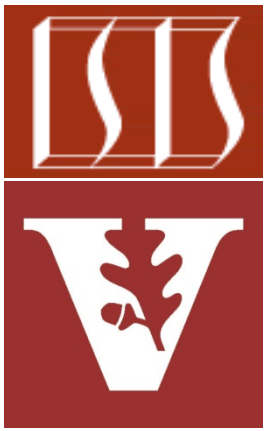
Benefits & Limitations of Patterns & Frameworks: Part 2

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



Professor of Computer Science

Institute for Software Integrated Systems

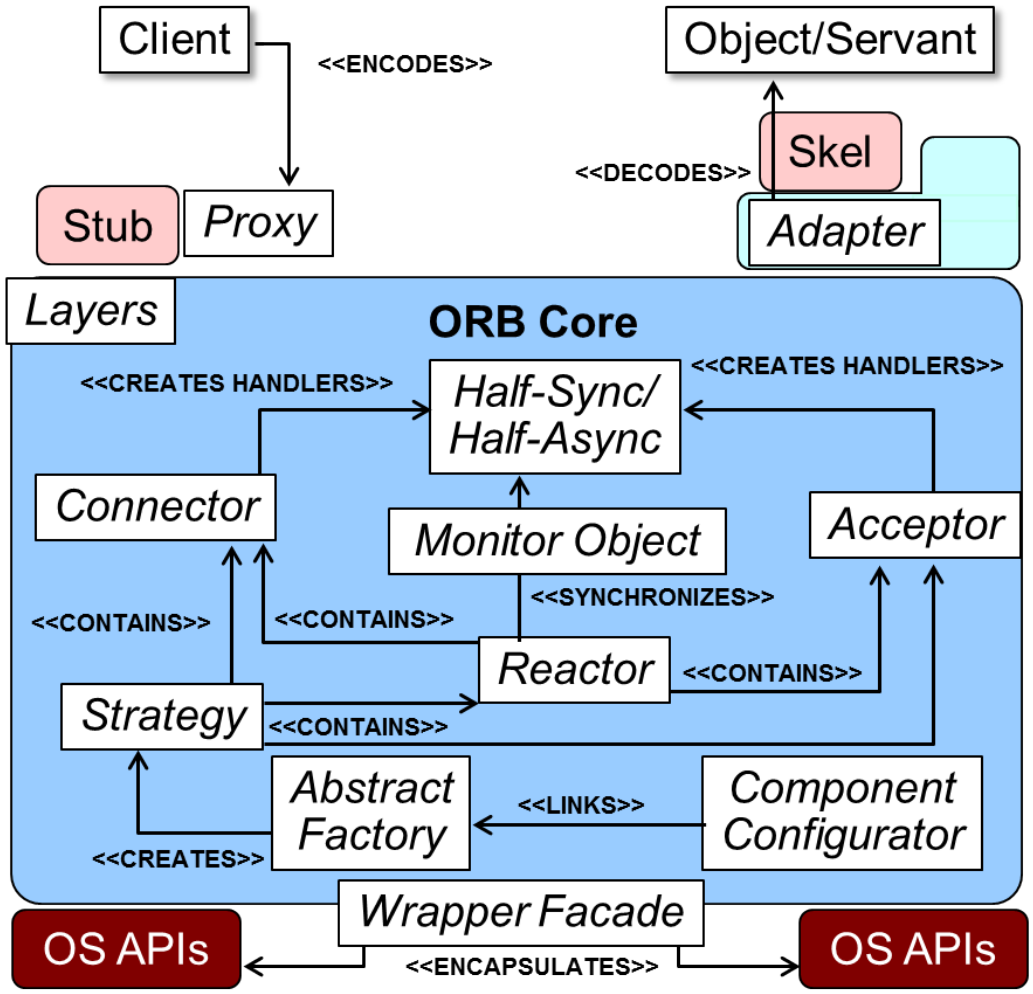
Vanderbilt University
Nashville, Tennessee, USA

Topics Covered in this Part of the Module

- Summarize the benefits & limitations of patterns
- Summarize the benefits & limitations of frameworks

Benefits of Frameworks

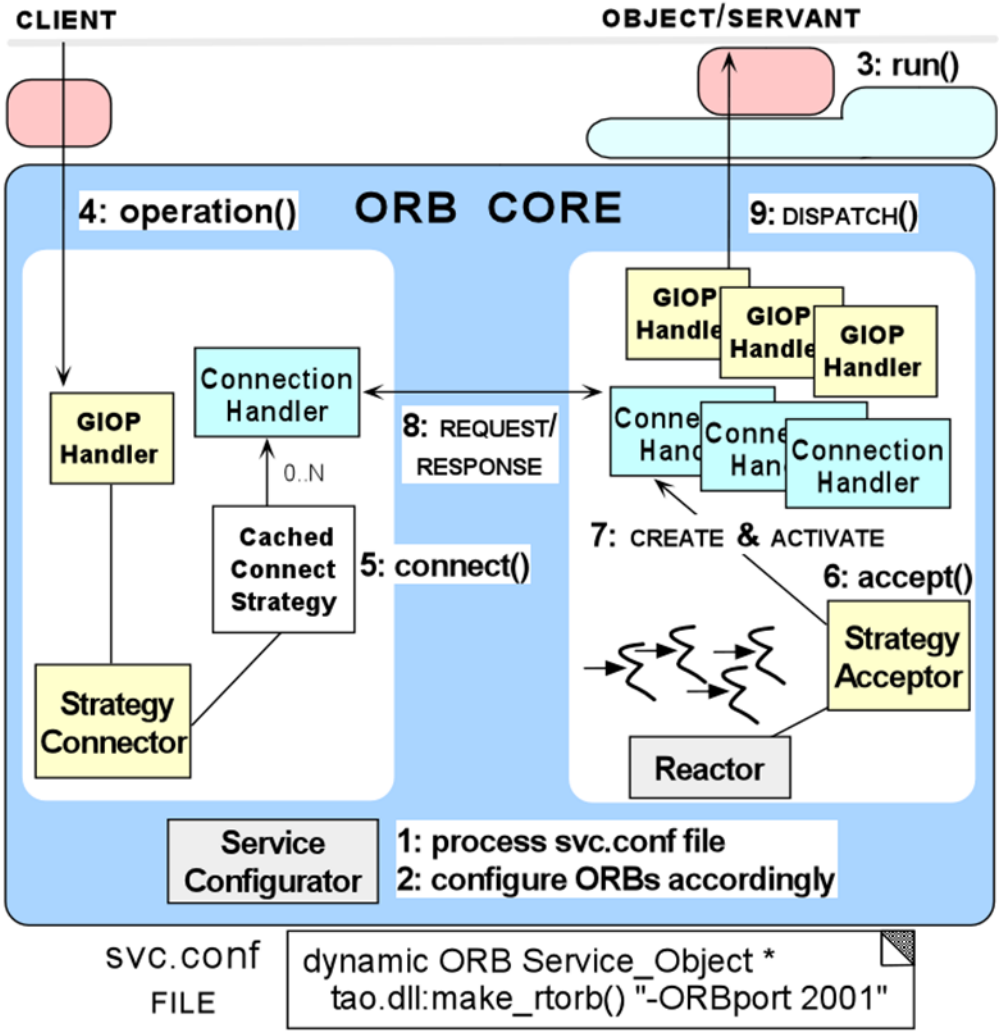
- **Design reuse**
  - e.g., by guiding app developers thru steps needed to ensure successful creation & deployment of software



See www.dre.vanderbilt.edu/~schmidt/PDF/ORB-patterns.pdf for more info

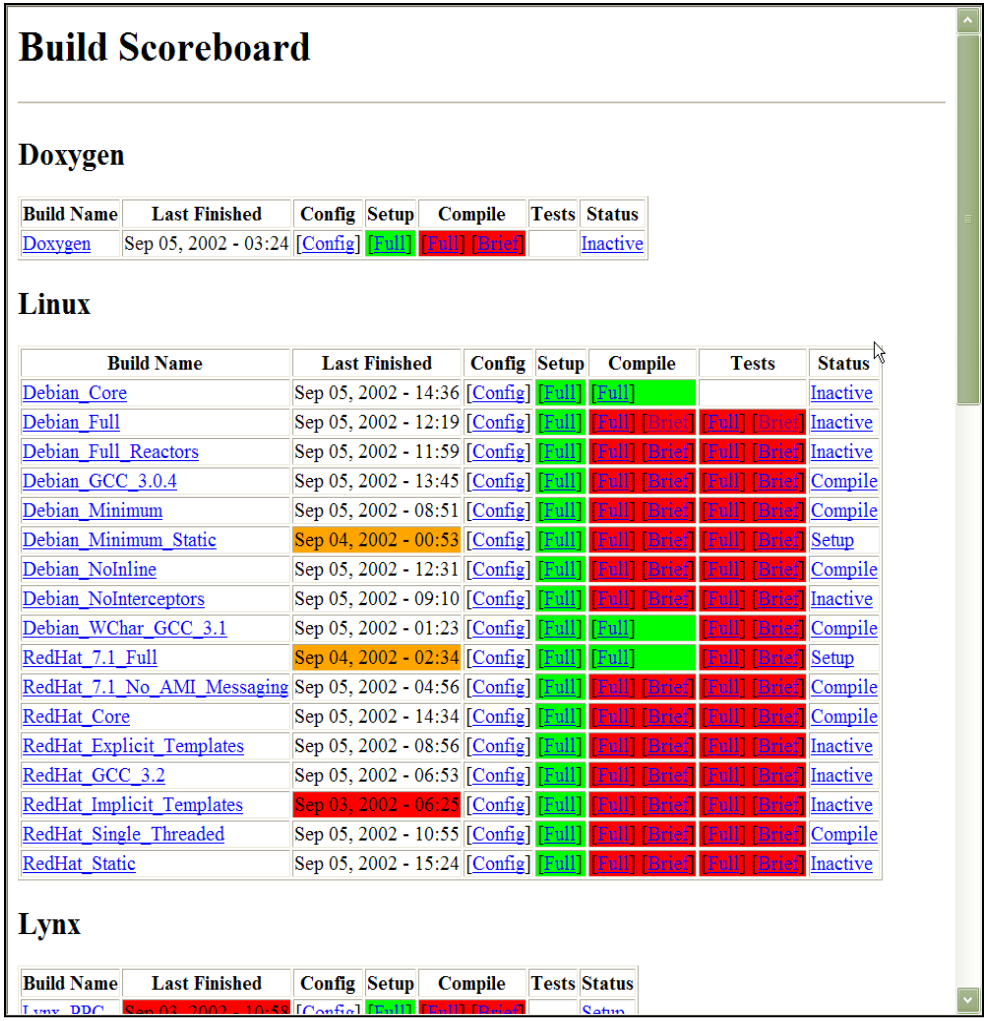
Benefits of Frameworks

- **Design reuse**
  - e.g., by guiding app developers thru steps needed to ensure successful creation & deployment of software
- **Implementation reuse**
  - e.g., by leveraging previous development & optimization efforts & amortizing software lifecycle costs



Benefits of Frameworks

- Design reuse
  - e.g., by guiding app developers thru steps needed to ensure successful creation & deployment of software
- Implementation reuse
  - e.g., by leveraging previous development & optimization efforts & amortizing software lifecycle costs
- **Validation reuse**
  - e.g., by amortizing the efforts of validating application- & platform-independent portions of software, thereby enhancing dependability & performance



Build Scoreboard

Doxygen

Build Name	Last Finished	Config	Setup	Compile	Tests	Status
Doxygen	Sep 05, 2002 - 03:24	[Config]	[Full]	[Full] [Brief]		Inactive

Linux

Build Name	Last Finished	Config	Setup	Compile	Tests	Status
Debian_Core	Sep 05, 2002 - 14:36	[Config]	[Full]	[Full]		Inactive
Debian_Full	Sep 05, 2002 - 12:19	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Inactive
Debian_Full_Reactors	Sep 05, 2002 - 11:59	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Inactive
Debian_GCC_3.0.4	Sep 05, 2002 - 13:45	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Compile
Debian_Minimum	Sep 05, 2002 - 08:51	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Compile
Debian_Minimum_Static	Sep 04, 2002 - 00:53	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Setup
Debian_NoInline	Sep 05, 2002 - 12:31	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Compile
Debian_NoInterceptors	Sep 05, 2002 - 09:10	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Inactive
Debian_WChar_GCC_3.1	Sep 05, 2002 - 01:23	[Config]	[Full]	[Full]	[Full] [Brief]	Compile
RedHat_7.1_Full	Sep 04, 2002 - 02:34	[Config]	[Full]	[Full]	[Full] [Brief]	Setup
RedHat_7.1_No_AMI_Messaging	Sep 05, 2002 - 04:56	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Compile
RedHat_Core	Sep 05, 2002 - 14:34	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Compile
RedHat_Explicit_Templates	Sep 05, 2002 - 08:56	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Inactive
RedHat_GCC_3.2	Sep 05, 2002 - 06:53	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Inactive
RedHat_Implicit_Templates	Sep 03, 2002 - 06:25	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Inactive
RedHat_Single_Threaded	Sep 05, 2002 - 10:55	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Compile
RedHat_Static	Sep 05, 2002 - 15:24	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	Inactive

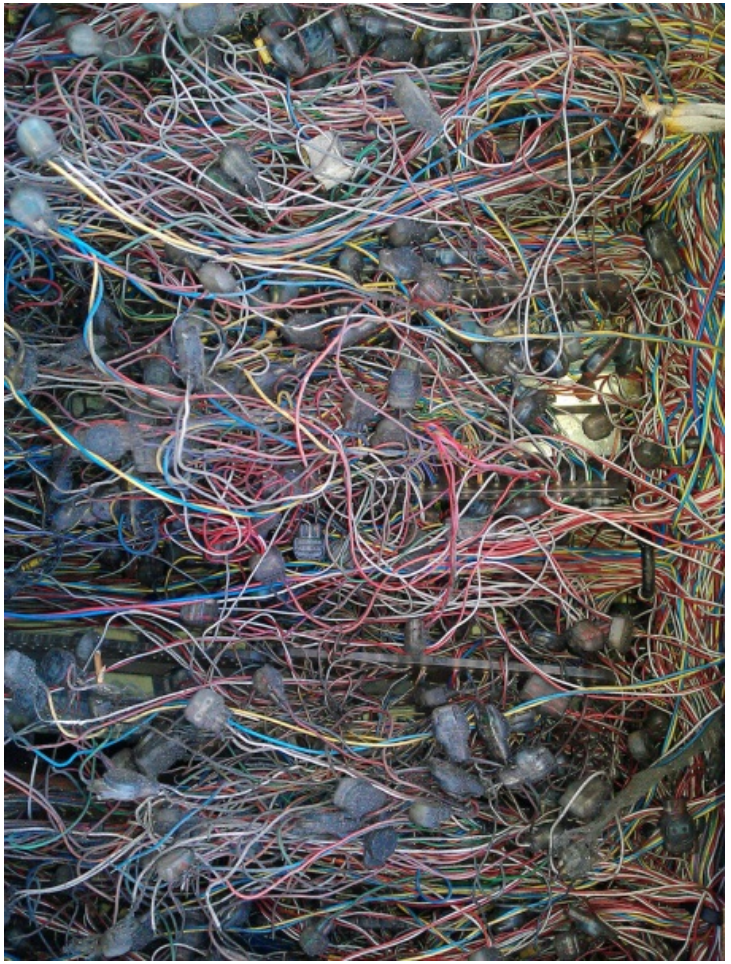
Lynx

Build Name	Last Finished	Config	Setup	Compile	Tests	Status
Lynx_PPC	Sep 03, 2002 - 10:58	[Config]	[Full]	[Full] [Brief]		Setup

www.dre.vanderbilt.edu/scoreboard

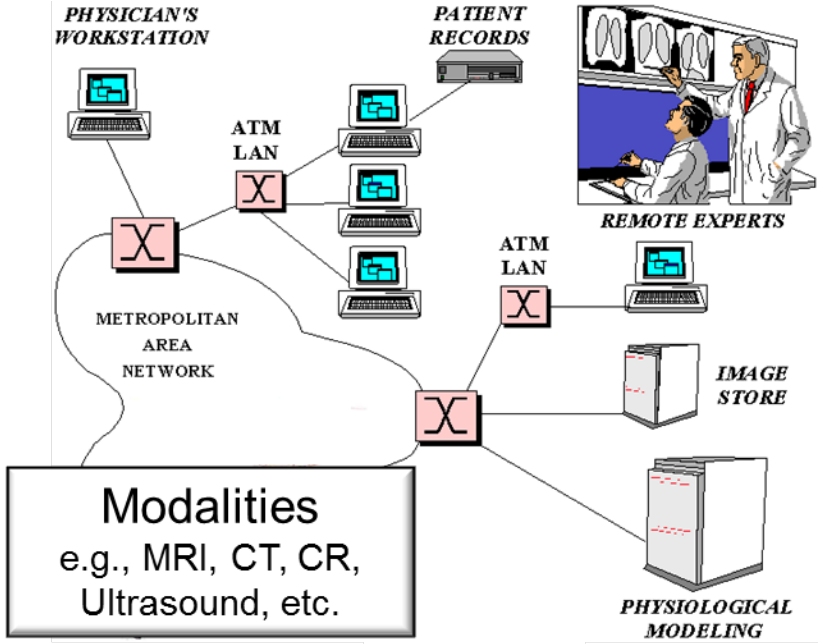
Limitations of Frameworks

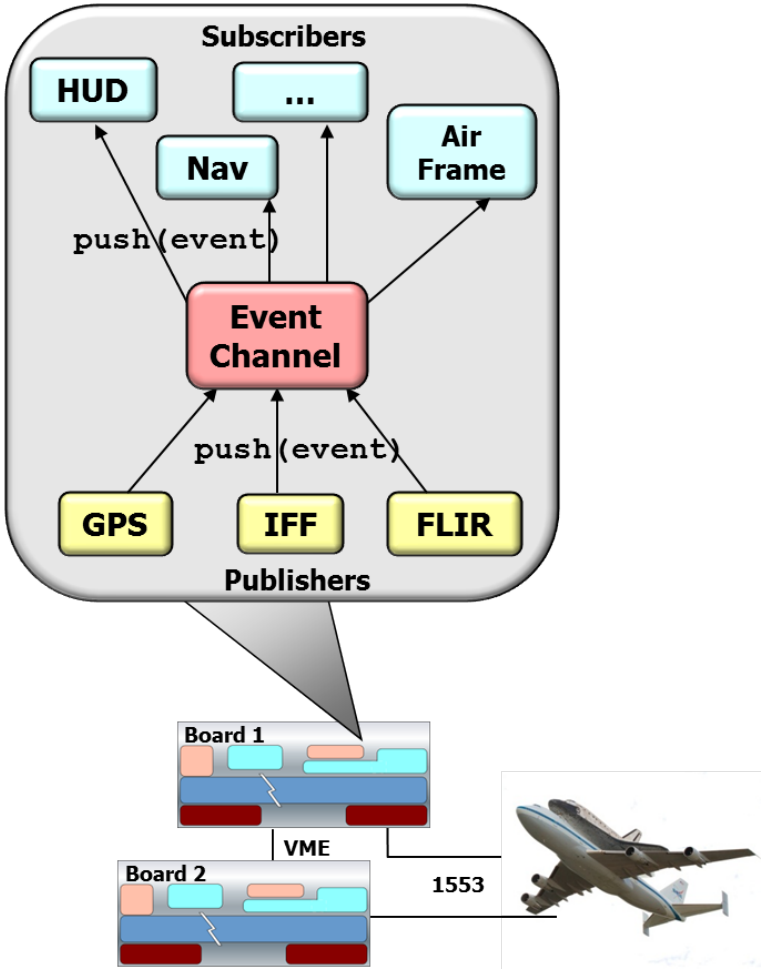
- Frameworks are powerful, but many app developers find them hard to create/use effectively



Limitations of Frameworks

- Frameworks are powerful, but many app developers find them hard to create/use effectively
  - Scope/Commonality/Variability analysis requires significant domain knowledge & reusable software development expertise





Limitations of Frameworks

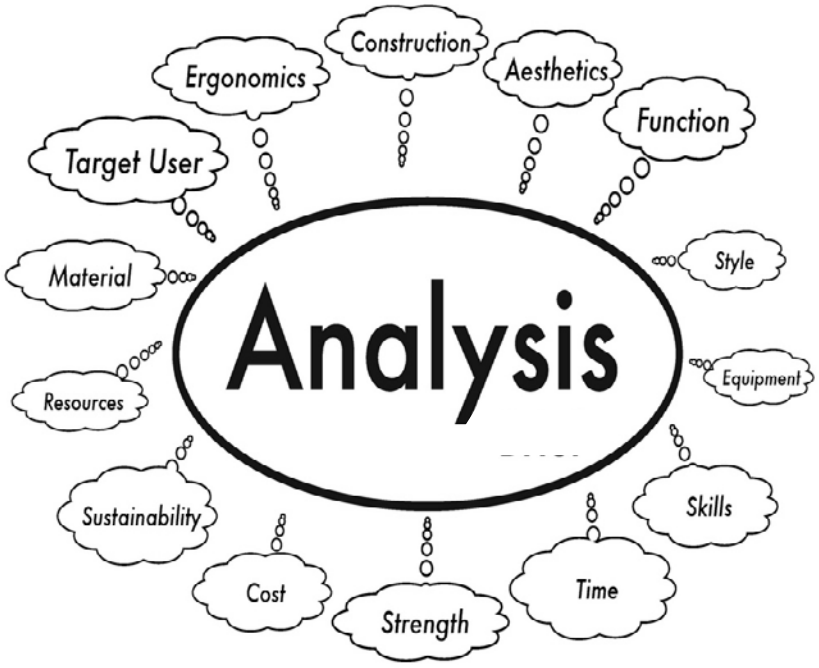
- Frameworks are powerful, but many app developers find them hard to create/use effectively
  - Scope/Commonality/Variability analysis requires significant domain knowledge & reusable software development expertise
  - Developing frameworks in non-OO languages is even harder

```
static int cmpstrs (const void *p1, const void *p2)
{ return strcmp (*(char* const*) p1, * (char* const*) p2); }

static int casecmpstrs (const void *p1, const void *p2)
{ return strcasecmp (*(char* const*) p1, * (char* const*) p2); }

int main(int argc, char *argv[]) {
  int j, icase = *argv[1] == 'i';
  qsort (&argv[2], argc - 1, sizeof(char *),
         icase ? casecmpstrs : cmpstrs);
  for (j = 2; j < argc; ++j) puts(argv[j]);
}
```

Strategy pattern

Limitations of Frameworks

- Frameworks are powerful, but many app developers find them hard to create/use effectively
  - Scope/Commonality/Variability analysis requires significant domain knowledge & reusable software development expertise
  - Developing frameworks in non-OO languages is even harder
- Significant time required to evaluate applicability & quality of a framework for a particular domain



Limitations of Frameworks

- Frameworks are powerful, but many app developers find them hard to create/use effectively
  - Scope/Commonality/Variability analysis requires significant domain knowledge & reusable software development expertise
  - Developing frameworks in non-OO languages is even harder
- Significant time required to evaluate applicability & quality of a framework for a particular domain
- Inversion of control makes debugging tricky

Limitations of Frameworks

- Frameworks are powerful, but many app developers fine them hard to create/use effectively
  - Scope/Commonality/Variability analysis requires significant domain knowledge & reusable software development expertise
  - Developing frameworks in non-OO languages is even harder
- Significant time required to evaluate applicability & quality of a framework for a particular domain
- Inversion of control makes debugging tricky
- Testing can be tricky due to “late binding”

Limitations of Frameworks

- Frameworks are powerful, but many app developers fine them hard to create/use effectively
  - Scope/Commonality/Variability analysis requires significant domain knowledge & reusable software development expertise
  - Developing frameworks in non-OO languages is even harder
- Significant time required to evaluate applicability & quality of a framework for a particular domain
- Inversion of control makes debugging tricky
- Testing can be tricky due to "late binding"
- Performance may degrade due to complex structures & extra levels of indirection

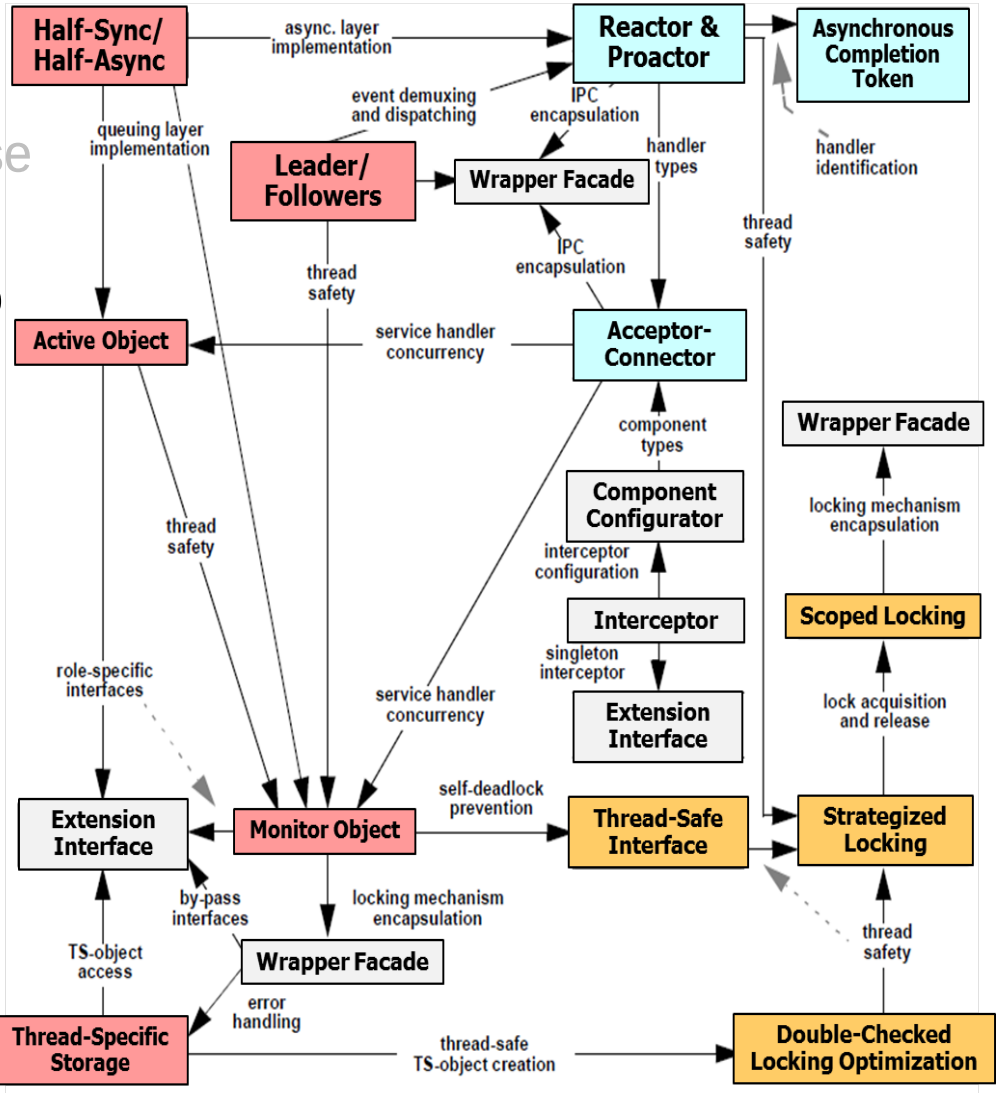
www.dre.vanderbilt.edu/~schmidt/PDF/Queue-04.pdf has more on frameworks

Summary

- Don't apply patterns & frameworks blindly
  - Abstraction/indirection can increase complexity, cost, & confusion

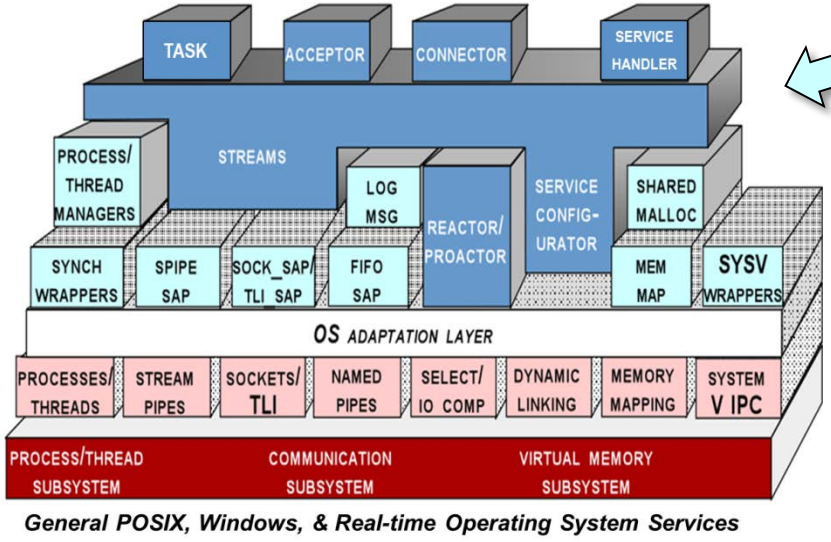
Summary

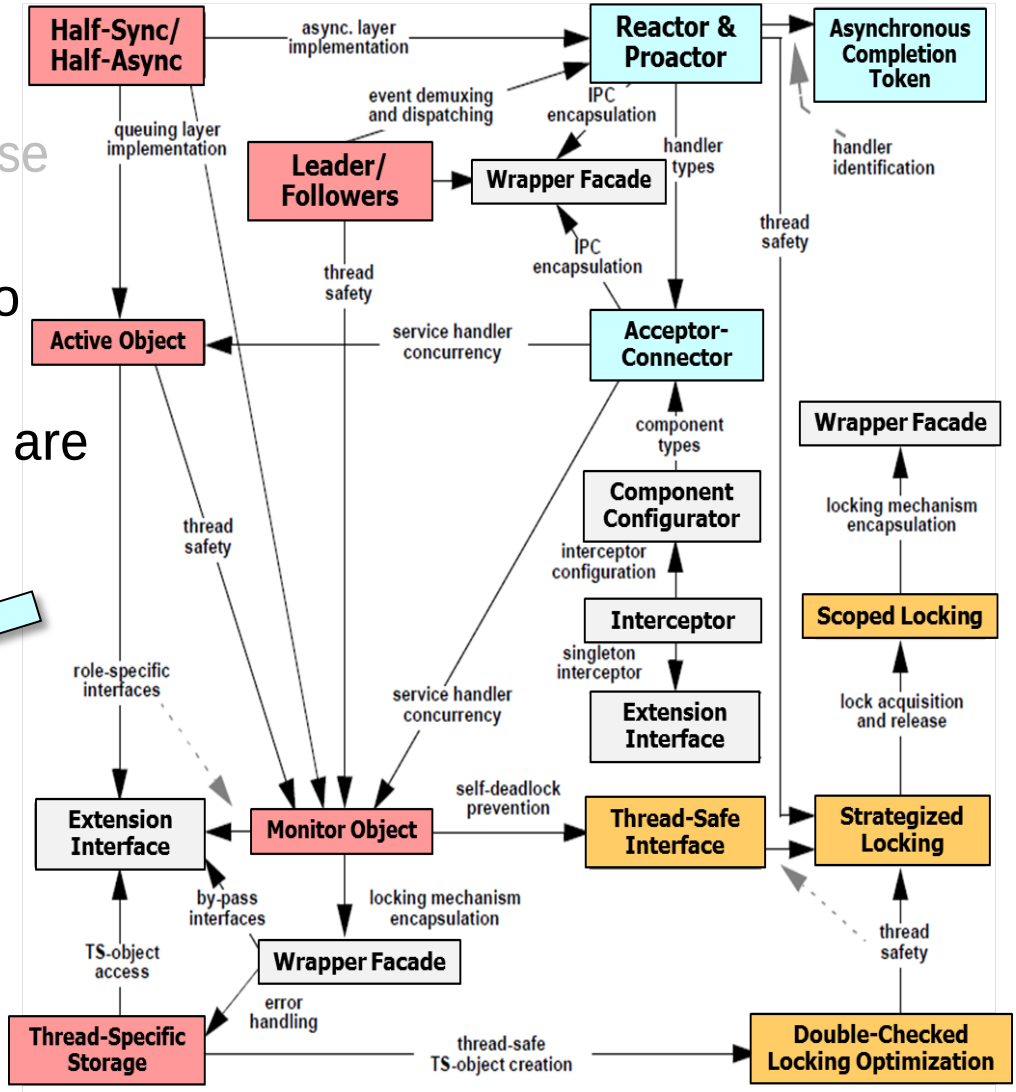
- Don't apply patterns & frameworks blindly
  - Abstraction/indirection can increase complexity, cost, & confusion
- Understand patterns to learn how to better develop & apply frameworks



Summary

- Don't apply patterns & frameworks blindly
  - Abstraction/indirection can increase complexity, cost, & confusion
- Understand patterns to learn how to better develop & apply frameworks
  - Some of the most useful patterns are used to describe frameworks

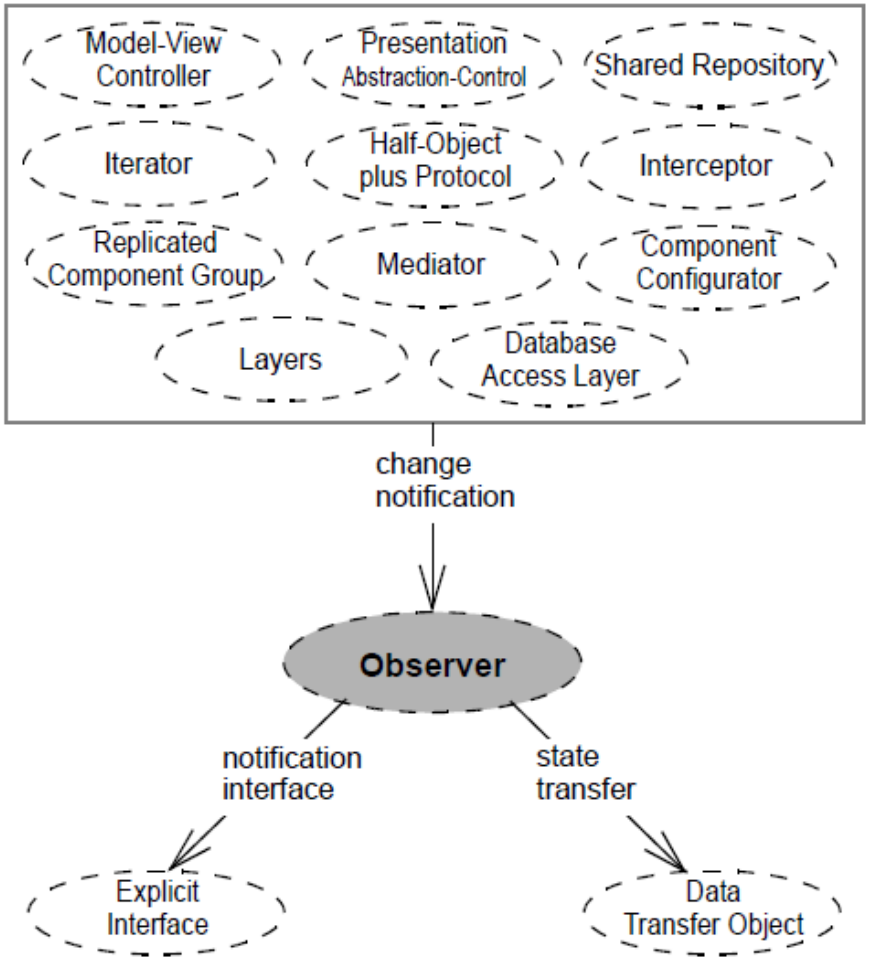




See www.dre.vanderbilt.edu/~schmidt/patterns-ace.html for more info

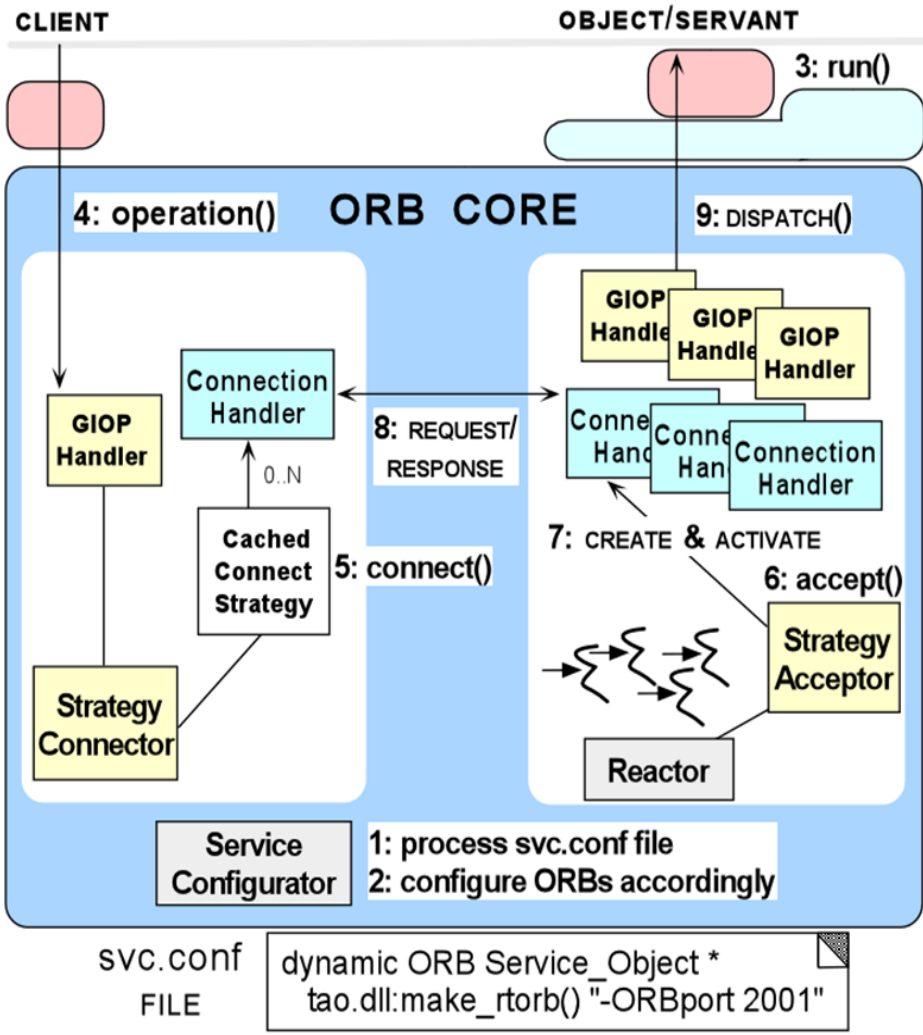
Summary

- Don't apply patterns & frameworks blindly
  - Abstraction/indirection can increase complexity, cost, & confusion
- Understand patterns to learn how to better develop & apply frameworks
  - Some of the most useful patterns are used to describe frameworks
- Patterns can be viewed as abstract descriptions of frameworks that enable broad reuse of software architecture



Summary

- Don't apply patterns & frameworks blindly
  - Abstraction/indirection can increase complexity, cost, & confusion
- Understand patterns to learn how to better develop & apply frameworks
  - Some of the most useful patterns are used to describe frameworks
- Patterns can be viewed as abstract descriptions of frameworks that enable broad reuse of software architecture
- Frameworks can be seen as concrete realizations of patterns that facilitate direct reuse of design & code



Summary

- Don't apply patterns & frameworks blindly
  - Abstraction/indirection can increase complexity, cost, & confusion
- Understand patterns to learn how to better develop & apply frameworks
  - Some of the most useful patterns are used to describe frameworks
- Patterns can be viewed as abstract descriptions of frameworks that enable broad reuse of software architecture
- Frameworks can be seen as concrete realizations of patterns that facilitate direct reuse of design & code
- Pattern & framework design is even harder than OO design!

Many frameworks limitations can be addressed with knowledge of patterns