Evaluating Patterns & Frameworks for Concurrent & Networked Software

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

Professor of Computer Science

Institute for Software
Integrated Systems

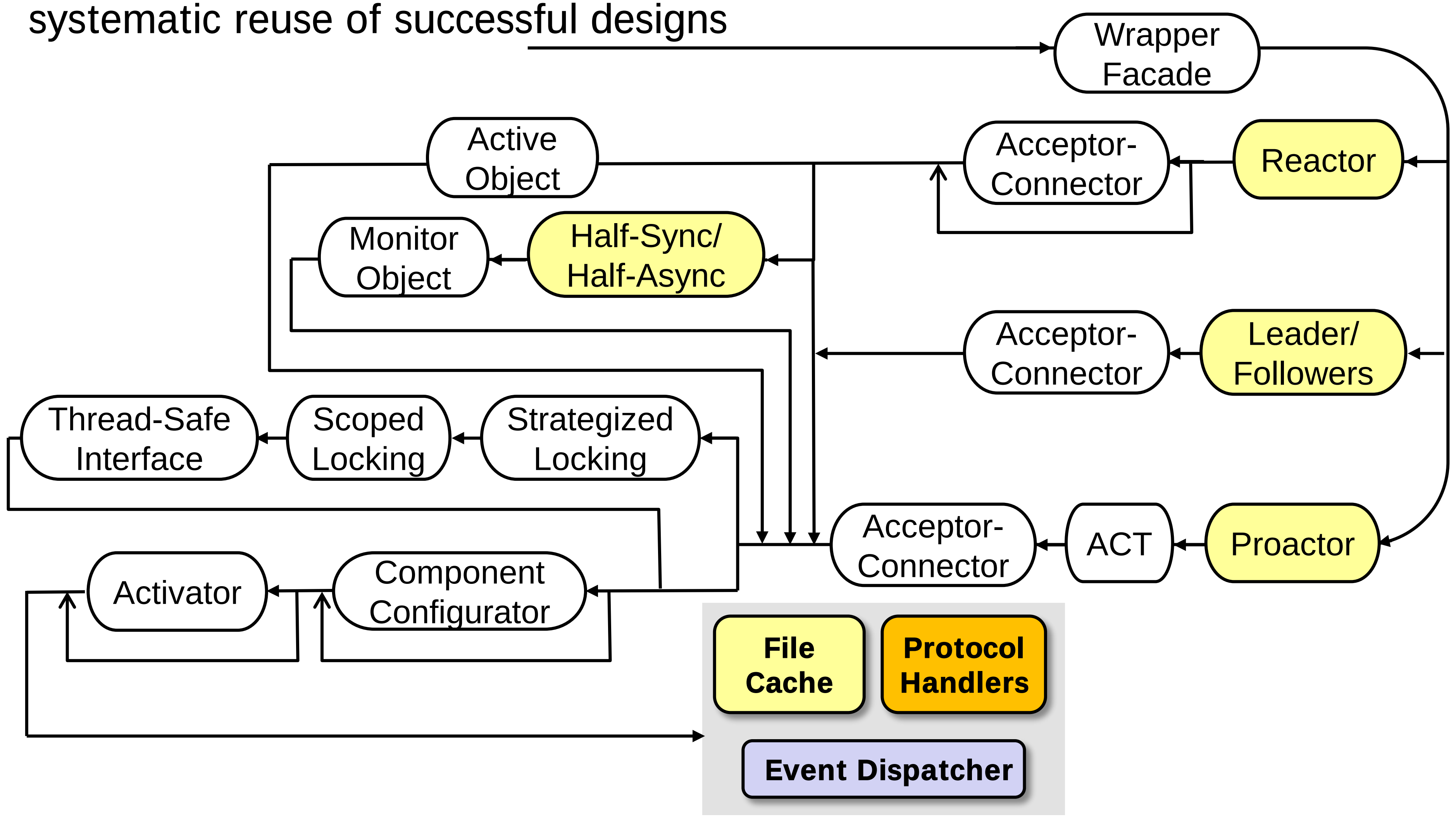
Vanderbilt University
Nashville, Tennessee, USA

Topics Covered in this Module

- Summarize the benefits & limitations of patterns & frameworks for concurrent & networked software

Benefits of Patterns

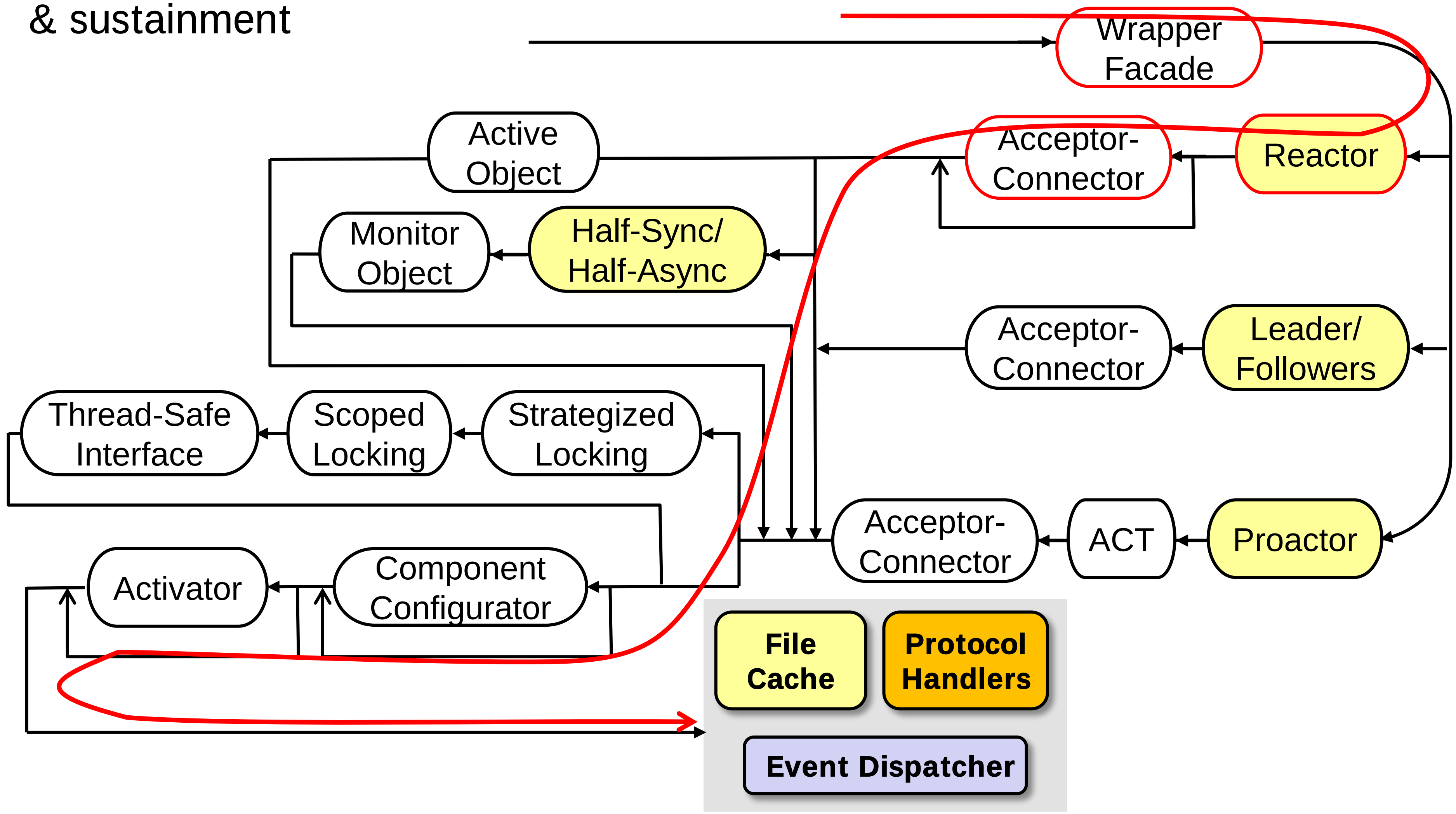
- Capture & abstract recurring software roles & relationships to facilitate systematic reuse of successful designs



No patterns or paths thru the pattern language are specific to web servers

Benefits of Patterns

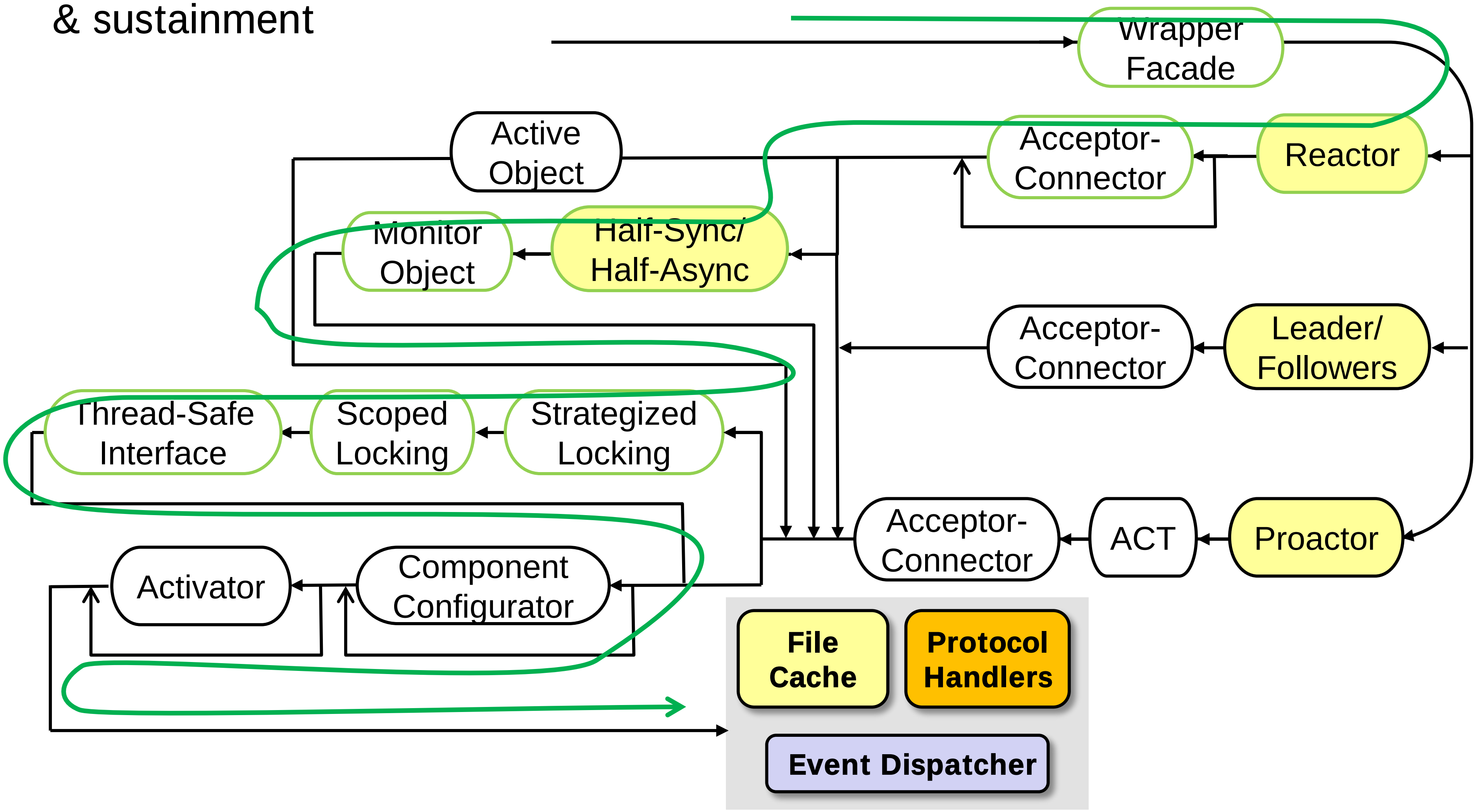
- Record engineering tradeoffs & design alternatives to enhance development & sustainment



Simple, reactive, but non-scalable

Benefits of Patterns

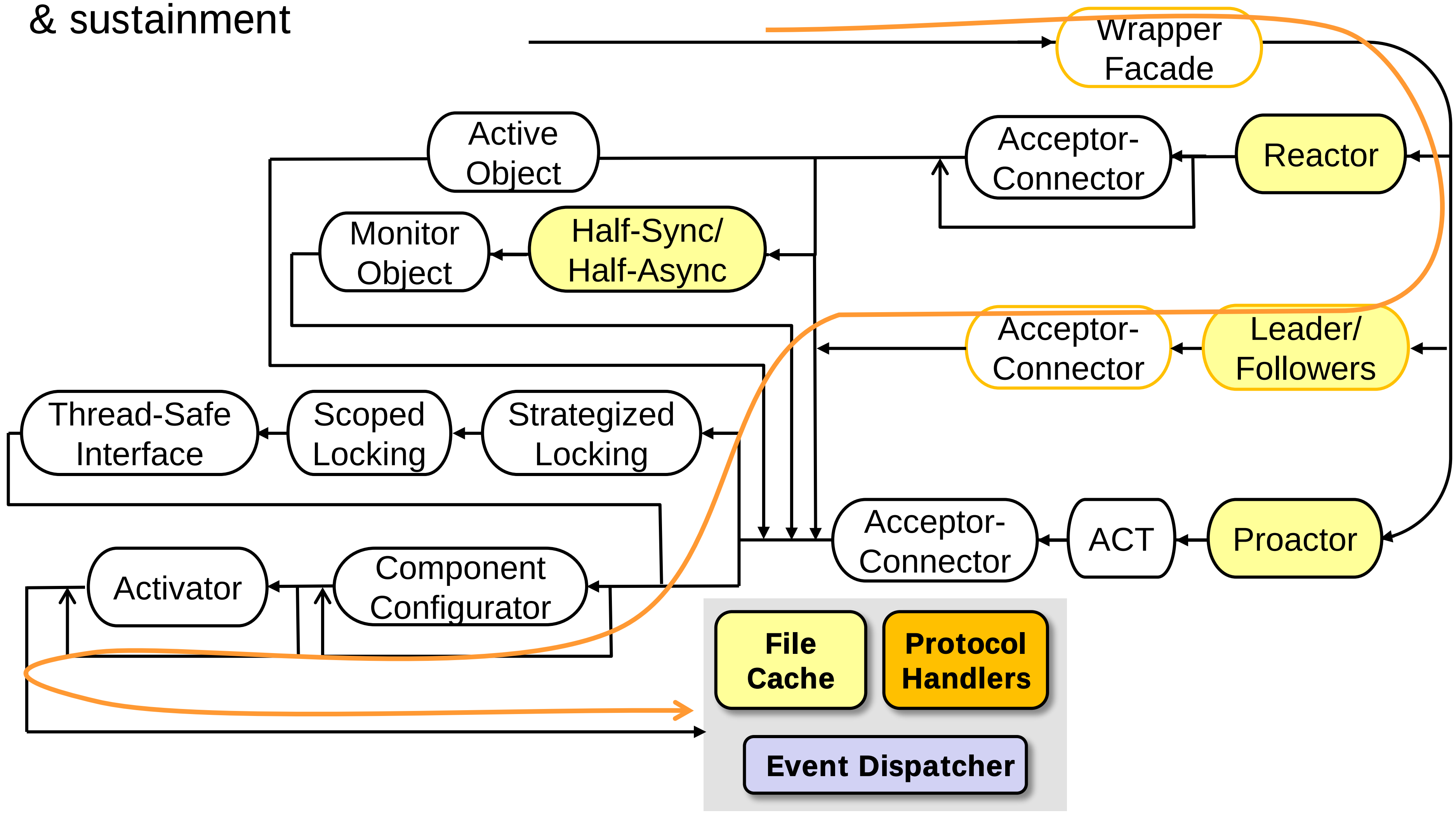
- Record engineering tradeoffs & design alternatives to enhance development & sustainment



Concurrent & scalable, but may be less predictable

Benefits of Patterns

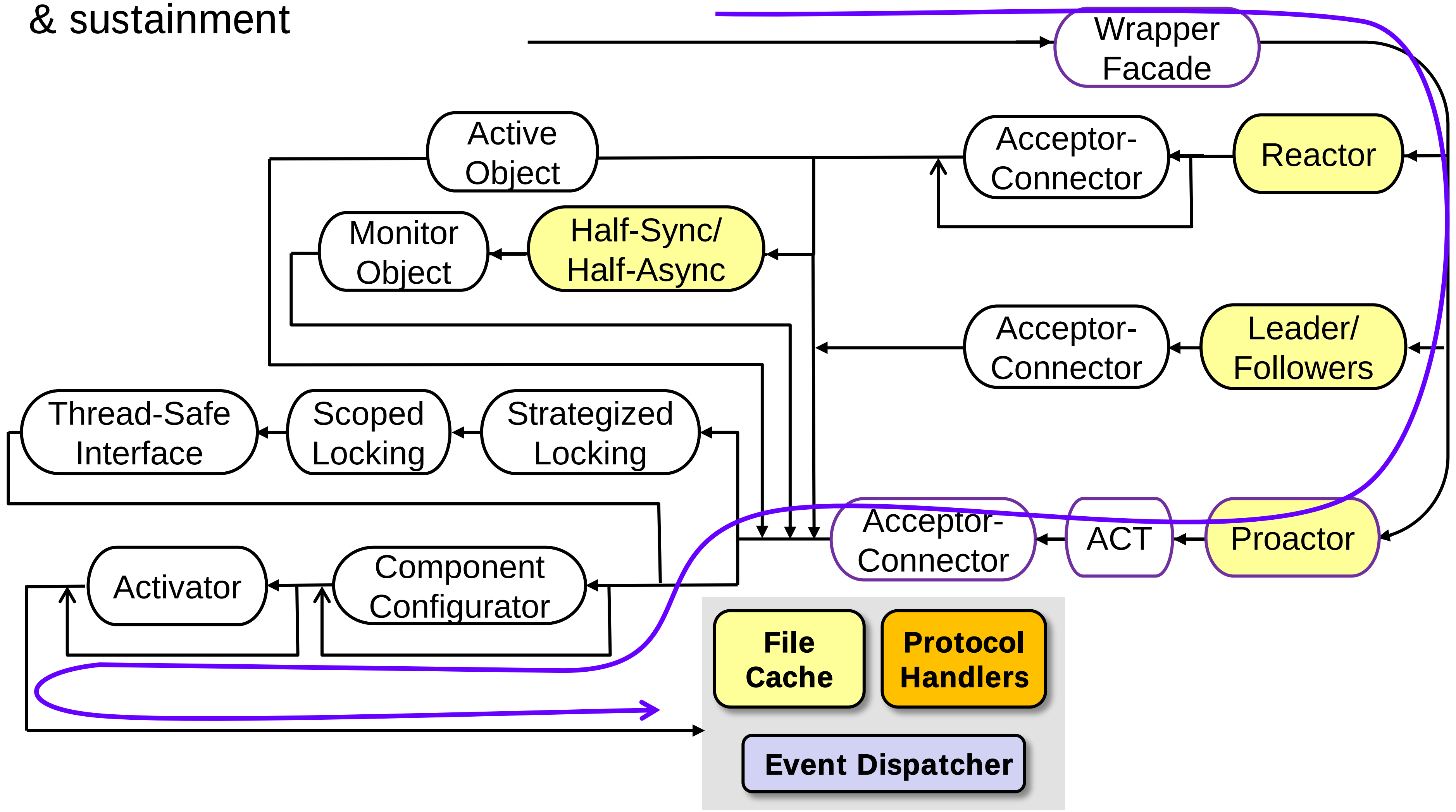
- Record engineering tradeoffs & design alternatives to enhance development & sustainment



Concurrent, predictable, but may be less scalable

Benefits of Patterns

- Record engineering tradeoffs & design alternatives to enhance development & sustainment

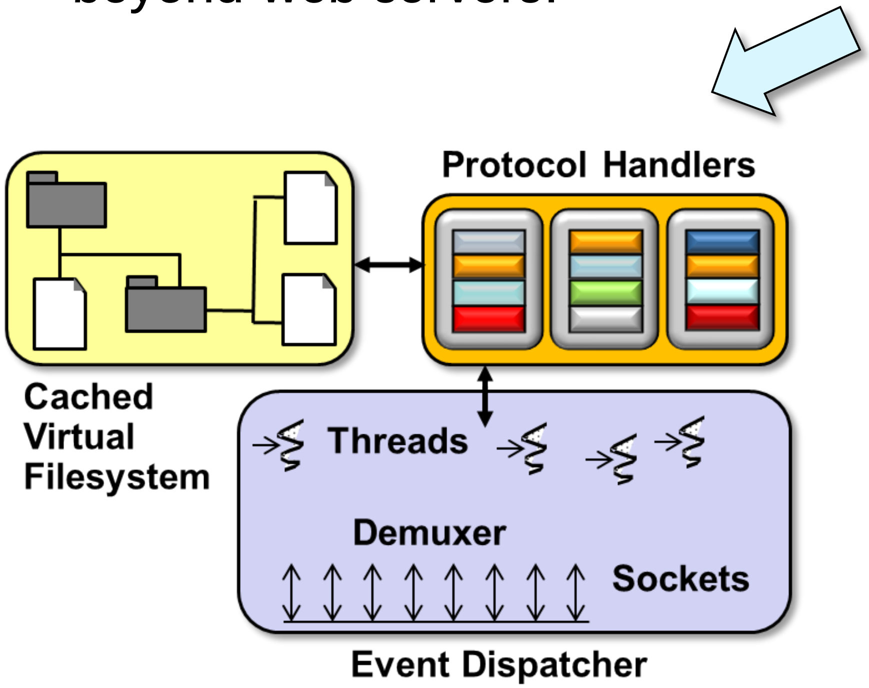


Asynchronous, concurrent, scalable, but limited portability

Benefits of Patterns

- Enable a shared design vocabulary that enhances understanding, (re)engineering effort, & team communication
  - This vocabulary generalizes far beyond web servers!

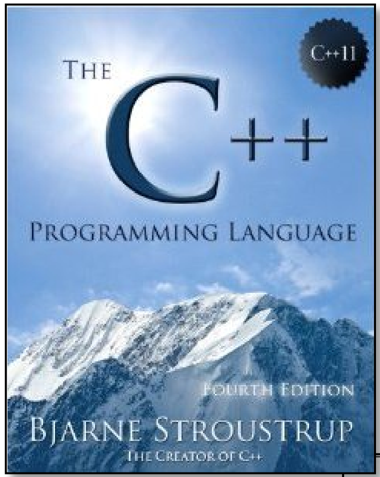
Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

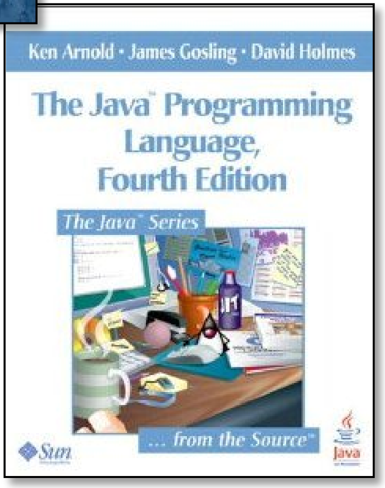


See www.dre.vanderbilt.edu/JAWS for more info

Benefits of Patterns

- Transcend language-centric biases

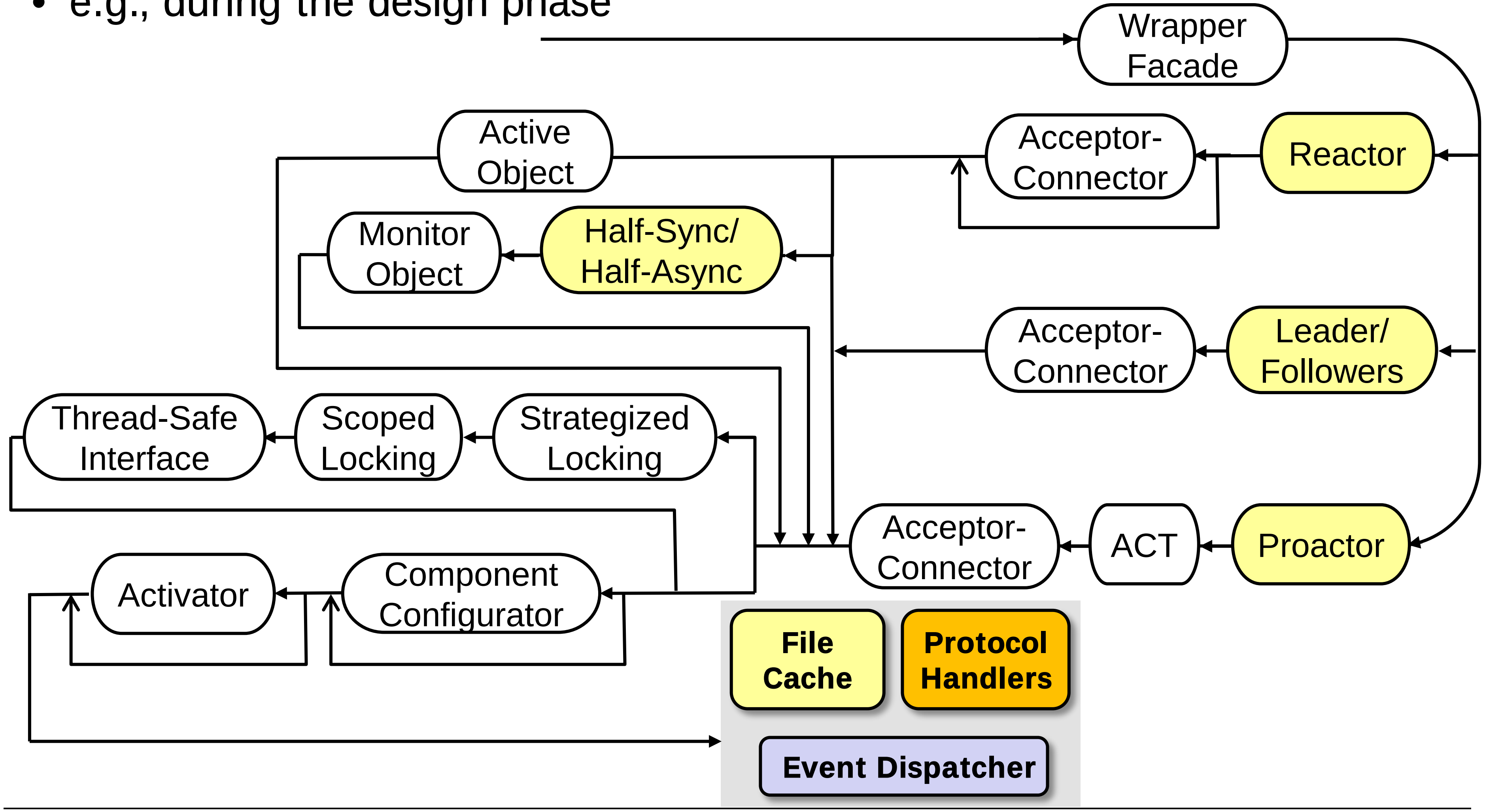



Ken Arnold · James Gosling · David Holmes
The Java™ Programming Language, Fourth Edition
The Java™ Series
... from the Source™
Sun
Java

Design Problem	Pattern(s)
Encapsulating low-level OS APIs	Wrapper Facade
Decoupling event demuxing & connection management from protocol processing	Reactor & Acceptor-Connector
Scaling up performance via multi-threading	Half-Sync/Half-Async & Active Object
Synchronized request queue	Monitor Object
Minimizing multi-threading overhead	Leader/Followers
Using asynchronous I/O effectively	Proactor
Efficiently demuxing asynchronous operations & completions	Asynchronous Completion Token
Enhancing server (re)configurability	Component Configurator
Minimizing unused server resources	Activator
Transparently parameterizing synchronization into components	Strategized Locking
Ensuring locks are released properly	Scoped Locking
Minimizing unnecessary locking	Thread-Safe Interface

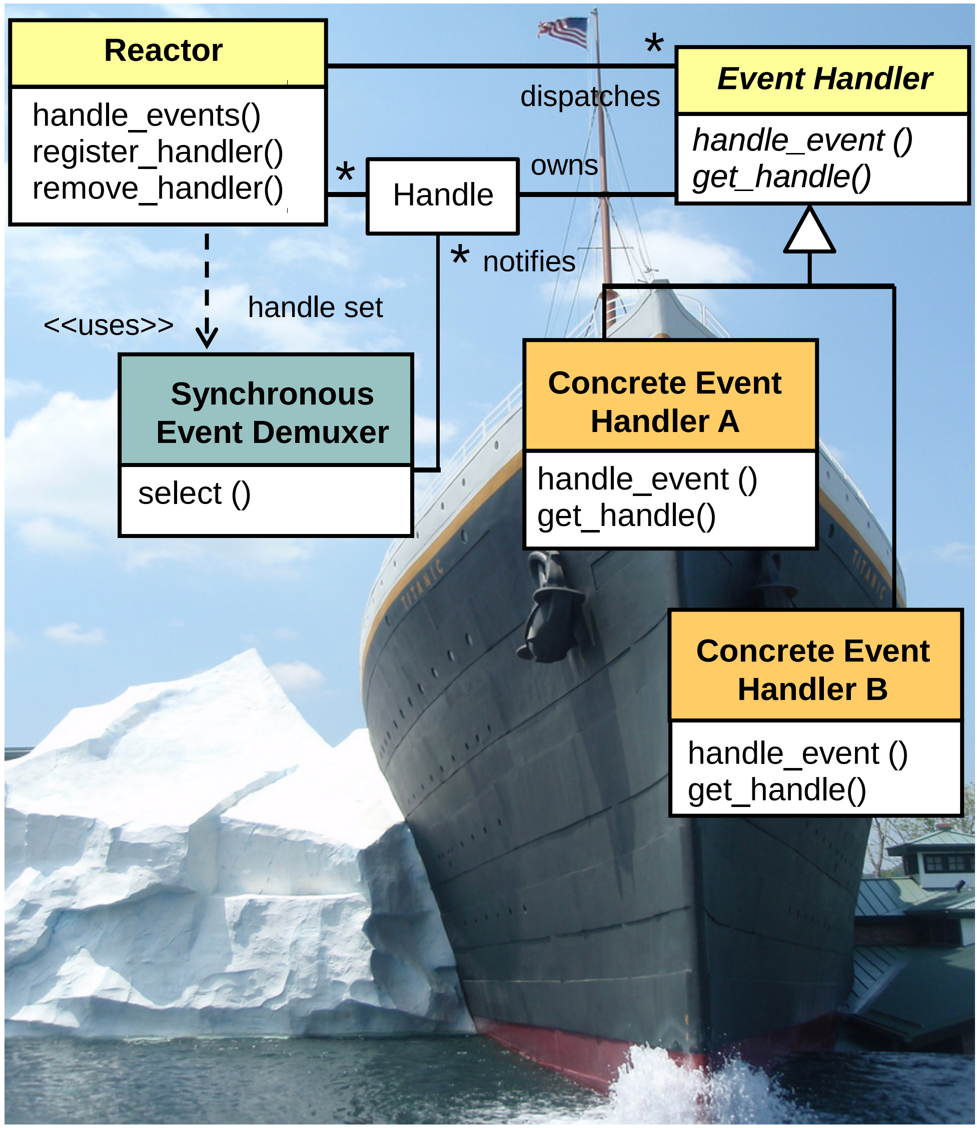
Benefits of Patterns

- Abstract away from non-essential implementation details
  - e.g., during the design phase



Limitations of Patterns

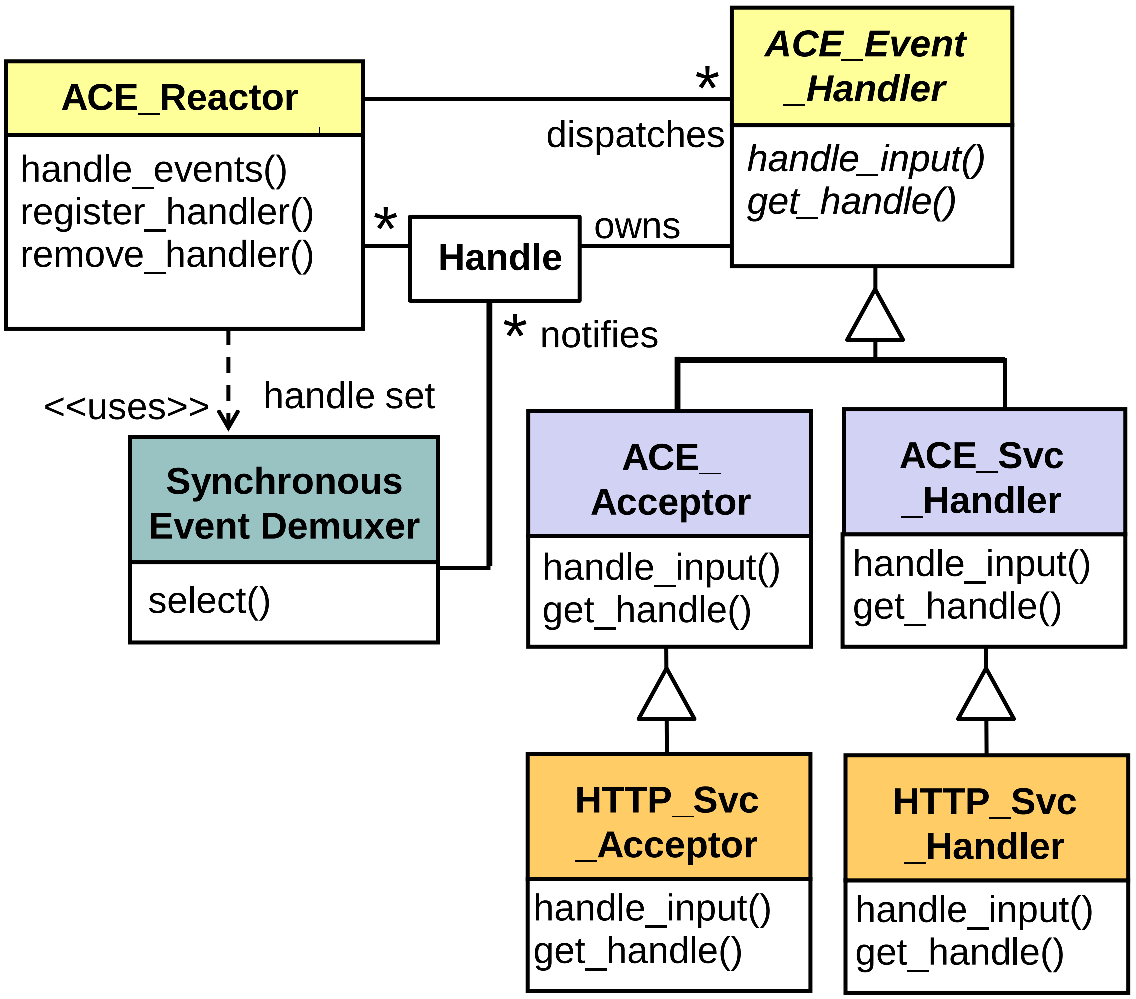
- May neglect essential details of implementations & optimization, e.g.
  - Edge-triggered vs. level-triggered event demuxers
  - **`WaitForMultipleObjects()`** vs. **`select()`**
  - Threading & synchronization semantics
  - Asynchronous I/O semantics on Windows vs. POSIX
  - Local & remote inter-process communication (IPC) mechanisms
  - Language-specific features



Some limitations from before don't apply when combining patterns & frameworks

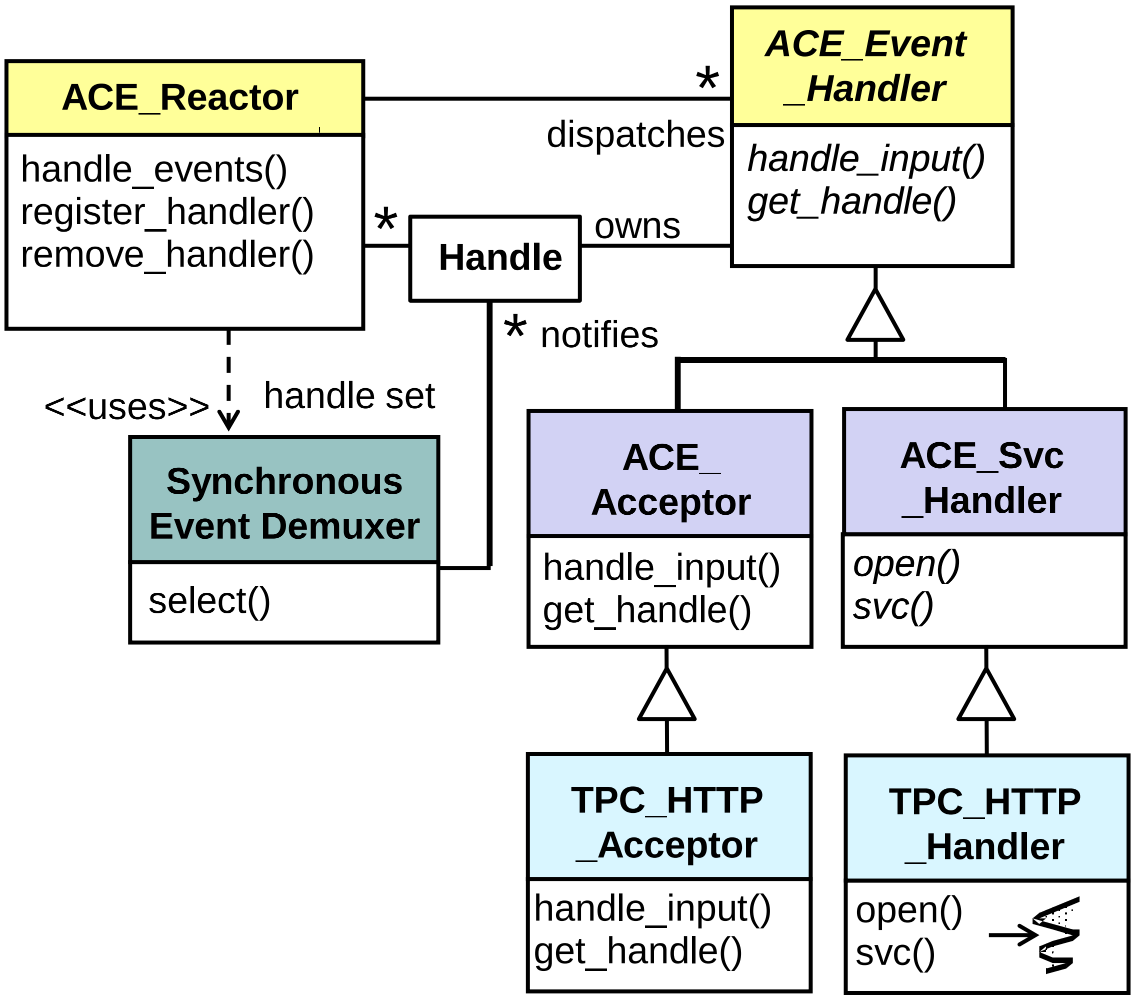
Benefits of Frameworks

- **Design reuse**
  - e.g., by guiding app developers thru steps needed to ensure successful creation & deployment of software



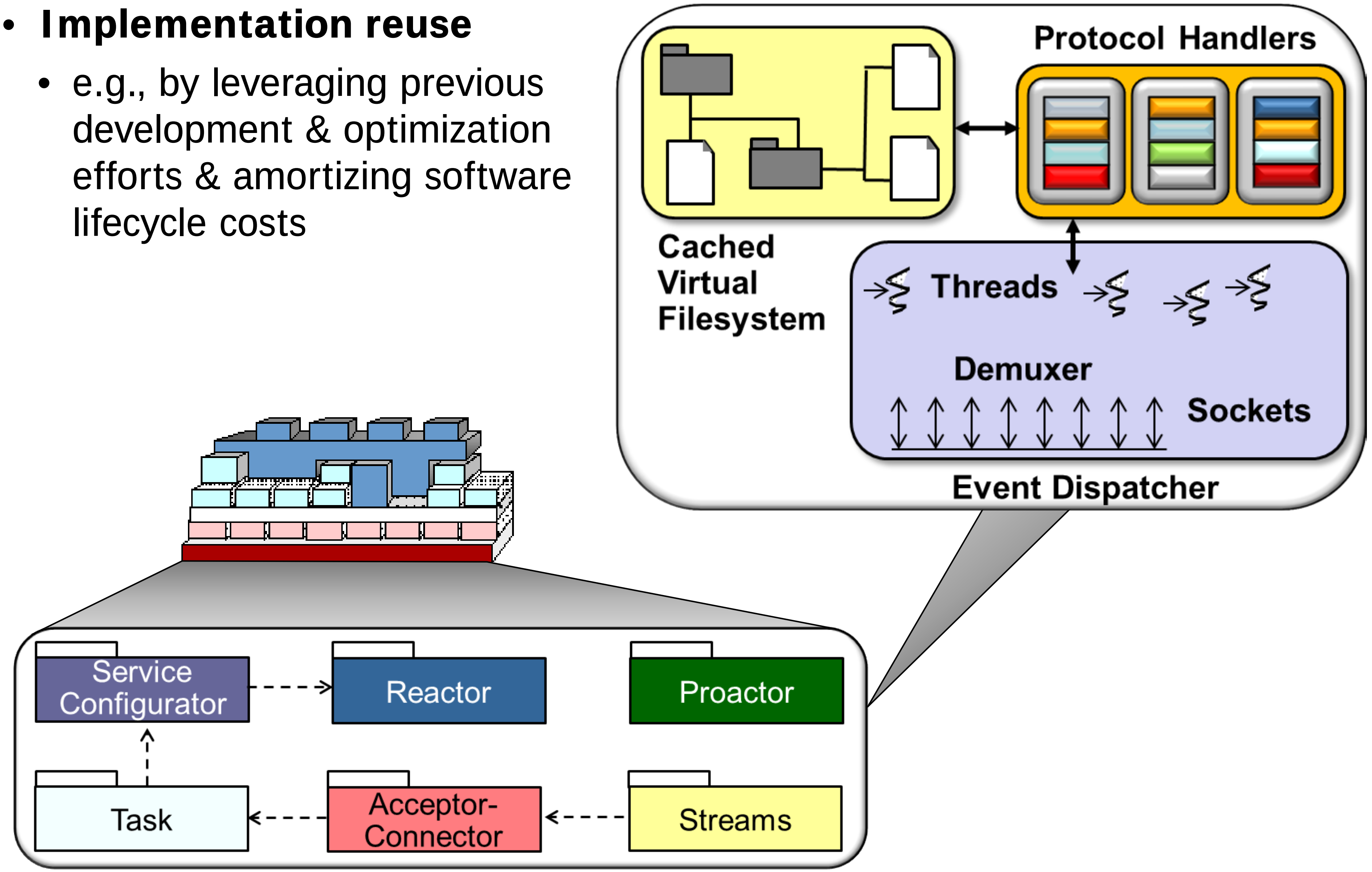
Benefits of Frameworks

- **Design reuse**
  - e.g., by guiding app developers thru steps needed to ensure successful creation & deployment of software



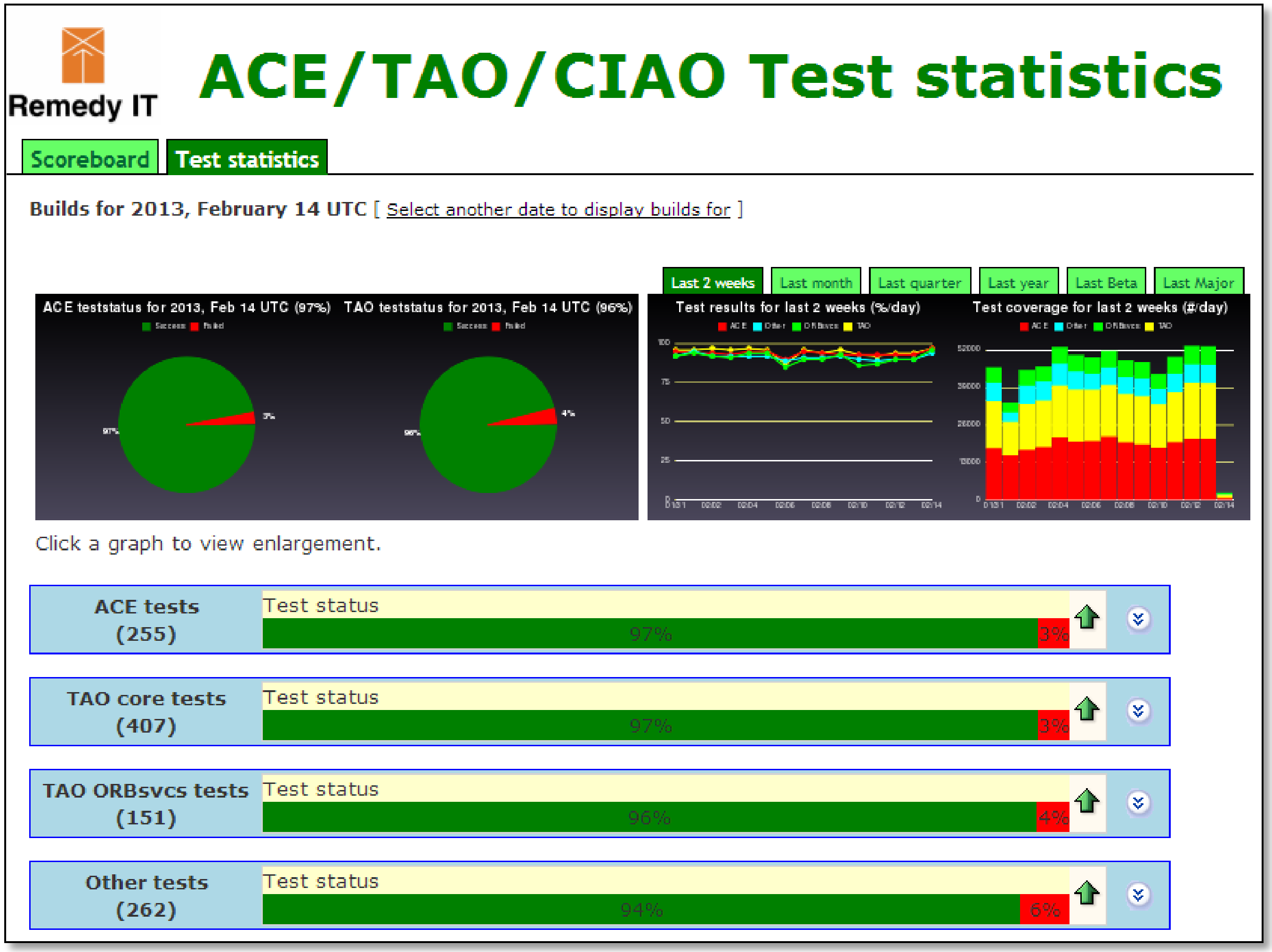
Benefits of Frameworks

- **Implementation reuse**
  - e.g., by leveraging previous development & optimization efforts & amortizing software lifecycle costs



Benefits of Frameworks

- **Validation reuse**
  - e.g., by amortizing the efforts of validating application- & platform-independent portions of software, thereby enhancing dependability & performance


Remedy IT

ACE/TAO/CIAO Test statistics

Scoreboard | Test statistics

Builds for 2013, February 14 UTC [Select another date to display builds for]

Last 2 weeks
Last month
Last quarter
Last year
Last Beta
Last Major
ACE teststatus for 2013, Feb 14 UTC (97%)
TAO teststatus for 2013, Feb 14 UTC (96%)
Test results for last 2 weeks (%/day)
Test coverage for last 2 weeks (#/day)

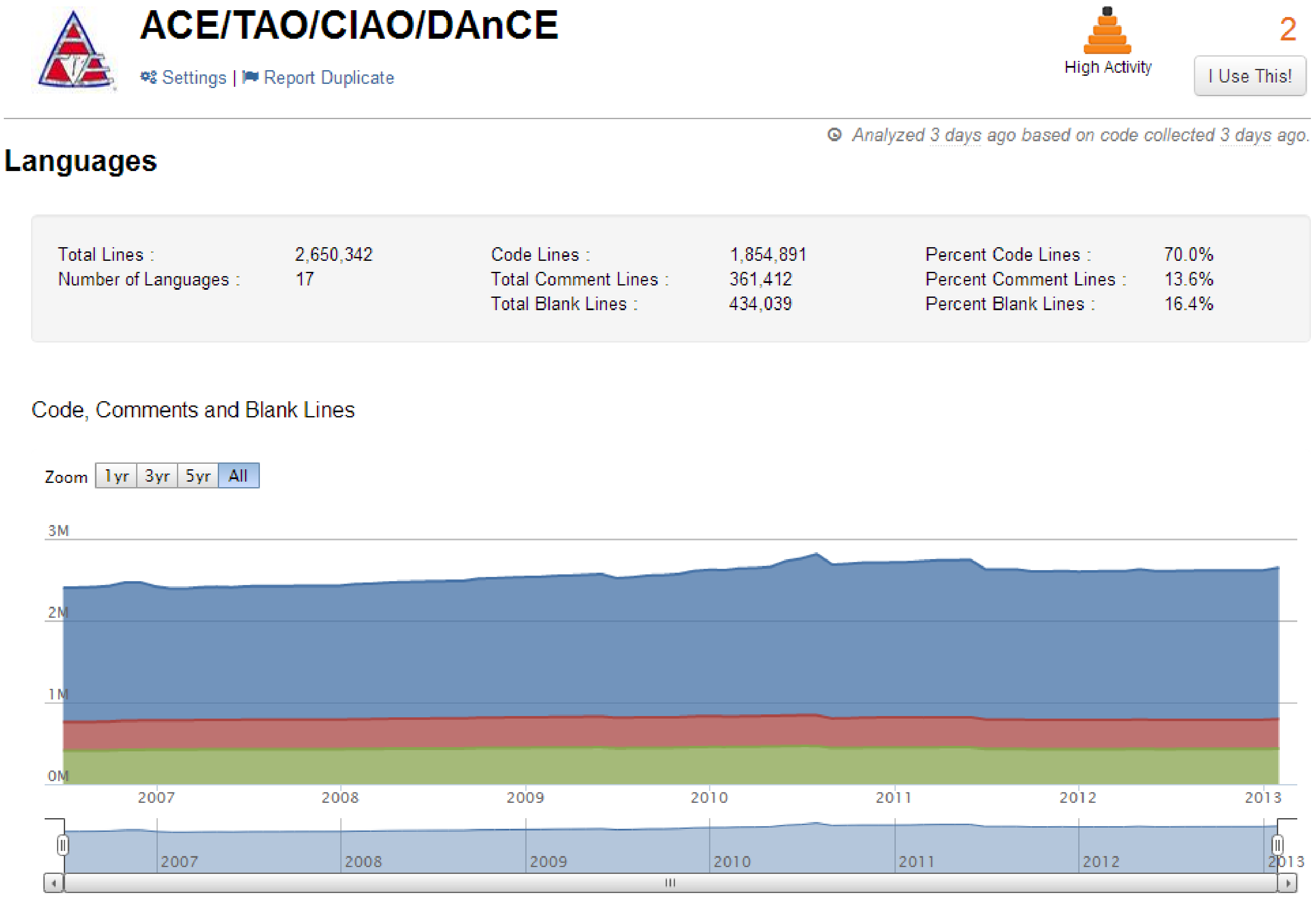
Click a graph to view enlargement.

ACE tests (255)	Test status 97% 3%
TAO core tests (407)	Test status 97% 3%
TAO ORBsvcs tests (151)	Test status 96% 4%
Other tests (262)	Test status 94% 6%

scoreboard.theaceorb.nl/test_stats

Limitations of Frameworks

- Significant time may required to learn how to use frameworks effectively



ACE/TAO/CIAO/DAnCE

Analyzed 3 days ago based on code collected 3 days ago.

Languages

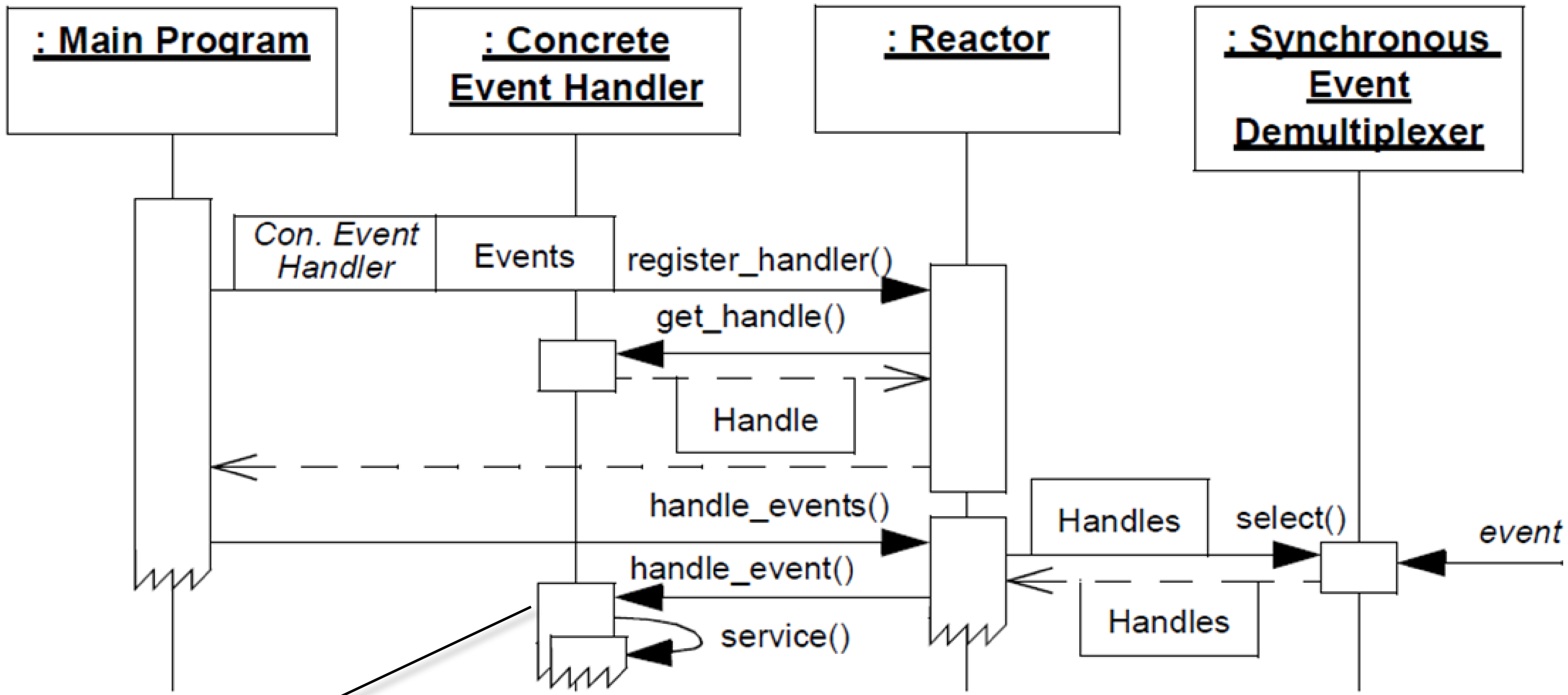
Total Lines :	2,650,342	Code Lines :	1,854,891	Percent Code Lines :	70.0%
Number of Languages :	17	Total Comment Lines :	361,412	Percent Comment Lines :	13.6%
		Total Blank Lines :	434,039	Percent Blank Lines :	16.4%

Code, Comments and Blank Lines

www.ohloh.net/p/acetaociao

Limitations of Frameworks

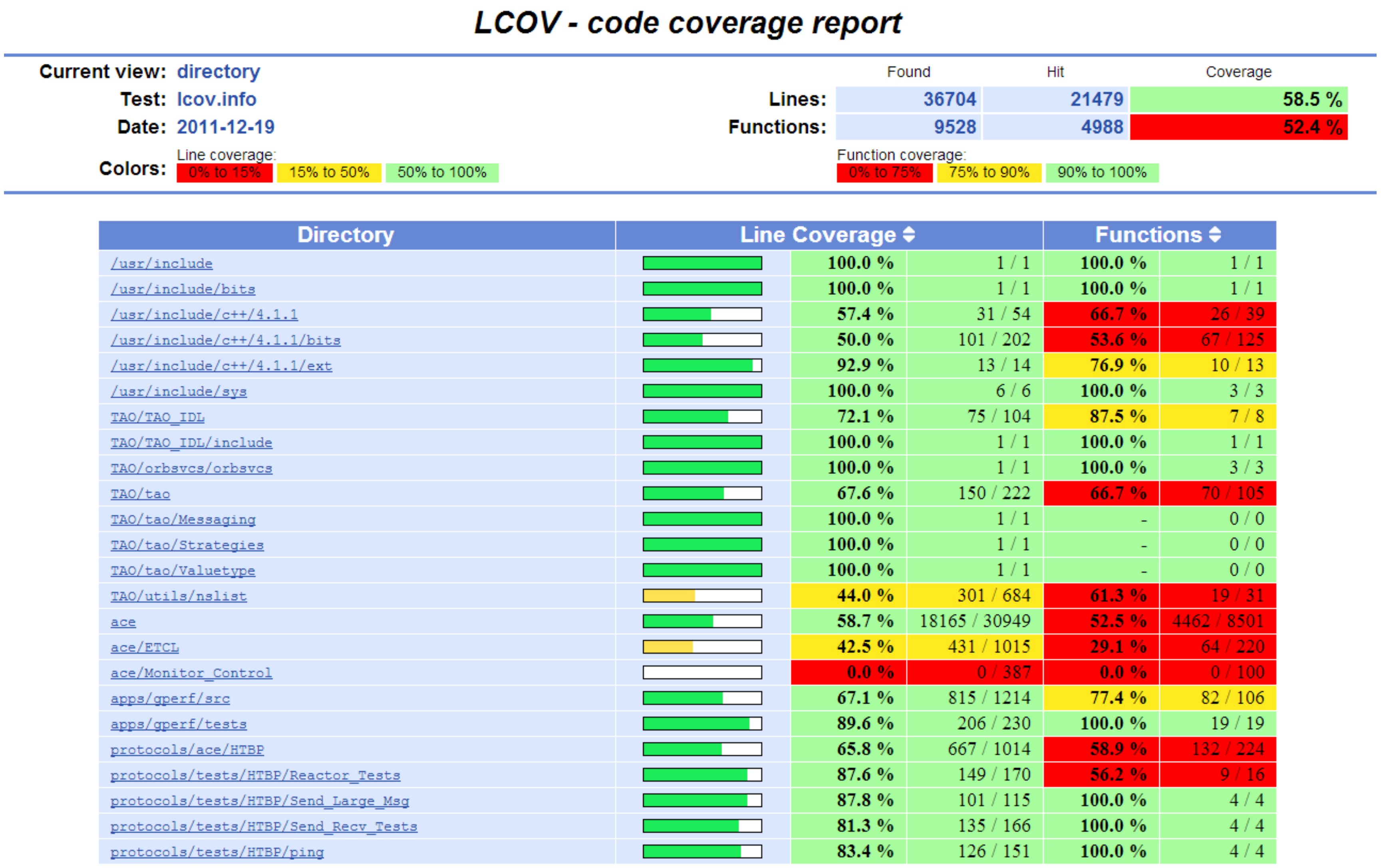
- Polymorphism, inversion of control, & concurrent (especially asynchronous) processing makes debugging tedious & challenging



Note inversion of control

Limitations of Frameworks

- Testing can be tricky due to "late binding"



LCOV - code coverage report

			Found	Hit	Coverage
Current view:	directory				
Test:	lcov.info	**Lines:**	36704	21479	58.5 %
Date:	2011-12-19	**Functions:**	9528	4988	52.4 %

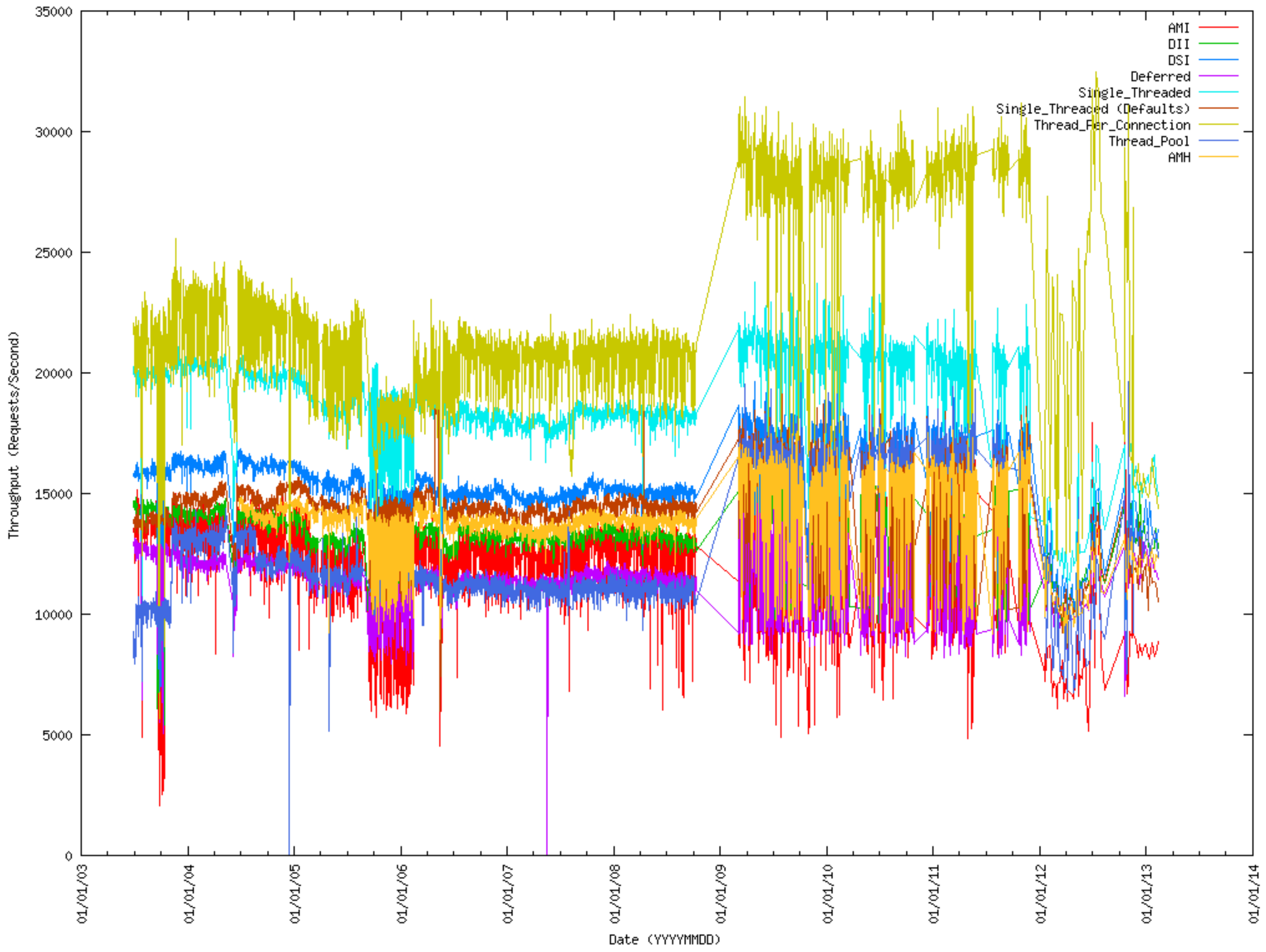
Colors: Line coverage: 🟥 0% to 15% 🟨 15% to 50% 🟩 50% to 100% Function coverage: 🟥 0% to 75% 🟨 75% to 90% 🟩 90% to 100%

Directory	Line Coverage			Functions	
/usr/include		100.0 %	1 / 1	100.0 %	1 / 1
/usr/include/bits		100.0 %	1 / 1	100.0 %	1 / 1
/usr/include/c++/4.1.1		57.4 %	31 / 54	66.7 %	26 / 39
/usr/include/c++/4.1.1/bits		50.0 %	101 / 202	53.6 %	67 / 125
/usr/include/c++/4.1.1/ext		92.9 %	13 / 14	76.9 %	10 / 13
/usr/include/sys		100.0 %	6 / 6	100.0 %	3 / 3
TAO/TAO_IDL		72.1 %	75 / 104	87.5 %	7 / 8
TAO/TAO_IDL/include		100.0 %	1 / 1	100.0 %	1 / 1
TAO/orbsvcs/orbsvcs		100.0 %	1 / 1	100.0 %	3 / 3
TAO/tao		67.6 %	150 / 222	66.7 %	70 / 105
TAO/tao/Messaging		100.0 %	1 / 1	-	0 / 0
TAO/tao/Strategies		100.0 %	1 / 1	-	0 / 0
TAO/tao/Valuetype		100.0 %	1 / 1	-	0 / 0
TAO/utils/nslist		44.0 %	301 / 684	61.3 %	19 / 31
ace		58.7 %	18165 / 30949	52.5 %	4462 / 8501
ace/ETCL		42.5 %	431 / 1015	29.1 %	64 / 220
ace/Monitor_Control		0.0 %	0 / 387	0.0 %	0 / 100
apps/gperf/src		67.1 %	815 / 1214	77.4 %	82 / 106
apps/gperf/tests		89.6 %	206 / 230	100.0 %	19 / 19
protocols/ace/HTBP		65.8 %	667 / 1014	58.9 %	132 / 224
protocols/tests/HTBP/Reactor_Tests		87.6 %	149 / 170	56.2 %	9 / 16
protocols/tests/HTBP/Send_Large_Msg		87.8 %	101 / 115	100.0 %	4 / 4
protocols/tests/HTBP/Send_Recv_Tests		81.3 %	135 / 166	100.0 %	4 / 4
protocols/tests/HTBP/ping		83.4 %	126 / 151	100.0 %	4 / 4

www.dre.vanderbilt.edu/Coverage

Limitations of Frameworks

- Performance may degrade due to complex structures & levels of indirection

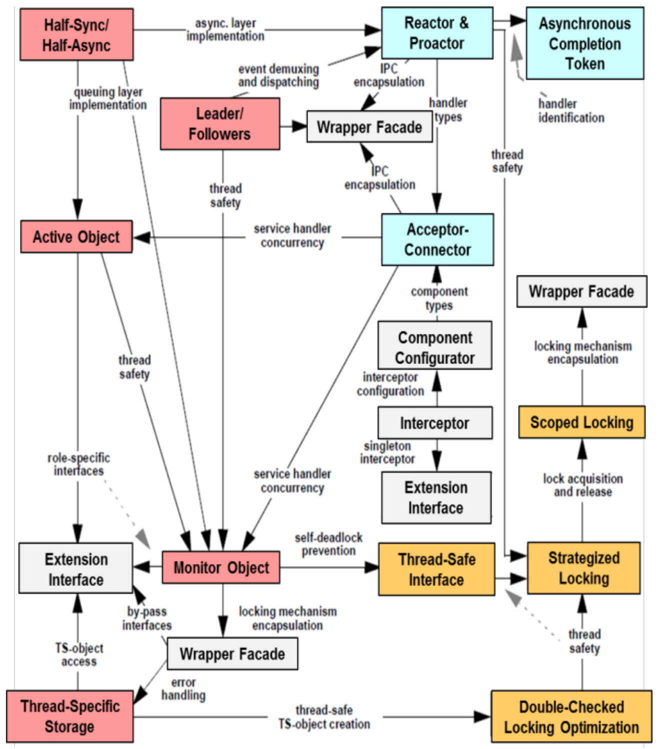


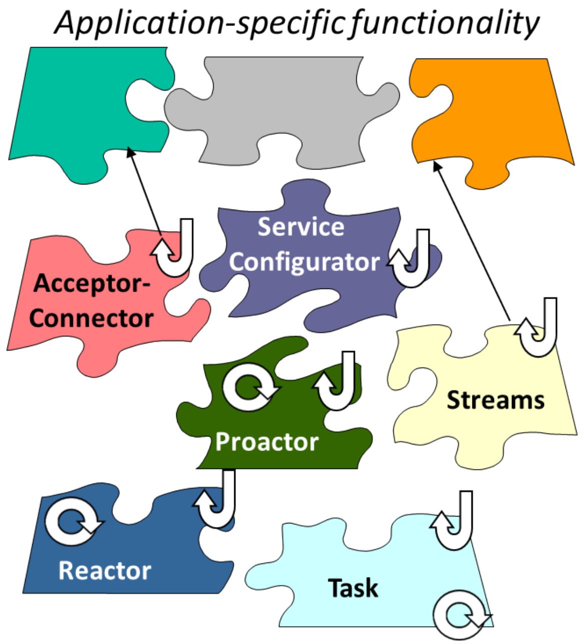
atcdstats.theaceorb.nl

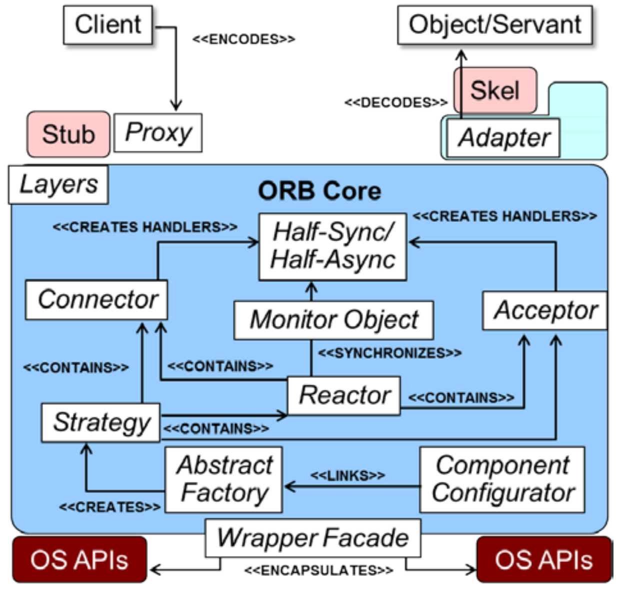
Summary

Patterns, frameworks, & middleware are synergistic

- Patterns codify expertise via reusable architecture design themes & styles
- Frameworks codify expertise via reuse of algorithms, extensible architectures, & components
- Middleware codifies expertise via common components that provide a façade to framework capabilities



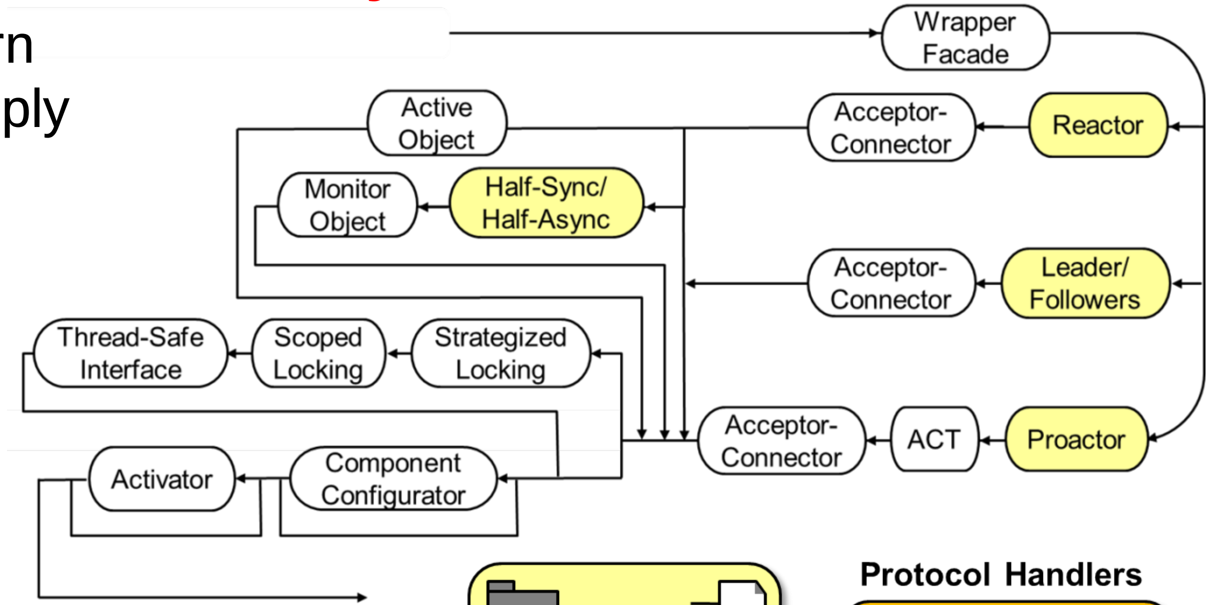


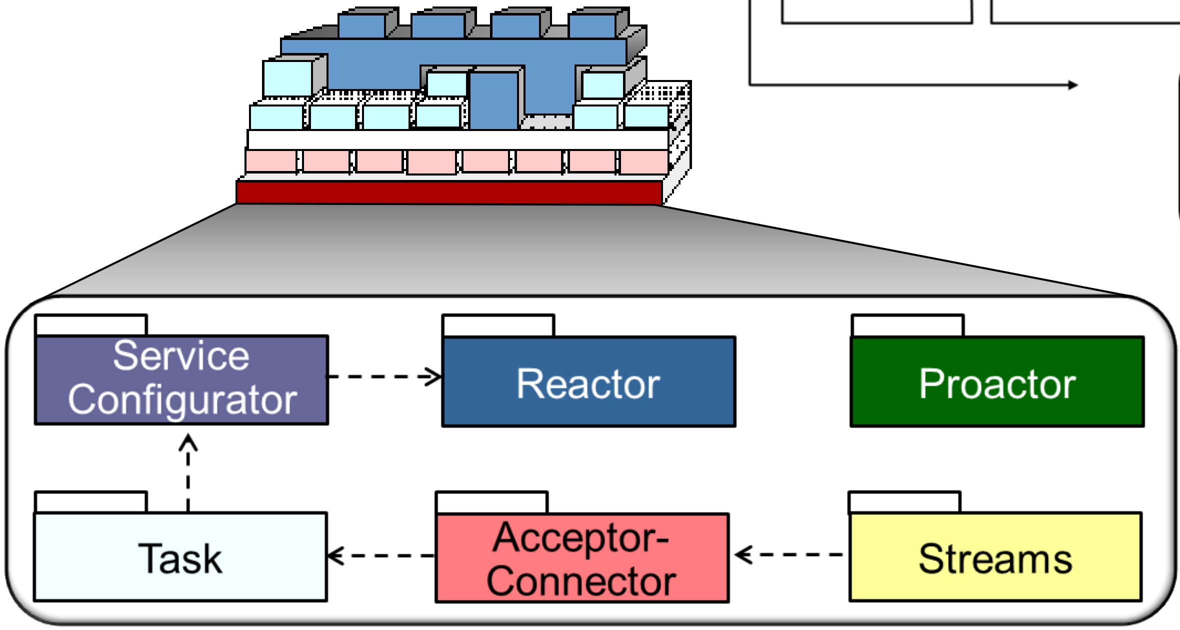


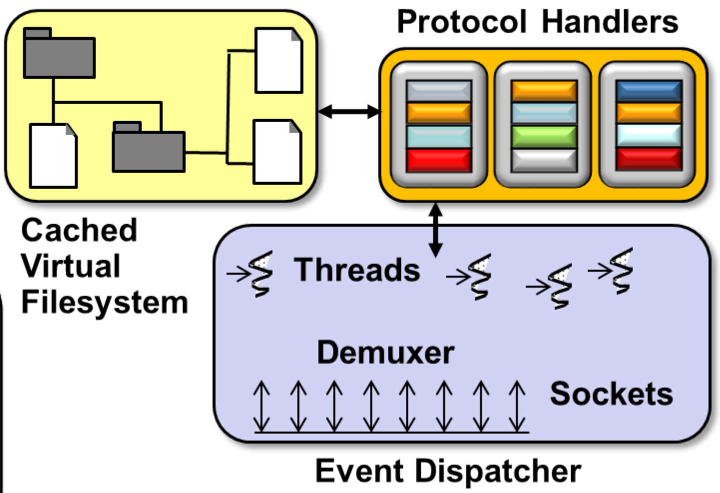
There are now powerful feedback loops advancing these technologies

Summary

- Understand patterns to learn how to better develop & apply frameworks & apps
  - Some of the most useful patterns are used to describe frameworks







Patterns also apply to non-framework technologies, as well