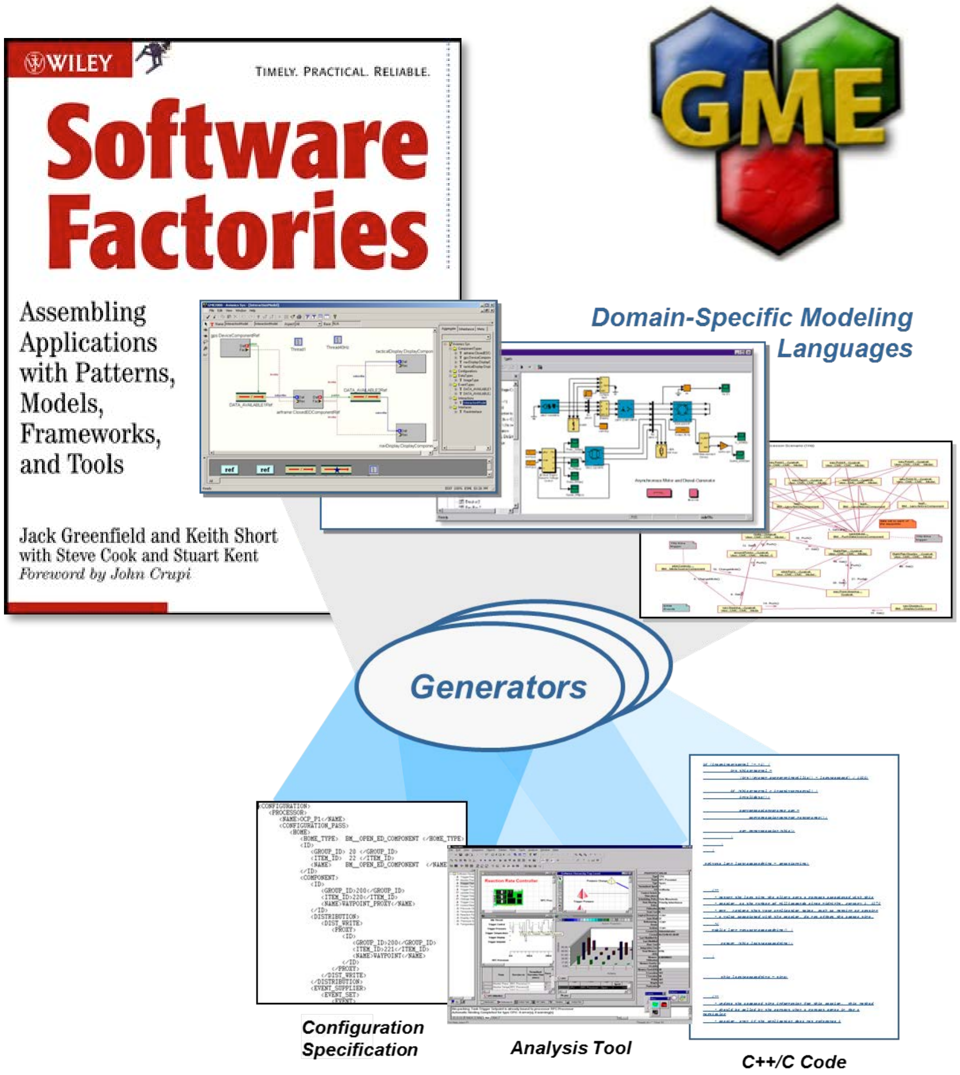
- Capture & abstract recurring software roles & relationships to facilitate systematic reuse of successful designs
- Record engineering tradeoffs & design alternatives to enhance development & sustainment
- Enable a shared design vocabulary that enhances understanding, (re)engineering effort, & team communication



See www.dre.vanderbilt.edu/~schmidt/corba-research-design.html for more

# Benefits of Patterns

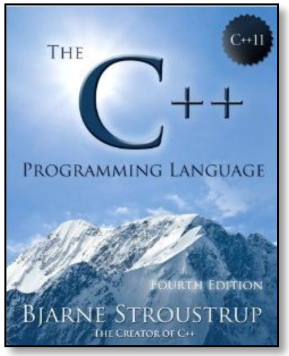
- Capture & abstract recurring software roles & relationships to facilitate systematic reuse of successful designs
- Record engineering tradeoffs & design alternatives to enhance development & sustainment
- Enable a shared design vocabulary that enhances understanding, (re)engineering effort, & team communication
- Provide a basis for automation

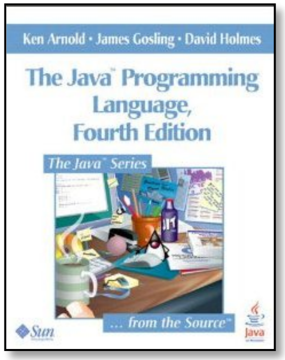

WILEY
Timely. Practical. Reliable.
Software Factories
Assembling Applications with Patterns, Models, Frameworks, and Tools
Jack Greenfield and Keith Short
with Steve Cook and Stuart Kent
Foreword by John Crupi
GME
Domain-Specific Modeling Languages
Generators
Configuration Specification
Analysis Tool
C++/C Code

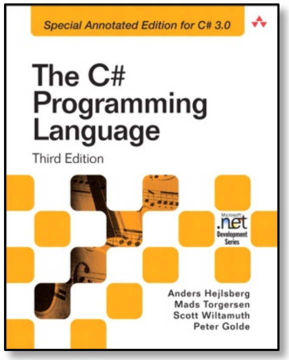
See www.dre.vanderbilt.edu/~schmidt/GEI.pdf for more info

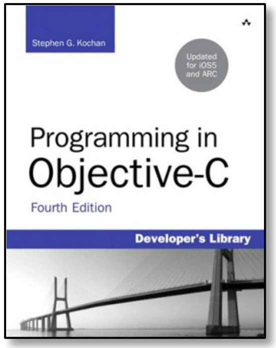
# Benefits of Patterns

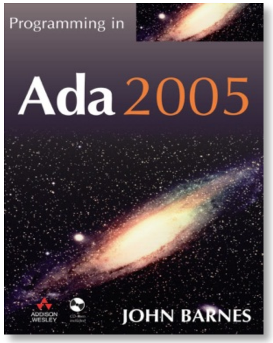
- Capture & abstract recurring software roles & relationships to facilitate systematic reuse of successful designs
- Record engineering tradeoffs & design alternatives to enhance development & sustainment
- Enable a shared design vocabulary that enhances understanding, (re)engineering effort, & team communication
- Provide a basis for automation
- Transcend language-centric biases

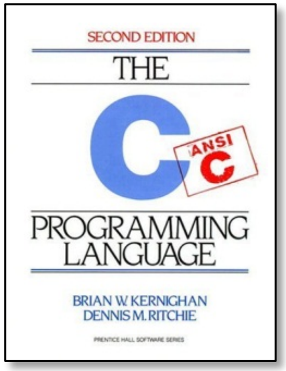



Ken Arnold · James Gosling · David Holmes
The Java Programming Language, Fourth Edition
The Java Series
... from the Source
Sun
Java



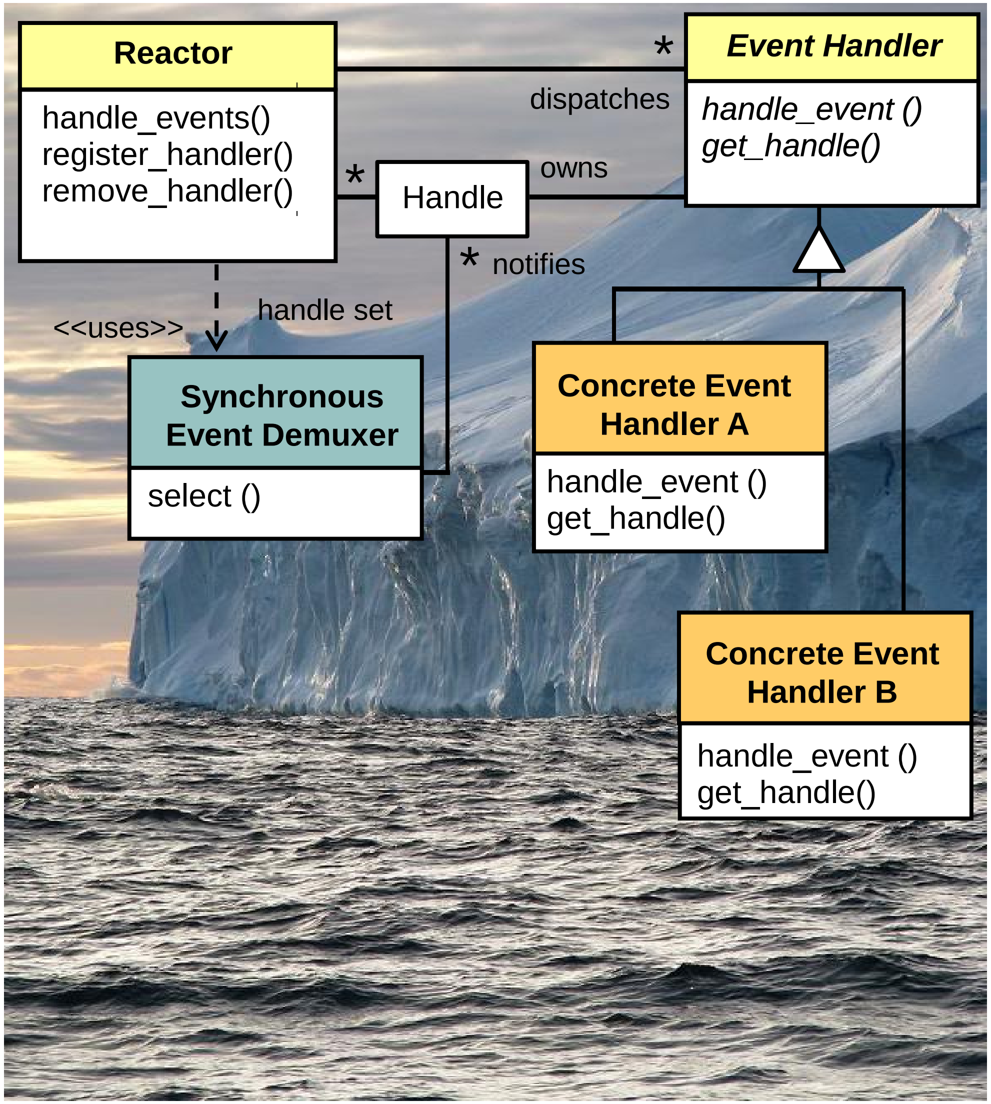






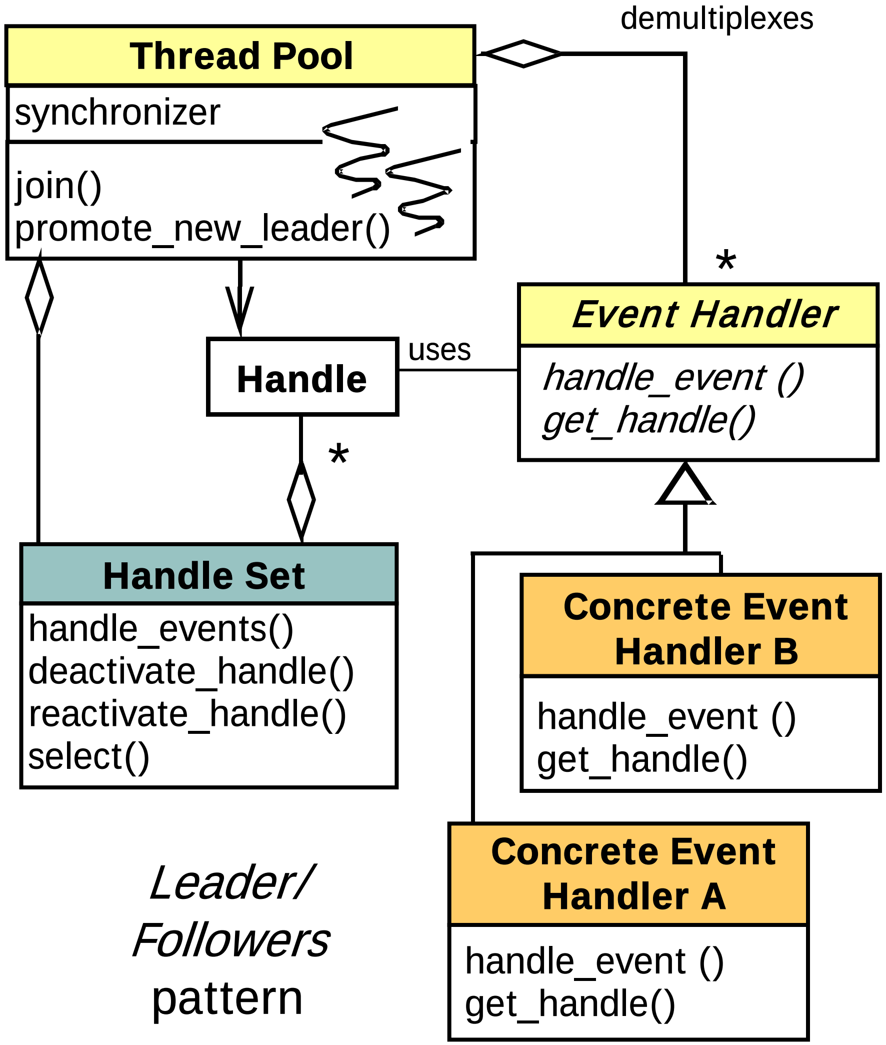
# Benefits of Patterns

- Capture & abstract recurring software roles & relationships to facilitate systematic reuse of successful designs
- Record engineering tradeoffs & design alternatives to enhance development & sustainment
- Enable a shared design vocabulary that enhances understanding, (re)engineering effort, & team communication
- Provide a basis for automation
- Transcend language-centric biases
- Abstract away from non-essential implementation details
  - e.g., during the design phase



# Limitations of Patterns

- Significant tedious & error-prone human effort may be needed to implement patterns manually

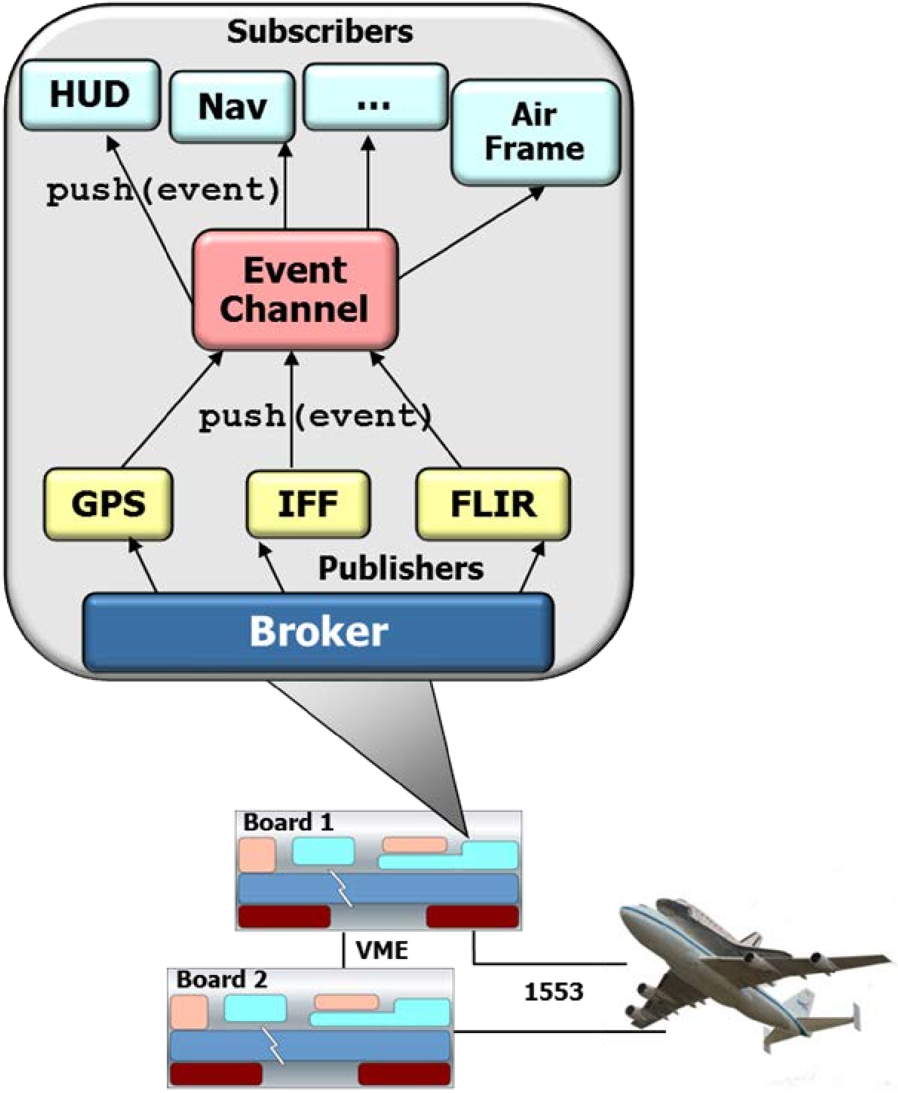


*Leader/ Followers* pattern

See www.dre.vanderbilt.edu/~schmidt/PDF/LF.pdf for more info

# Limitations of Patterns

- Significant tedious & error-prone human effort may be needed to implement patterns manually
- Shared design vocabulary can be deceptively simple



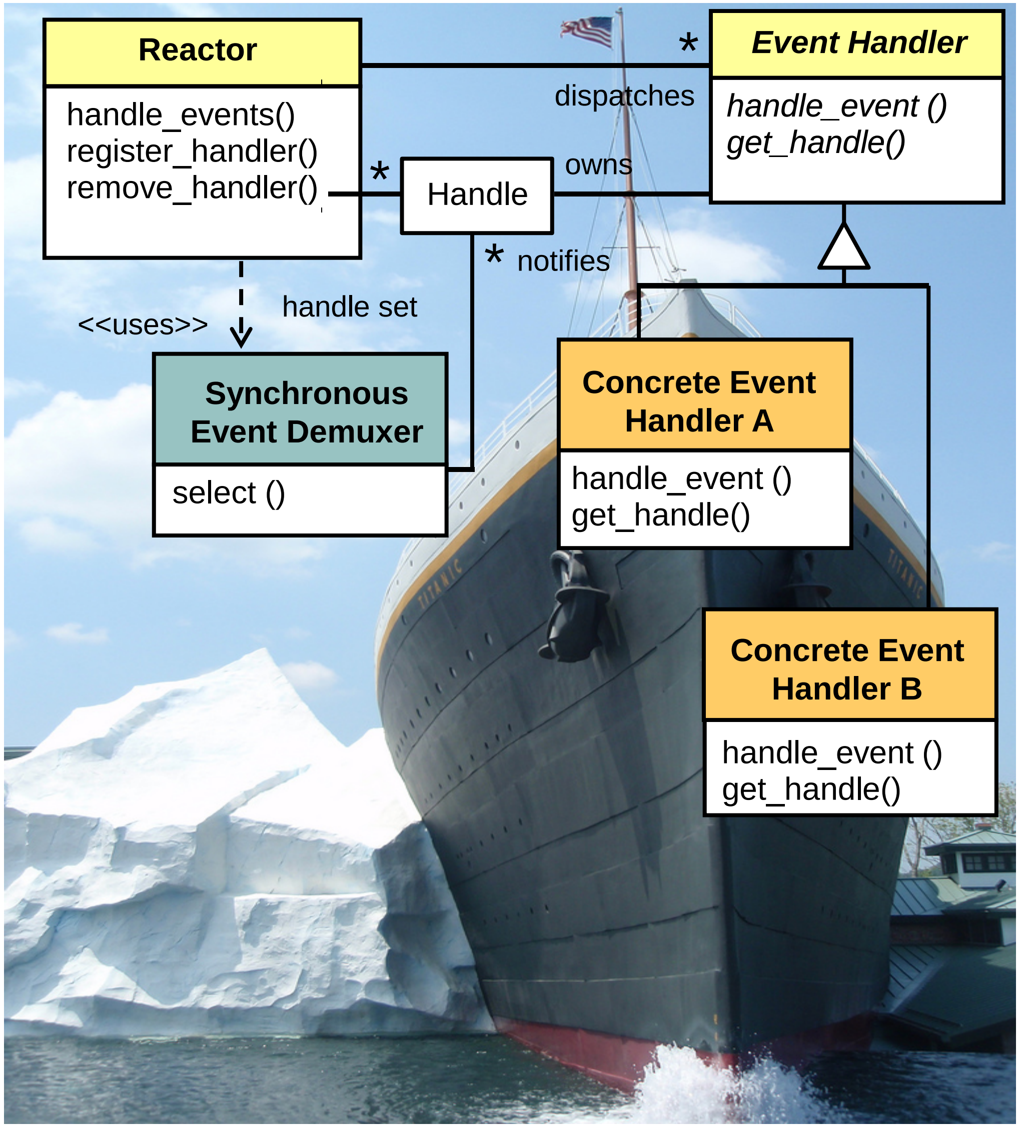
# Limitations of Patterns

- Significant tedious & error-prone human effort may be needed to implement patterns manually
- Shared design vocabulary can be deceptively simple
- Limited knowledge of patterns can constrain design options

Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

# Limitations of Patterns

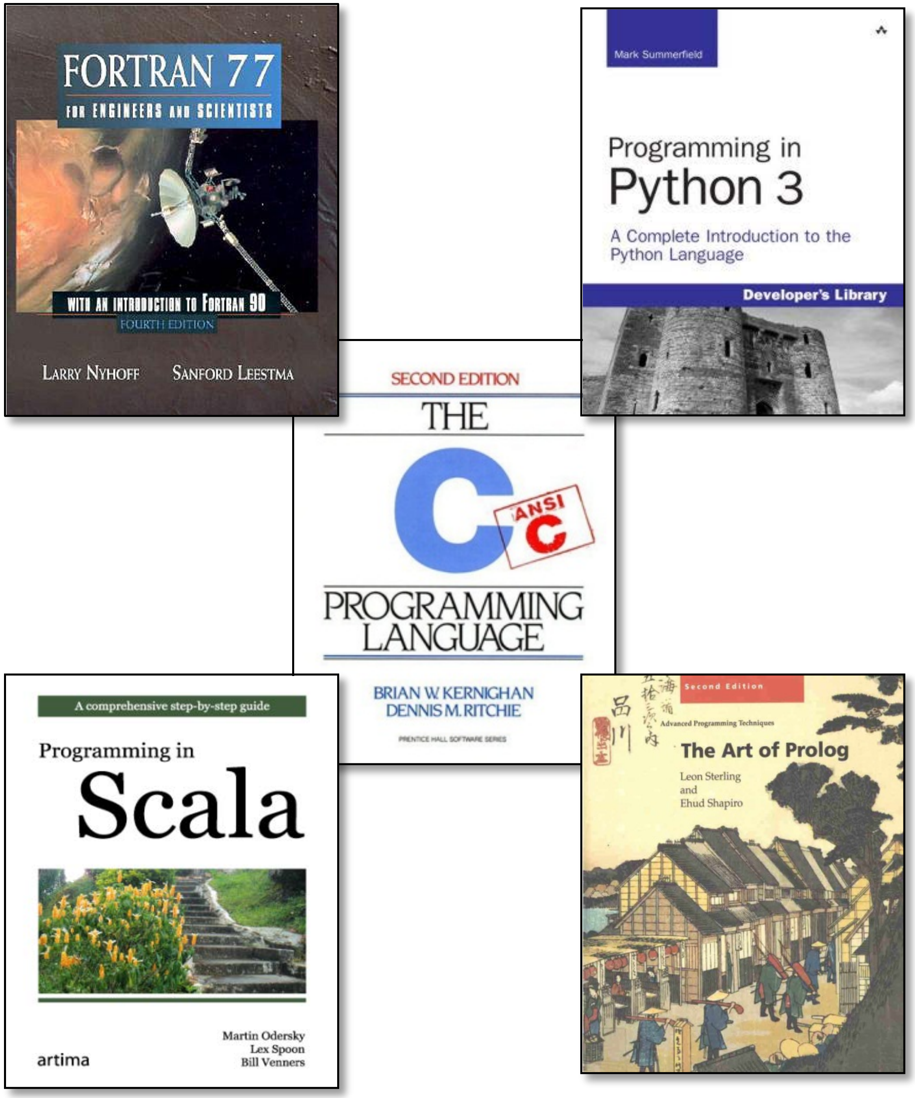
- Significant tedious & error-prone human effort may be needed to implement patterns manually
- Shared design vocabulary can be deceptively simple
- Limited knowledge of patterns can constrain design options
- Essential implementation & optimization details may be neglected
  - e.g., edge-triggered vs. level-triggered event demuxers



See en.wikipedia.org/wiki/Epoll for more info on epoll()

# Limitations of Patterns

- Significant tedious & error-prone human effort may be needed to implement patterns manually
- Shared design vocabulary can be deceptively simple
- Limited knowledge of patterns can constrain design options
- Essential implementation & optimization details may be neglected
- Not all patterns are applicable to non-object-oriented languages
  - Nor are they always applicable in the same way



# Summary

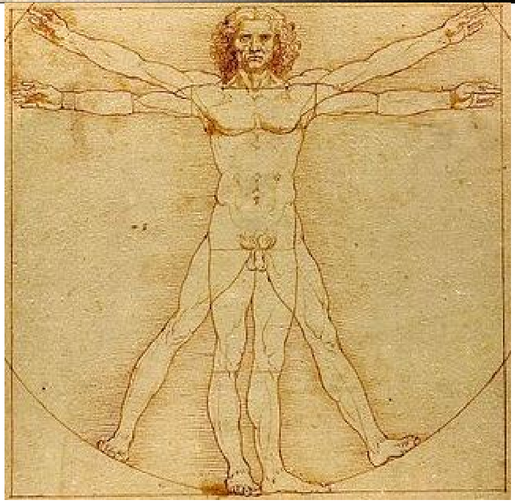
- The goal of patterns is not to replace developer creativity with rote application of rigid design rules & coding laws

# Summary

- The goal of patterns is not to replace developer creativity with rote application of rigid design rules & coding laws
- Nor is the goal to replace humans with automated tools

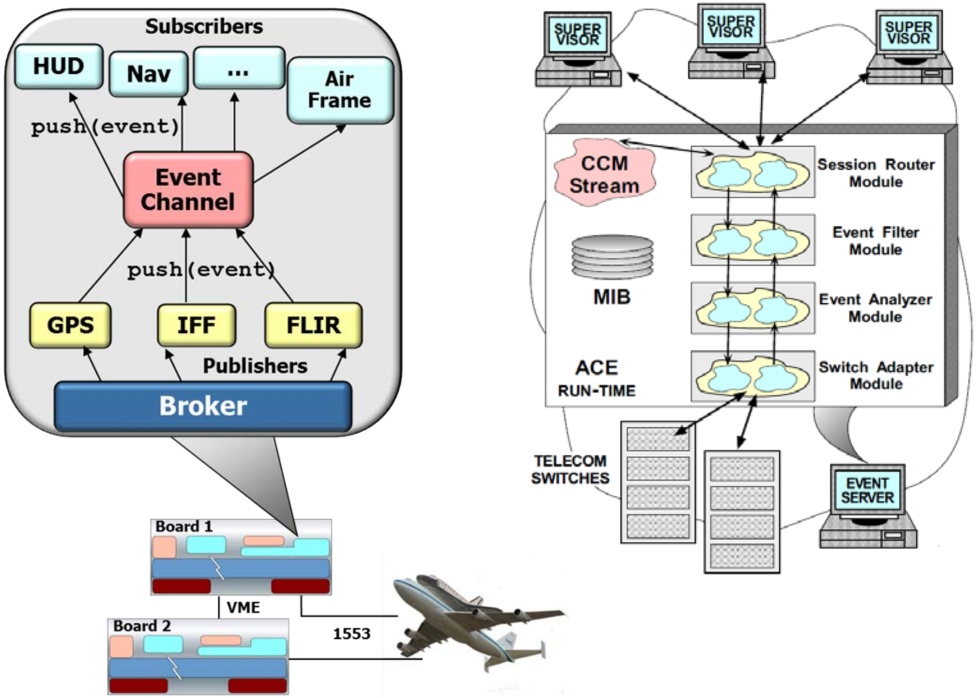
# Summary

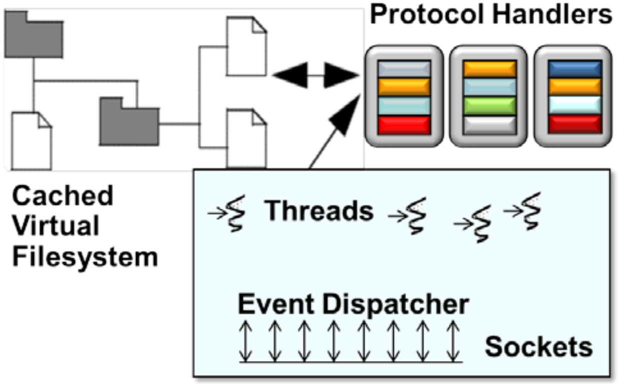
- The goal of patterns is not to replace developer creativity with rote application of rigid design rules & coding laws
- Nor is the goal to replace humans with automated tools
- Instead, the goal is to codify important human insights & experience associated with developing software

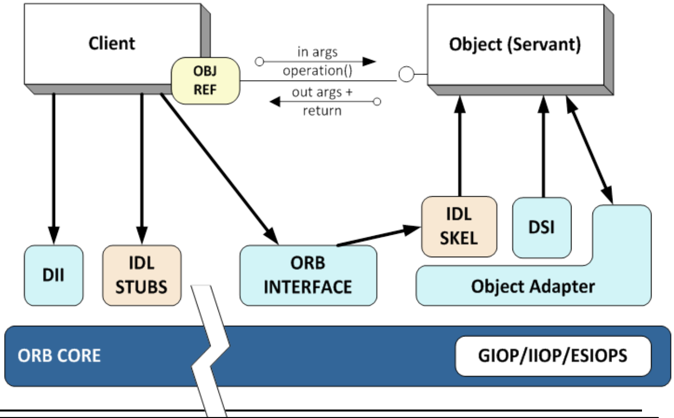


# Summary

- The goal of patterns is not to replace developer creativity with rote application of rigid design rules & coding laws
- Nor is the goal to replace humans with automated tools
- Instead, the goal is to codify important human insights & experience associated with developing software
- Good patterns arise by generalizing from practical experience



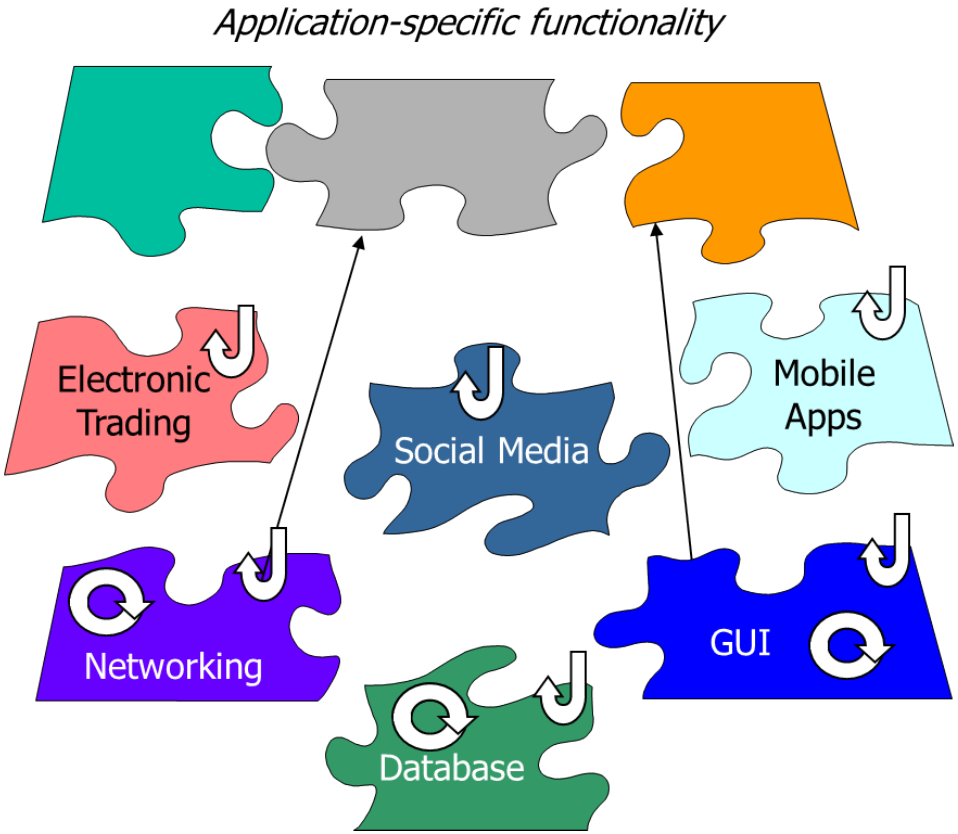




See www.dre.vanderbilt.edu/~schmidt/patterns-experience.html for more info

# Summary

- The goal of patterns is not to replace developer creativity with rote application of rigid design rules & coding laws
- Nor is the goal to replace humans with automated tools
- Instead, the goal is to codify important human insights & experience associated with developing software
- Good patterns arise by generalizing from practical experience
- Addressing key limitations of patterns requires more than just *design* reuse


Application-specific functionality
Electronic Trading
Social Media
Mobile Apps
Networking
GUI
Database