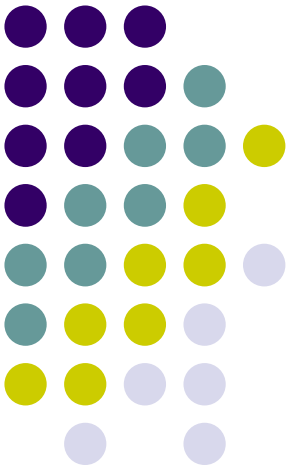
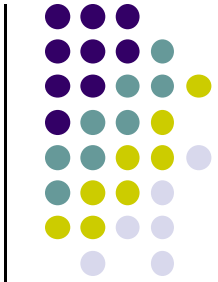
Software Architectures:
Concepts, Lessons Learned, Extensions to CPS



Workshop on Architectures for Cyber-Physical Systems

David Garlan & Bruce Krogh
Carnegie Mellon University

Outline



- What is Software Architecture?
  - Definition & Intent
  - Architecture Representations as Models
  - Potential Benefits
- Basic concepts of software architecture
  - Views, Styles, and Tactics
- CPS Architectures
  - Adding physical elements to the models
  - Reconciling multiple views

Examples of Software Architecture

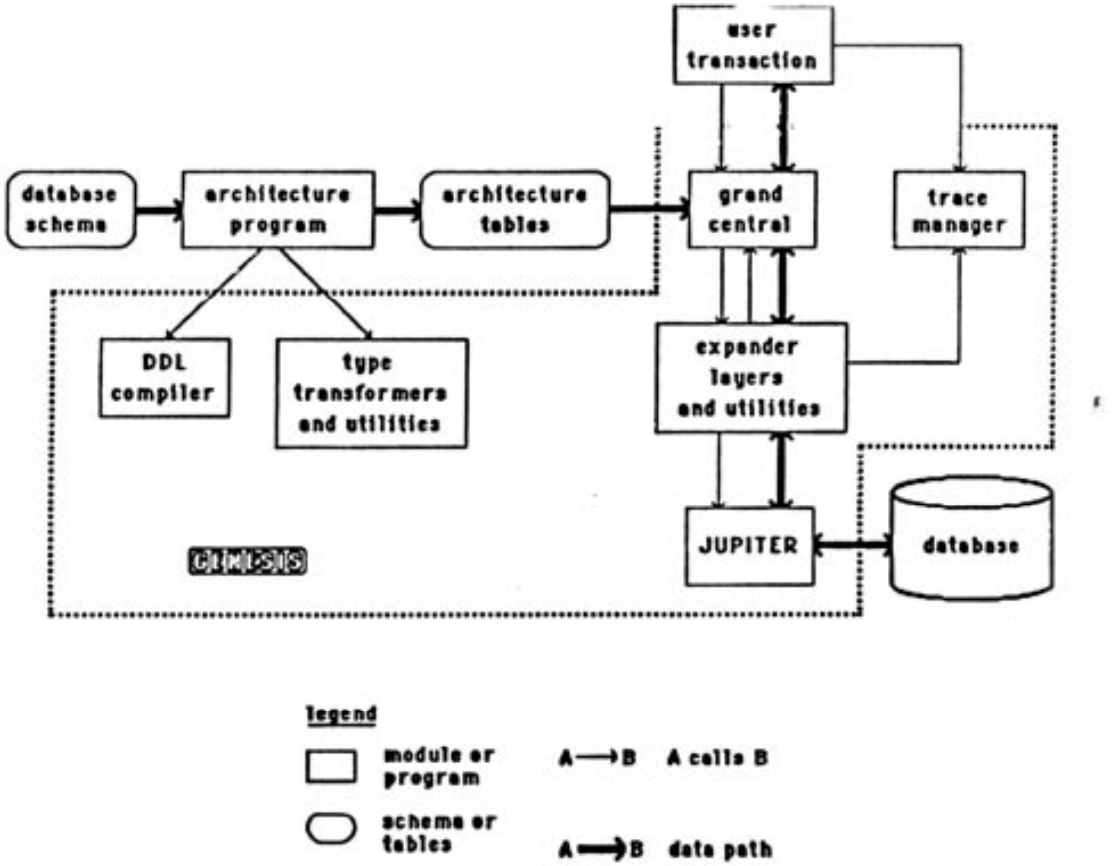


Figure 3.1 The Configuration of the GENESIS Prototype

Genesis: A Reconfiguration Database Management System, D. S. Batory, J.R. Barnett, J.F. Garza, K.P. Smith, K. Tsukuda, B.C. Twichell, T.E. Wise, Department of Computer Sciences, University of Texas at Austin,

More Examples

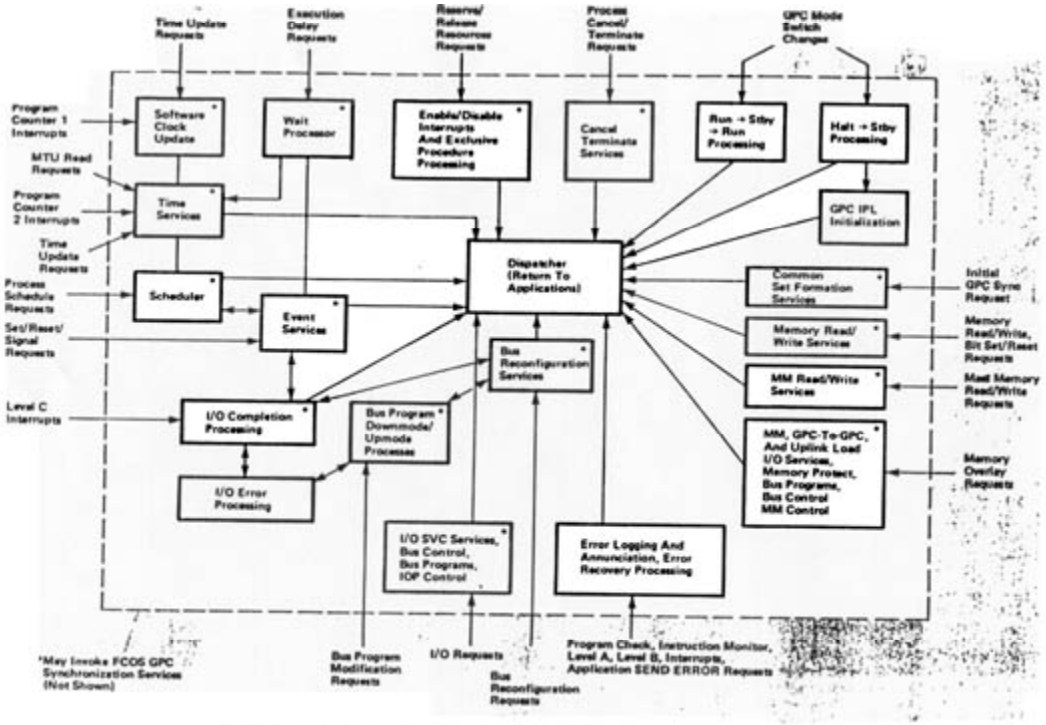


FIGURE 7. Flight Computer Operating System (The FCOS dispatcher coordinates and controls all work performed by the on-board computers.)

Communications of the ACM, "Architecture of the Space Shuttle Primary Avionics Software Systems," Gene D. Carlow, September 1984, Vol. 27, No. 9, P. 933

More Examples

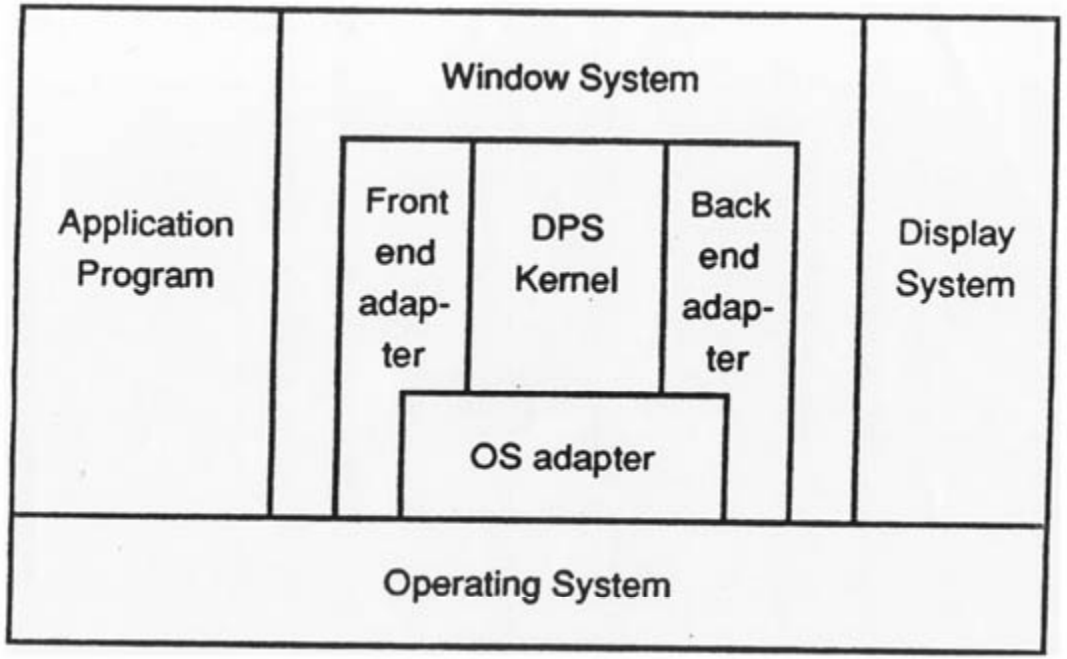


Figure 2. Display PostScript interpreter components.

An Overview of the DISPLAY POSTSCRIPT™ System, Adobe Systems Incorporated, March 16, 1988, P. 10

More Examples

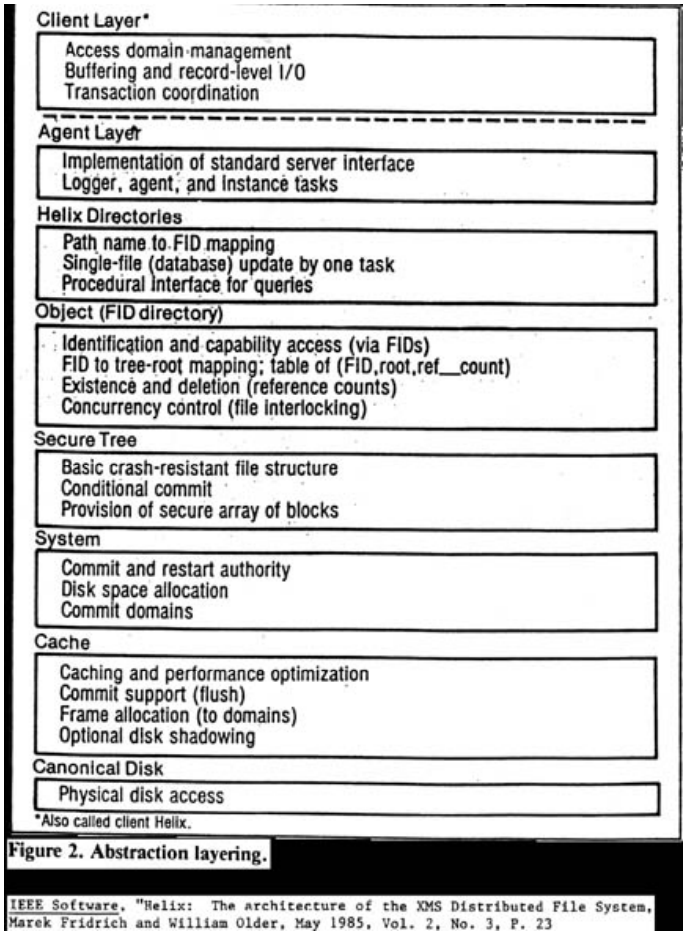
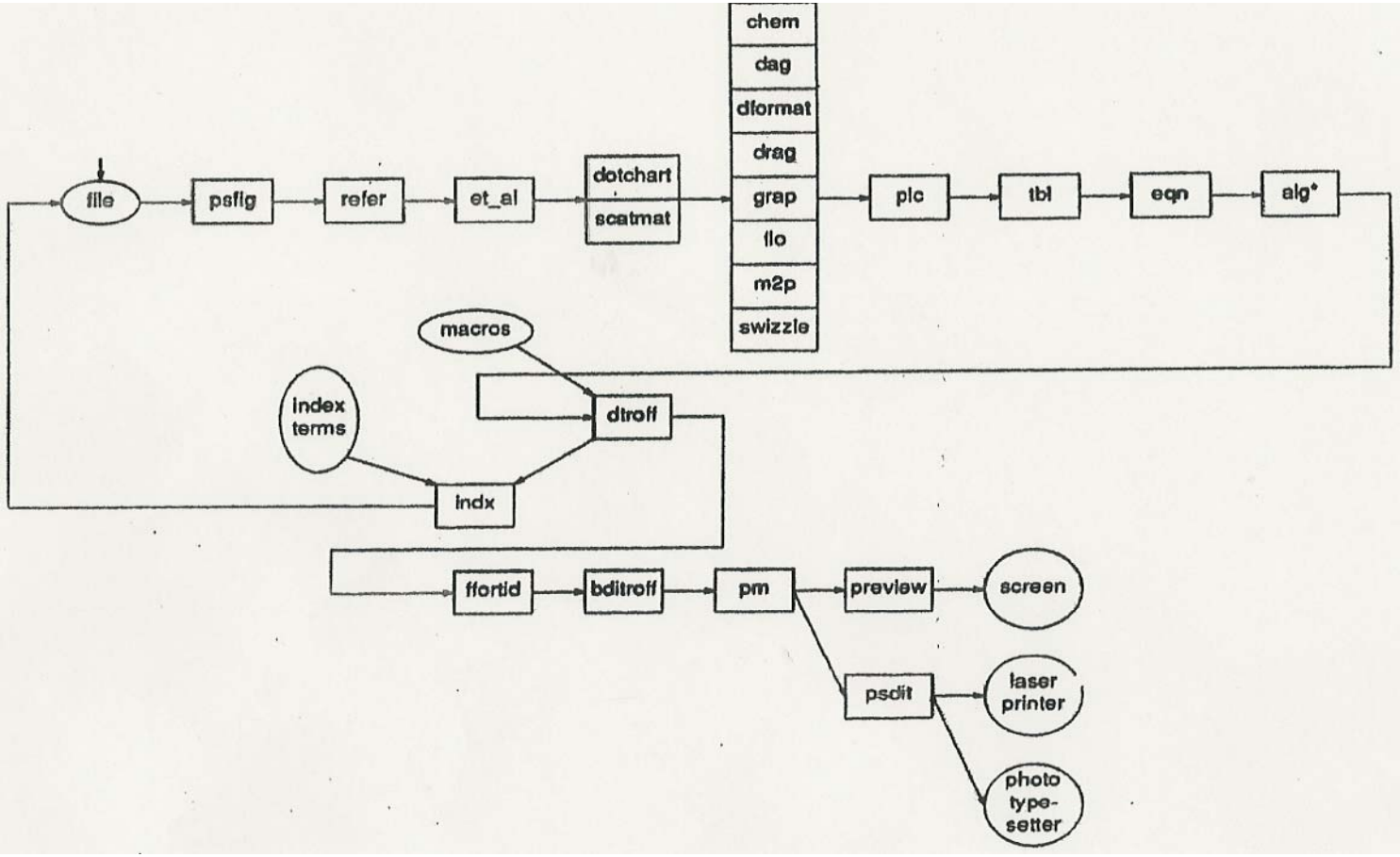


Figure 2. Abstraction layering.

IEEE Software, "Helix: The Architecture of the XMS Distributed File System, Marek Fridrich and William Older, May 1985, Vol. 2, No. 3, P. 23

More Examples



More Examples

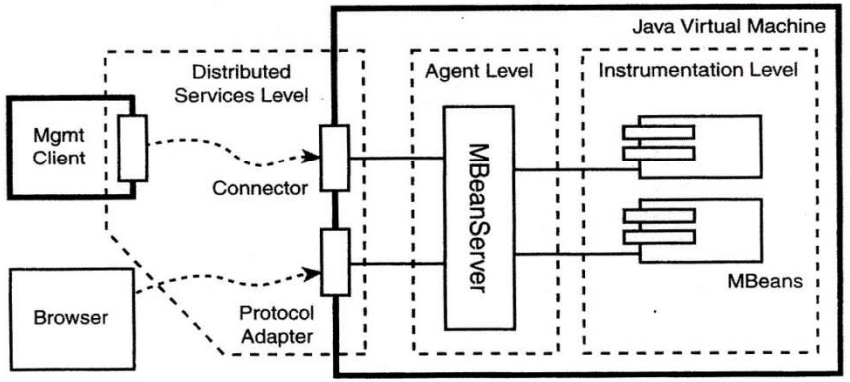
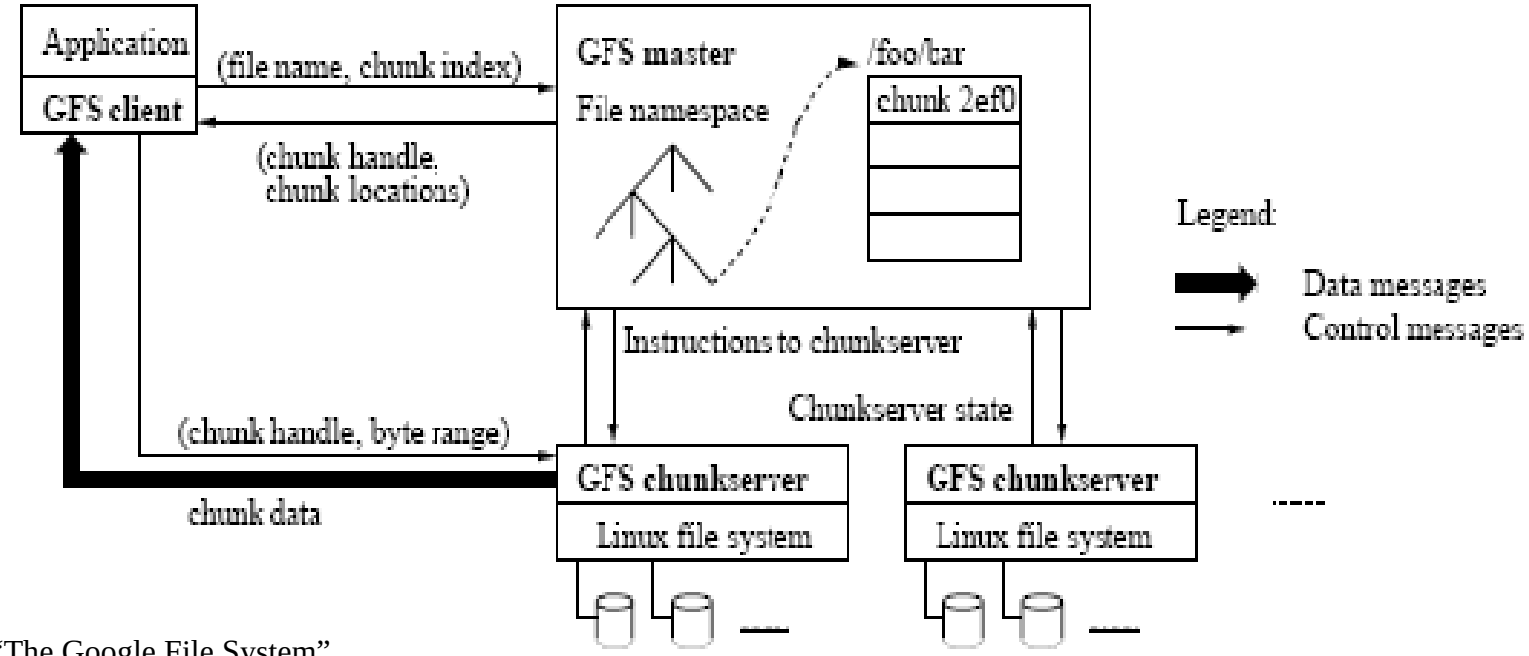


FIGURE 2.1
JMX Management Architecture.

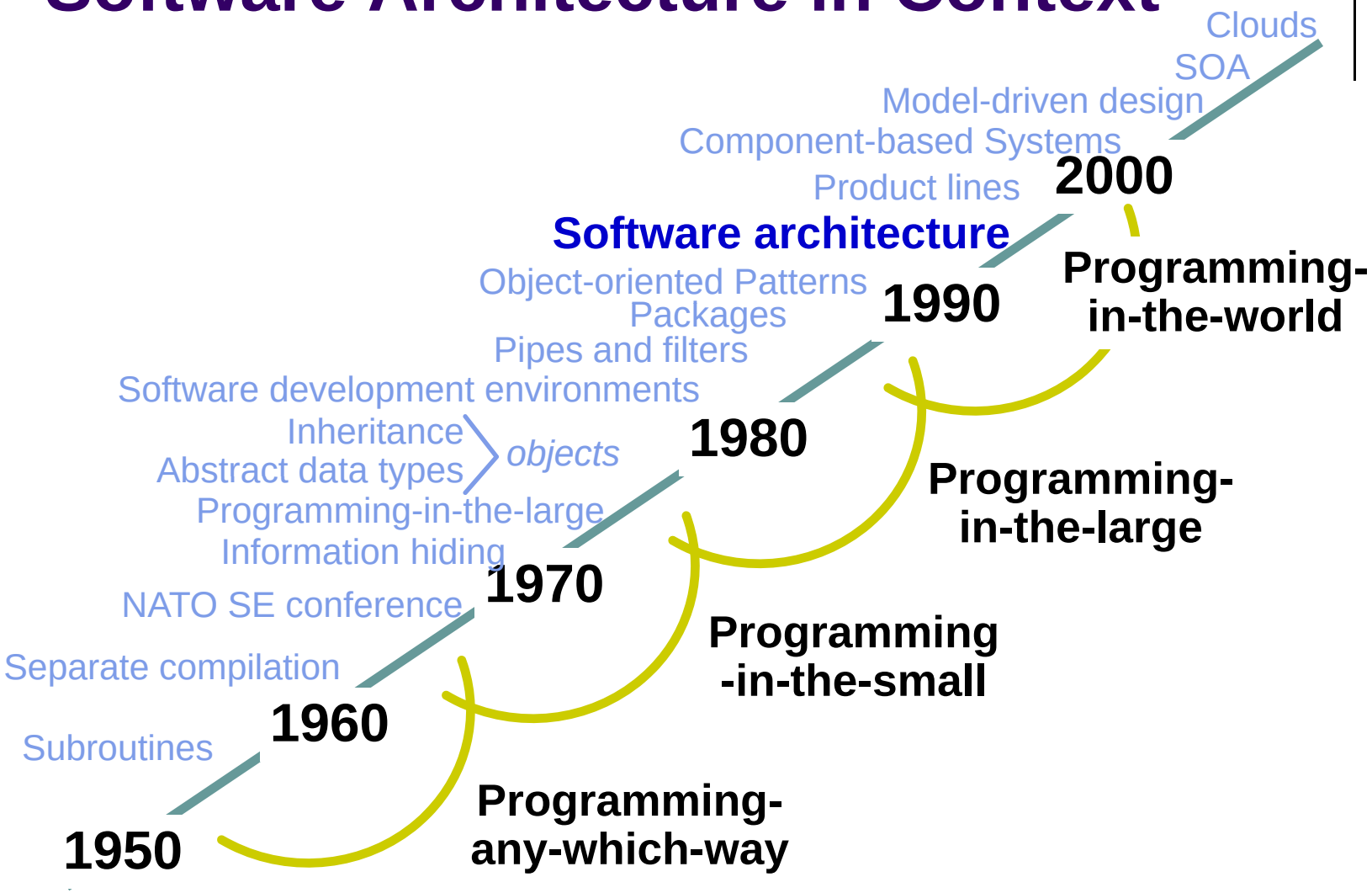
More Examples



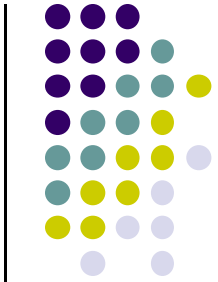
Source: "The Google File System" Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. SOSP 2003.

Figure 1: GFS Architecture

Software Architecture in Context



What is Software Architecture?

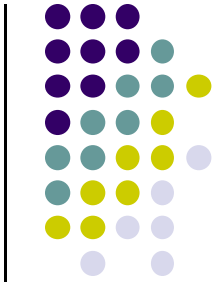


- There are many definitions in the literature
- The definition that I currently use is*

> The software architecture of a computing system is the set of structures needed to reason about the system, which comprise software elements, relations among them and properties of both.*

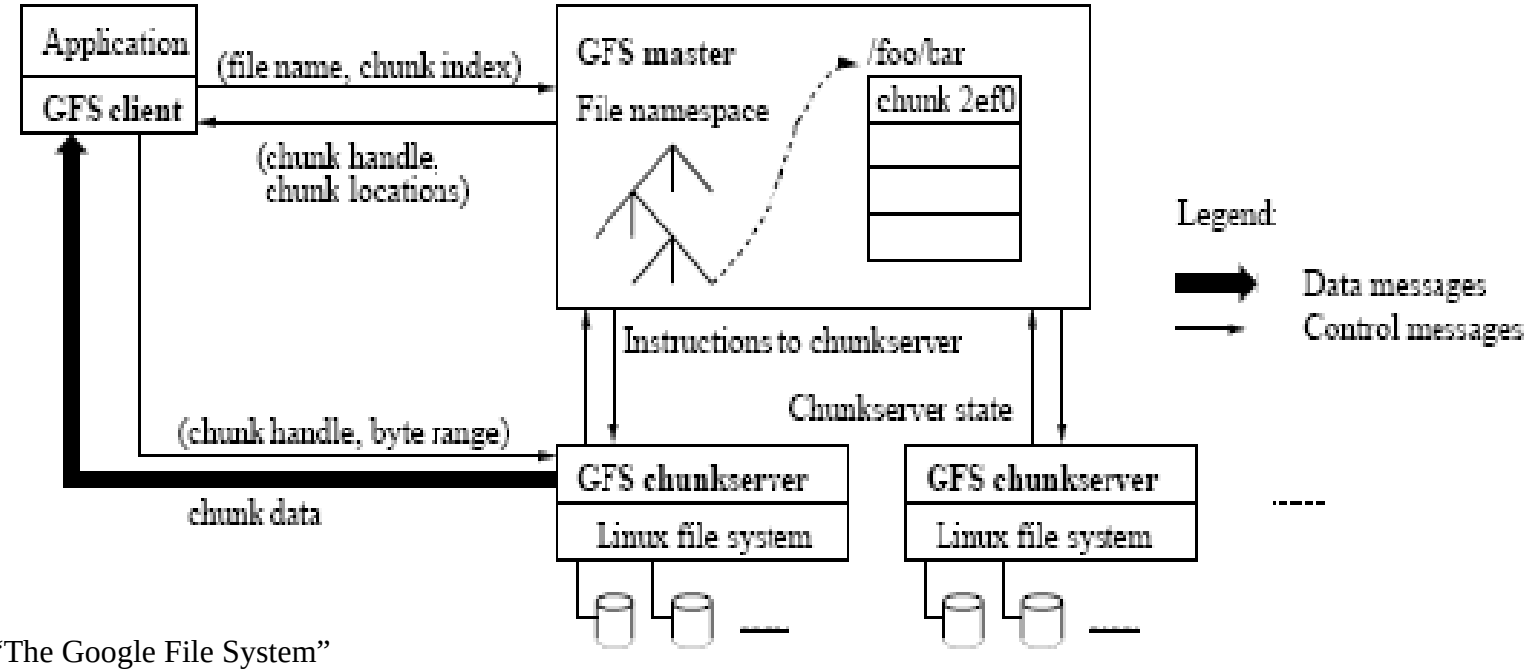
* Documenting Software Architecture: Views and Beyond, 2nd Ed. Clements et al. 2010.

Issues Addressed by Software Architecture



- Gross decomposition of a system into parts
  - often using rich abstractions for *component interaction* (or system “glue”)
  - often using common design *patterns/styles*
- Emergent system properties
  - performance, throughput, latencies
  - reliability, security, fault tolerance, evolvability
- Rationale
  - justifying architectural decisions and tradeoffs
- Envelope of allowed change
  - “load-bearing walls”

Example of Google File System

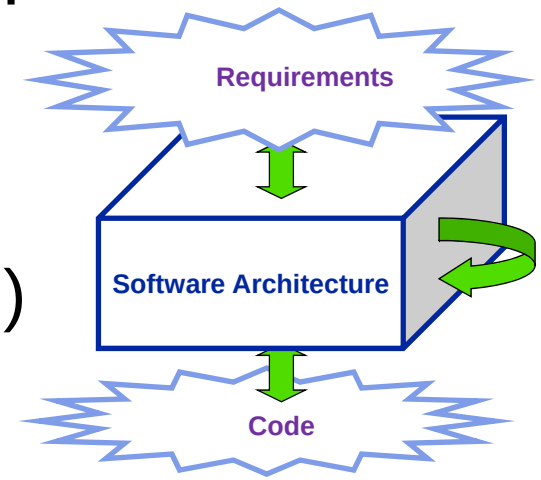


Source: "The Google File System" Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. SOSP 2003.

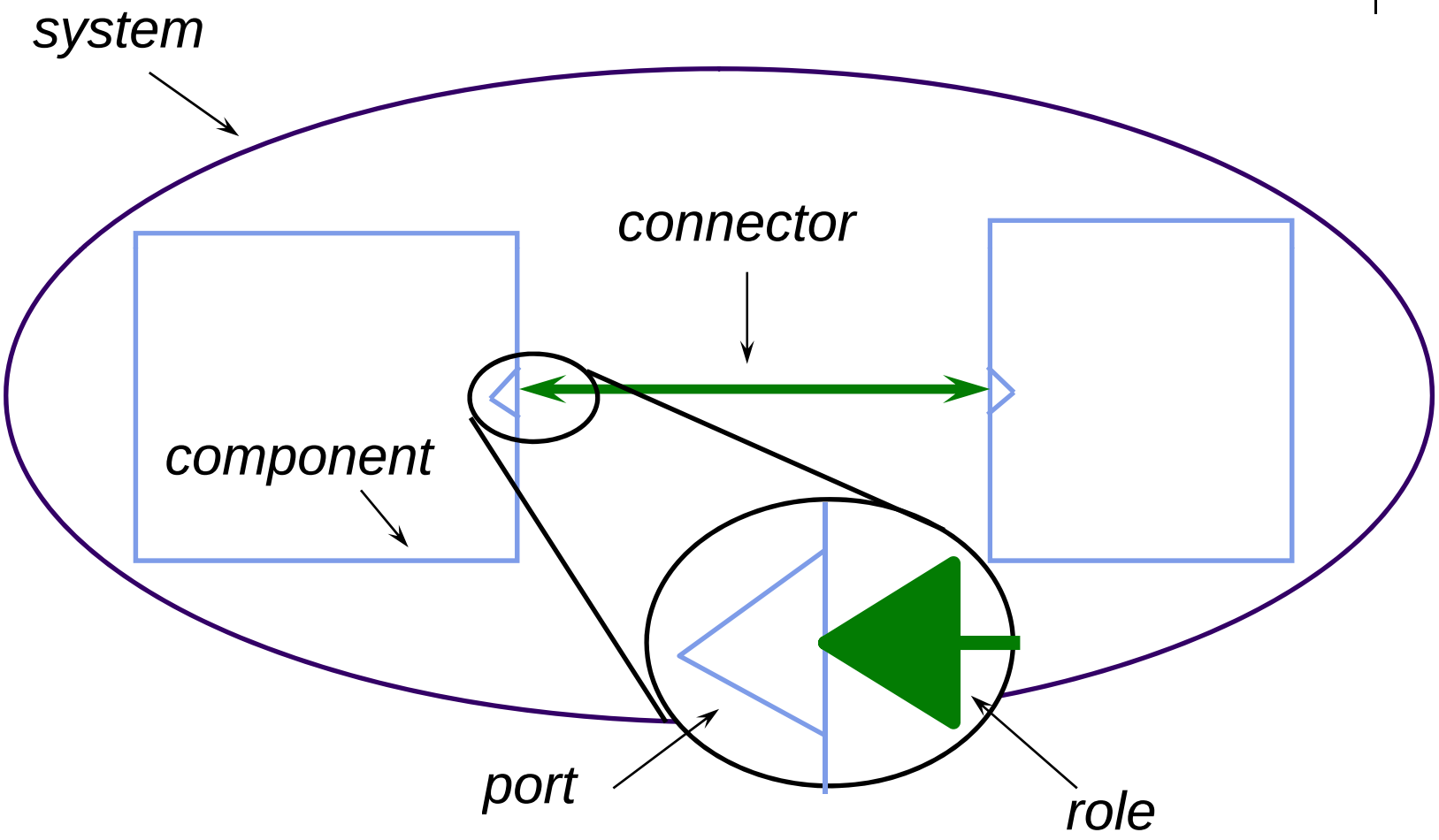
Figure 1: GFS Architecture

Architectures as Models

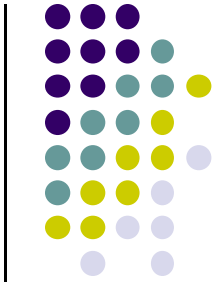
- Representations of software architecture can be treated as models
- Architecture-based design is then a form of model-based design
- Representations
  - Informal: box-and-line (ppt, visio, …)
  - Semi-formal: formal syntax (UML, SysML)
  - Formal: formal semantics (AADL, Acme, …)



Component-and-Connector Models

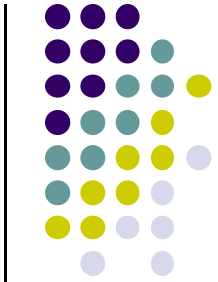


Potential Benefits



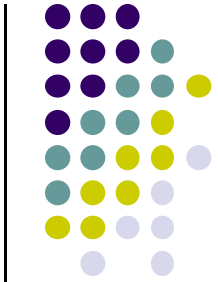
- Abstraction – manage complexity, support reuse (component, style, tactic), naturally represent computations in a given domain
- Guidance – constrain developers, support conceptual integrity
- Implementation support –tools for moving from architecture to code
- Analysis – support decision making, allowing application of existing analytical theories & tools

Outline



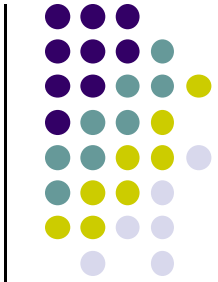
- What is Software Architecture?
  - Definition & Intent
  - Architecture Representations as Models
  - Benefits
- Basic concepts of software architecture
  - Views, Styles, and Tactics
- CPS Architectures
  - Adding physical elements to the models
  - Reconciling multiple views

Recall



The software architecture of a computing system is the set of structures needed to reason about the system, which comprise software elements, relations among them and properties of both.

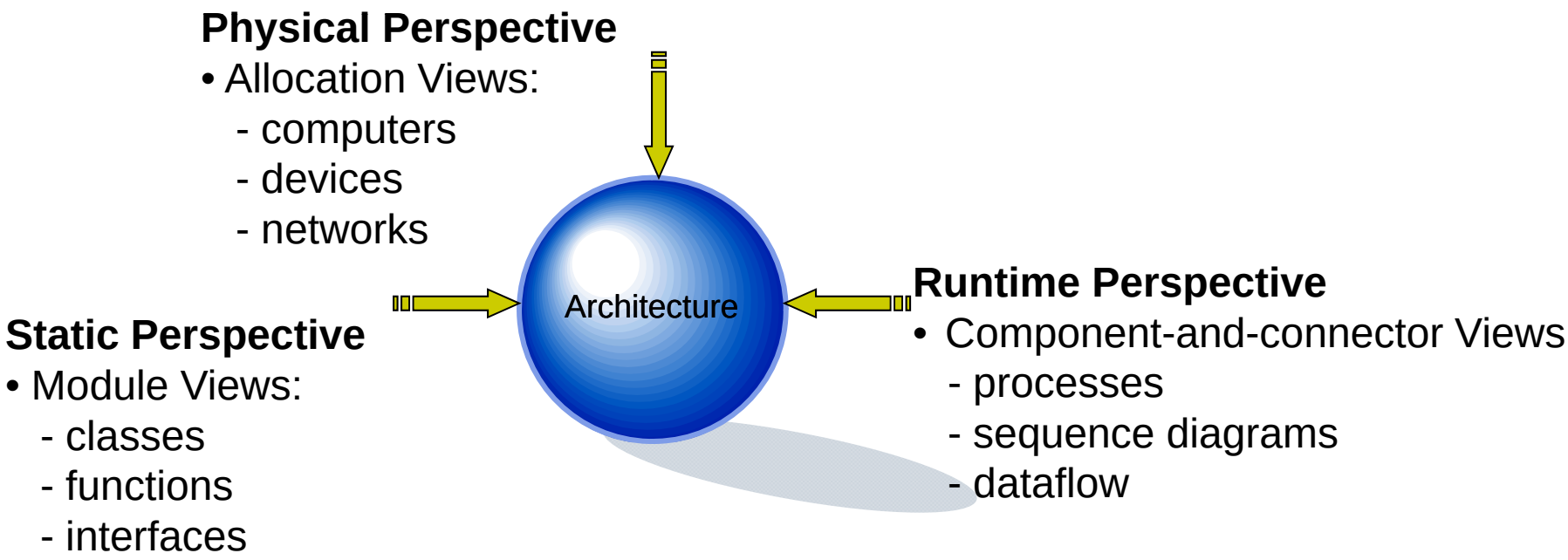
What is a Structure? – 1



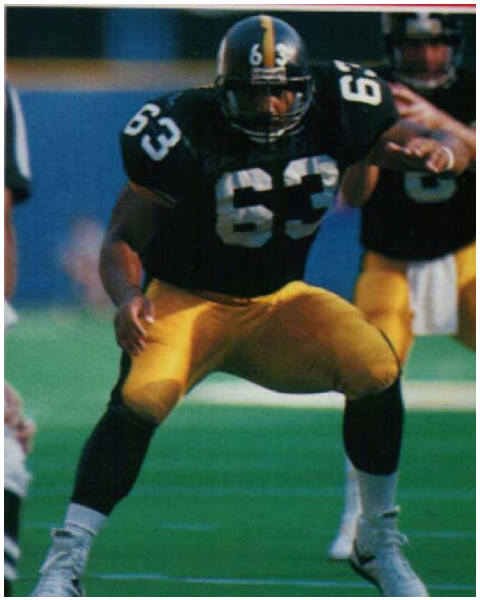
- Software architecture is an abstraction of the structures that comprise the software that is part of a software-intensive system.
- Systems have many structures
  - code
  - processes/threads
  - files

What Is A Structure? – 2

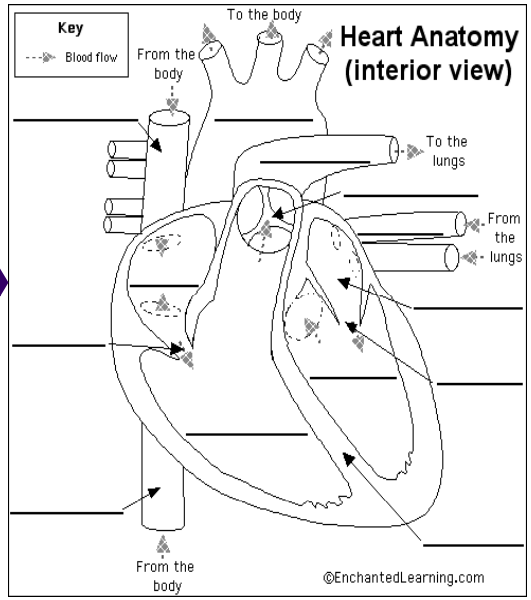
- A representation of a structure is usually called *a view* of the system.



Structures and Views – 1



A human body is comprised of multiple *structures.*

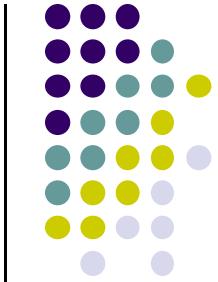


a *static view* of one human *structure*

a *dynamic view* of that *structure*

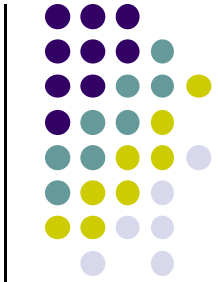
One body has many structures, and those structures have many views. So it is with software…

Three Important Classes of Architectural View

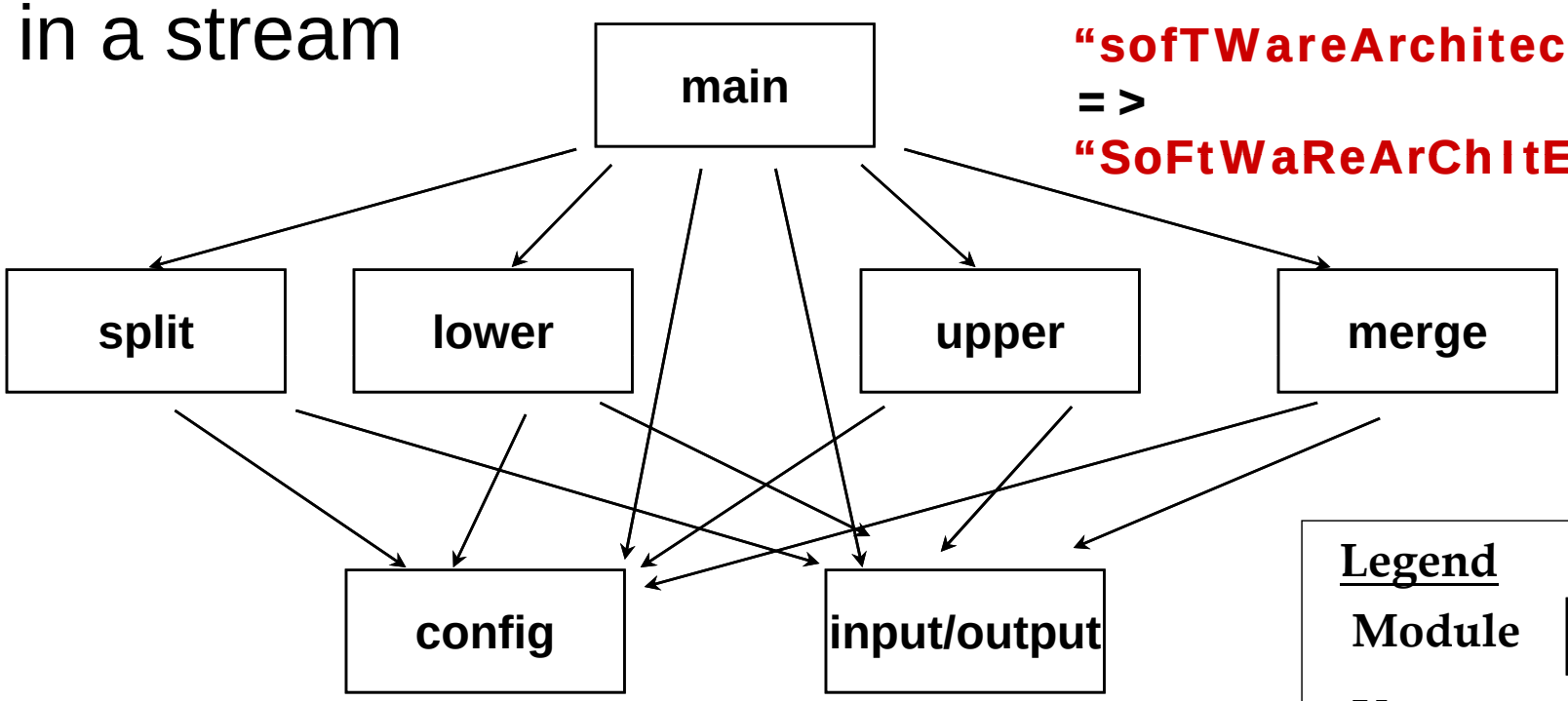


1. How it is structured as a set of code units
   module views
2. How it is structured as a set of elements that have run-time behavior and interactions
   component-and-connector views
3. How it relates to non-software structures in its environment?
   allocation views

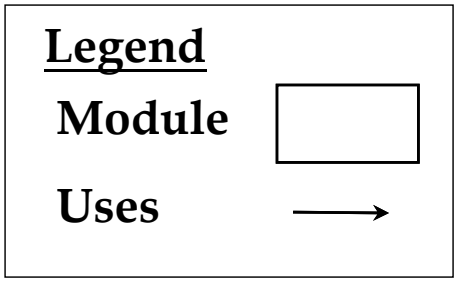
Example: Alternating Characters - Module View



Produce alternating case of characters in a stream

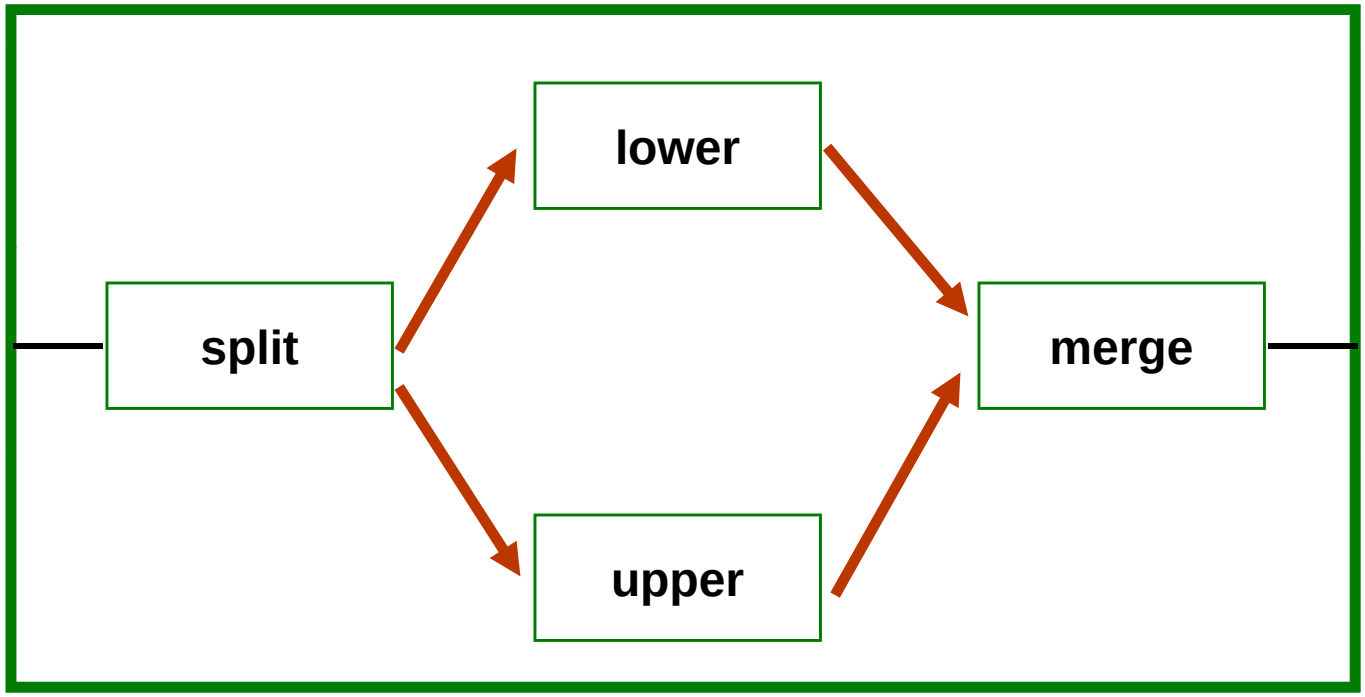


"sofTWareArchitecture"
= >
"SoFtWaReArChItEcTuRe"

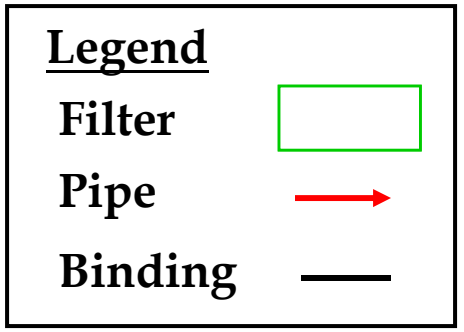


Definition/Use Modularization

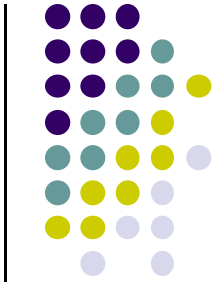
Example continued: C&C View



Components and Connectors

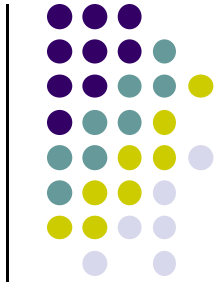


Module Views



- Elements are design-time artifacts: code, libraries, modules, packages, config files
- Common types of module views
  - Decomposition views – part-whole relations
  - Class diagrams – usage, inheritance, realization
  - Layer diagrams – restricts usage patterns

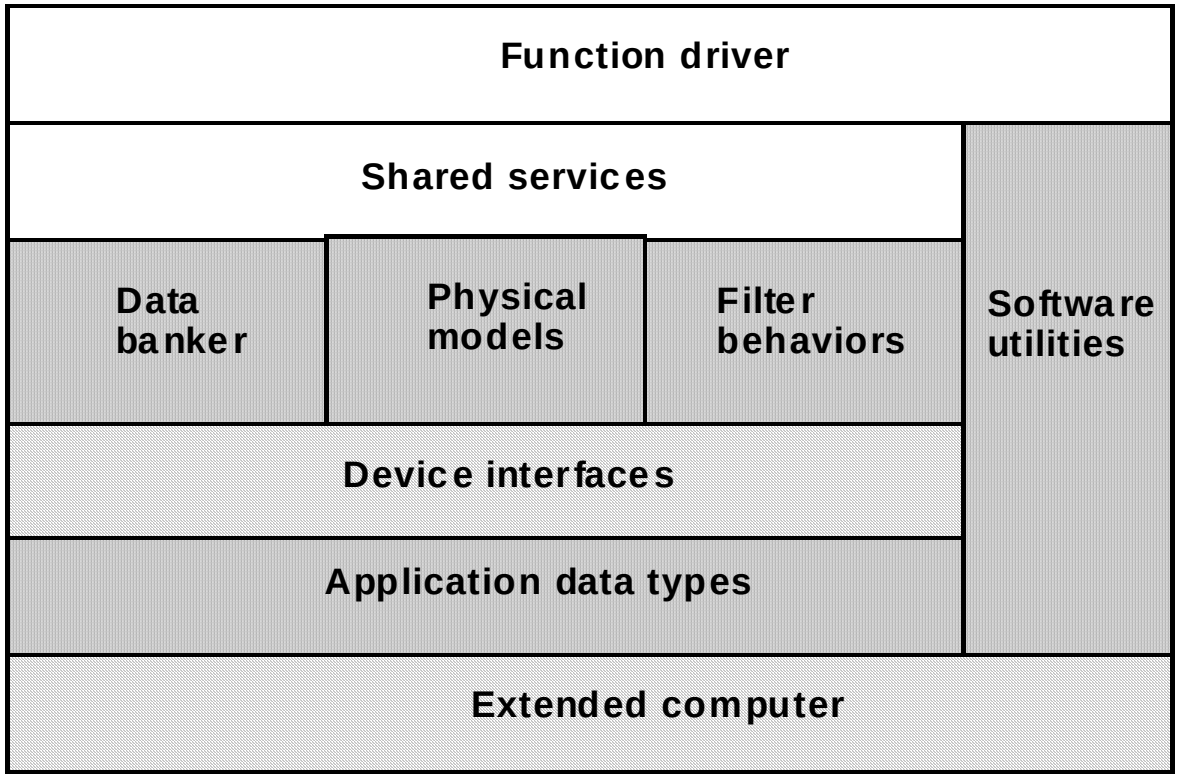
A-7E Decomposition View



- Behavior-Hiding Module
  - Function Driver Module
    - Air Data Computer Module
    - Audible Signal Module
    - Computer Fail Signal Module
    - Doppler Radar Module
    - Flight Information Display Module
    - Forward Looking Radar Module
    - Head-Up Display Module
    - Inertial Measurement Set Module
    - Panel Module
    - Projected Map Display Set Module
    - Shipboard Inertial Nav. Sys. Mod.
    - Visual Indicator Module
    - Weapon Release Module
    - Ground Test Module
  - Shared Services Module
    - Mode Determination Module
    - Panel I/O Support Module
    - Shared Subroutine Module
    - Stage Director Module
    - System Value Module

- Software Decision Module
  - Application Data Type Module
    - Numeric Data Type Module
    - State Transition Event Mod.
  - Data Banker Module
    - Singular Values Module
    - Complex Event Module
  - Filter Behavior Module
  - Physical Models Module
    - Aircraft Motion Module
    - Earth Characteristics Module
    - Human Factors Module
    - Target Behavior Module
    - Weapon Behavior Module
  - Software Utility Module
    - Power-Up Initialization Module
    - Numerical Algorithms Module
  - System Generation Module
    - System Generation Parameter Mod.
    - Support Software Module

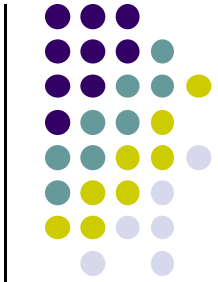
A-7E Layered View



Key:

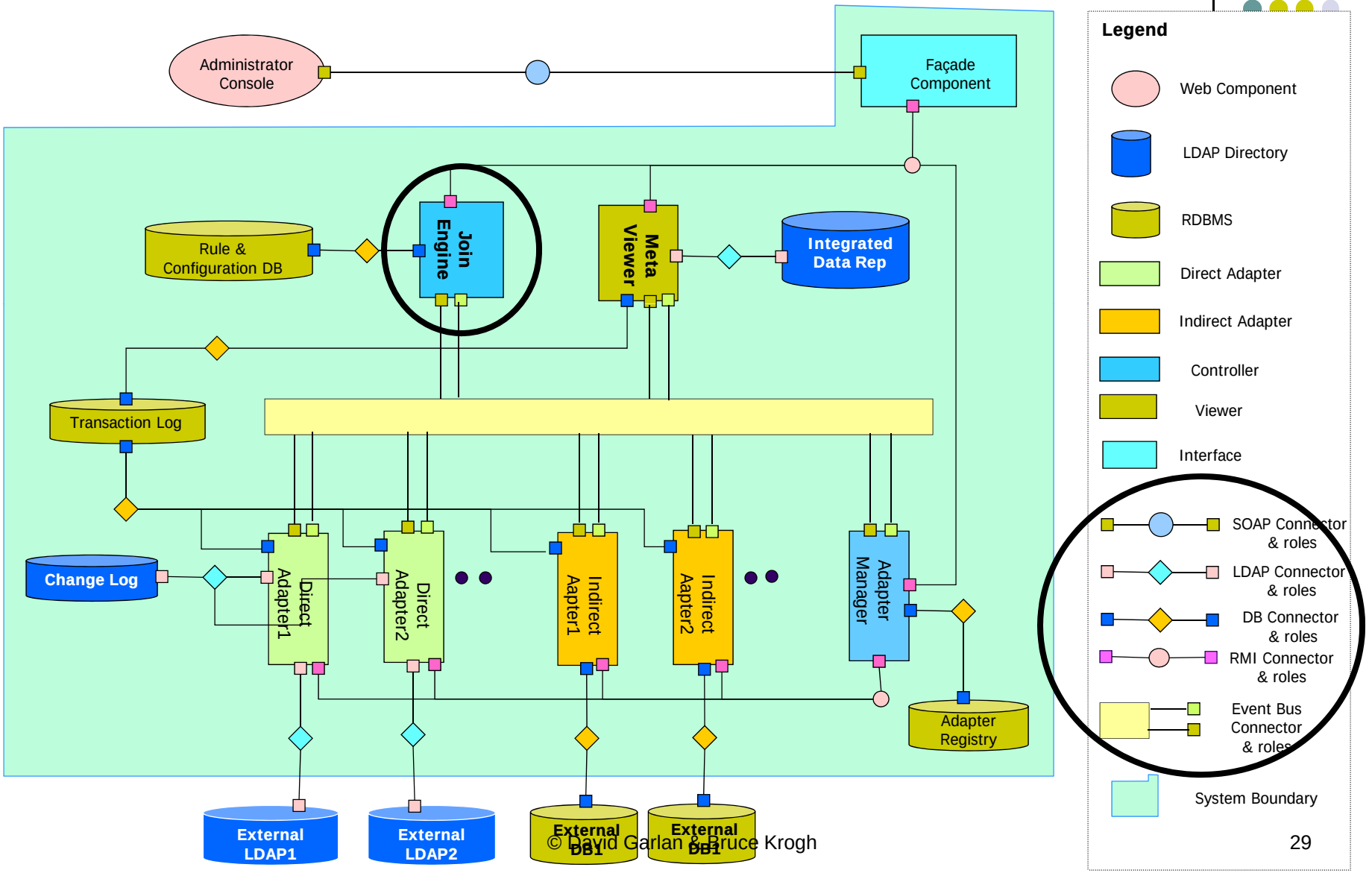
- Behavior-hiding module
- Software decision hiding module
- Hardware hiding module

Component & Connector Views

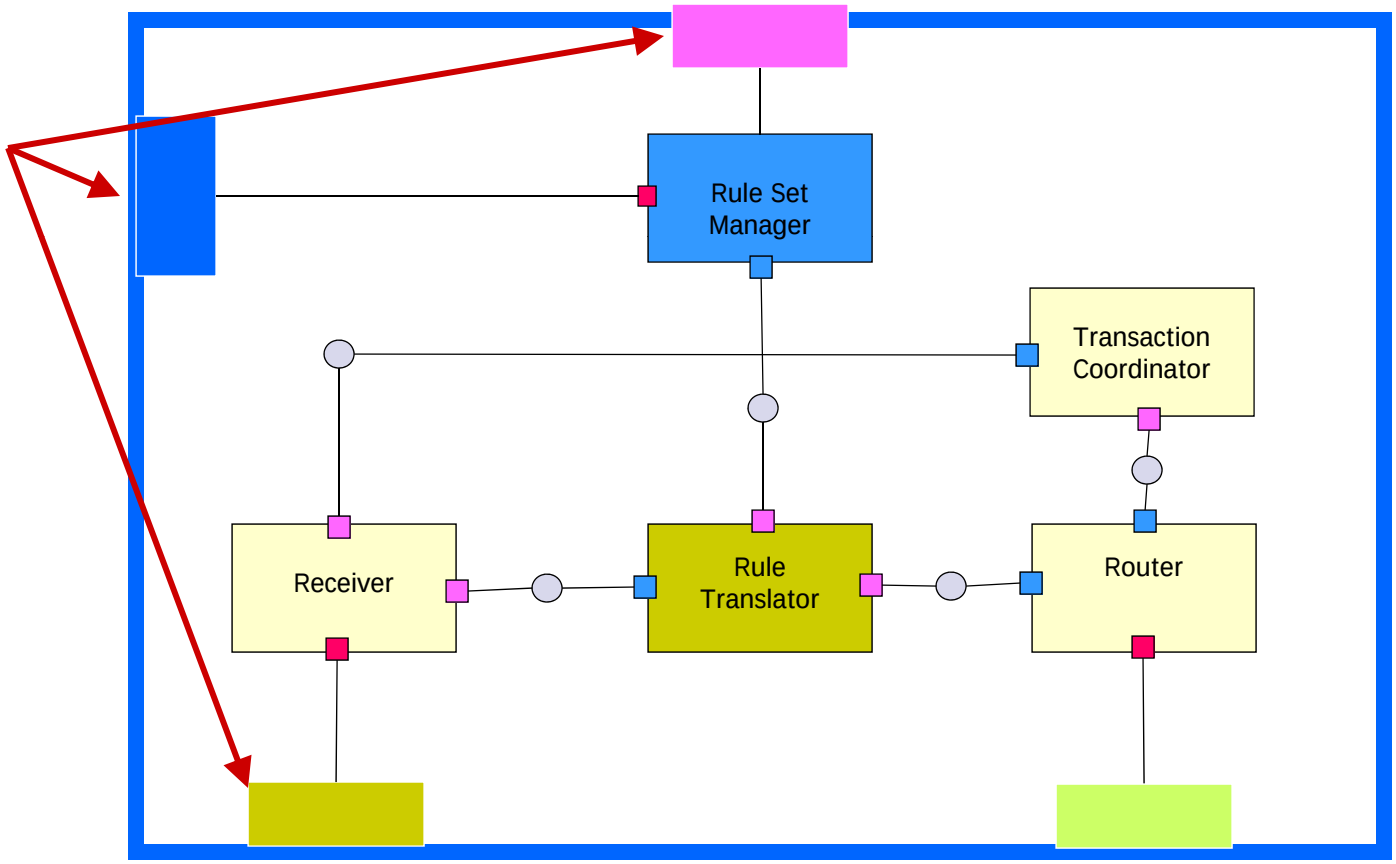


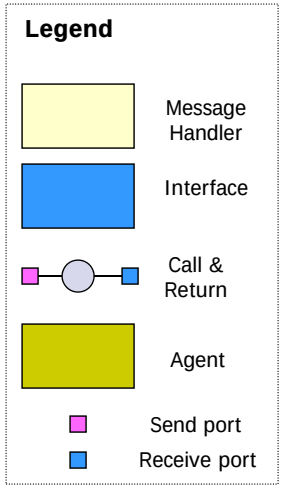
- Elements are components and connectors
- May be hierarchical
- Annotations provide semantics

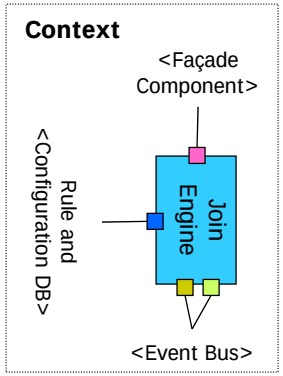
Eample C&C View



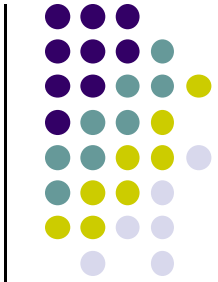
Elaboration of Join Engine





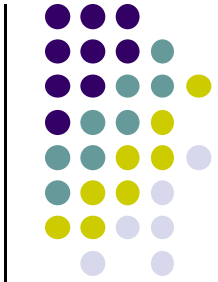


Styles



- Define a specialized vocabulary for a kind of view
  - Pipes and filters
  - Clients and servers
  - Publishers and subscribers
- Establish constraints
  - Clients can’t talk to each other directly
  - No cycles in a pipe-and-filter system
- Provide analysis opportunities

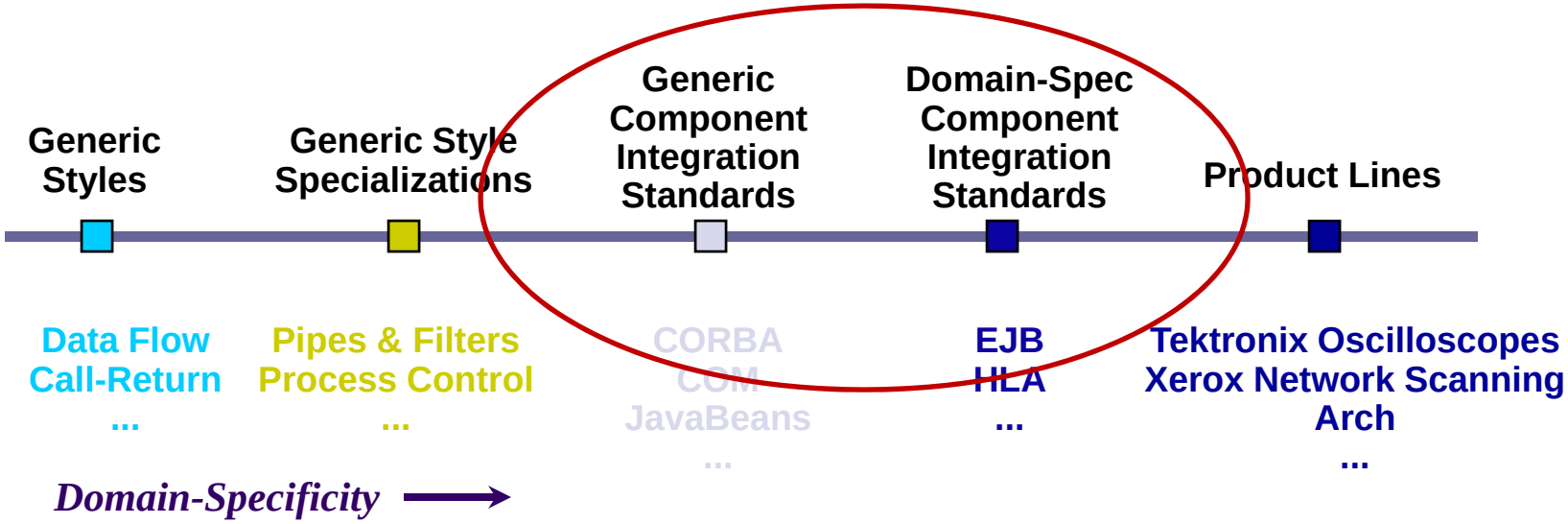
A (Partial) Catalogue of Styles



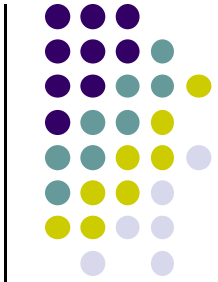
- Data flow
  - batch sequential
  - pipes and filters
  - process control
- Call-return
  - main program-subroutine
  - object-oriented
  - component-based
  - peer-to-peer
  - service-oriented
  - N-tiered
- Event-based
  - asynchronous messaging
  - publish-subscribe
  - implicit invocation
  - data-triggered
- Data-centered
  - repository
  - blackboard
  - shared variable

Specialized Architectural Styles

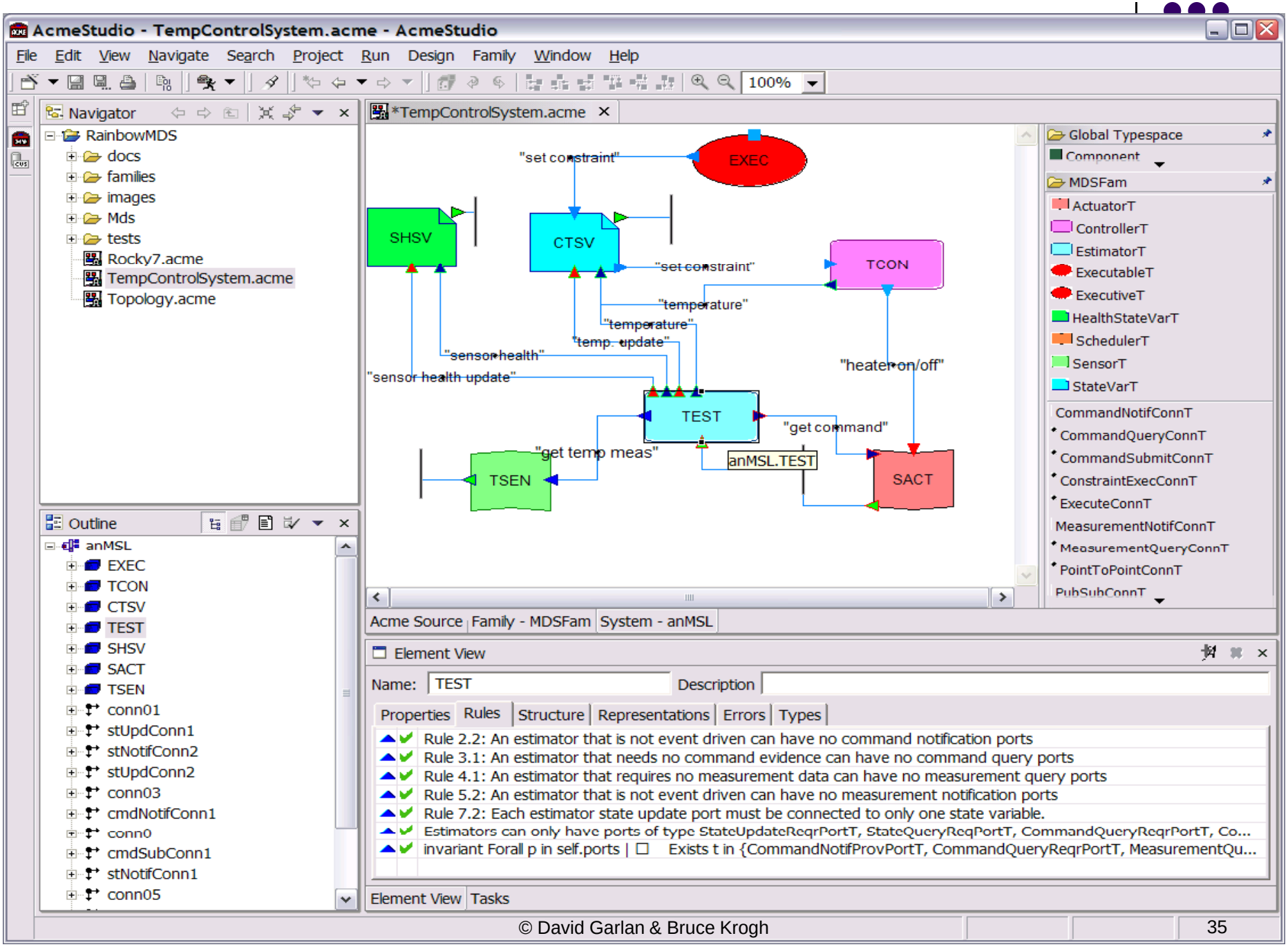
- There is a spectrum of architectural styles
  - Some are generic; others are more domain-specific and specialized.
  - Specialization supports analysis, code reuse, tools



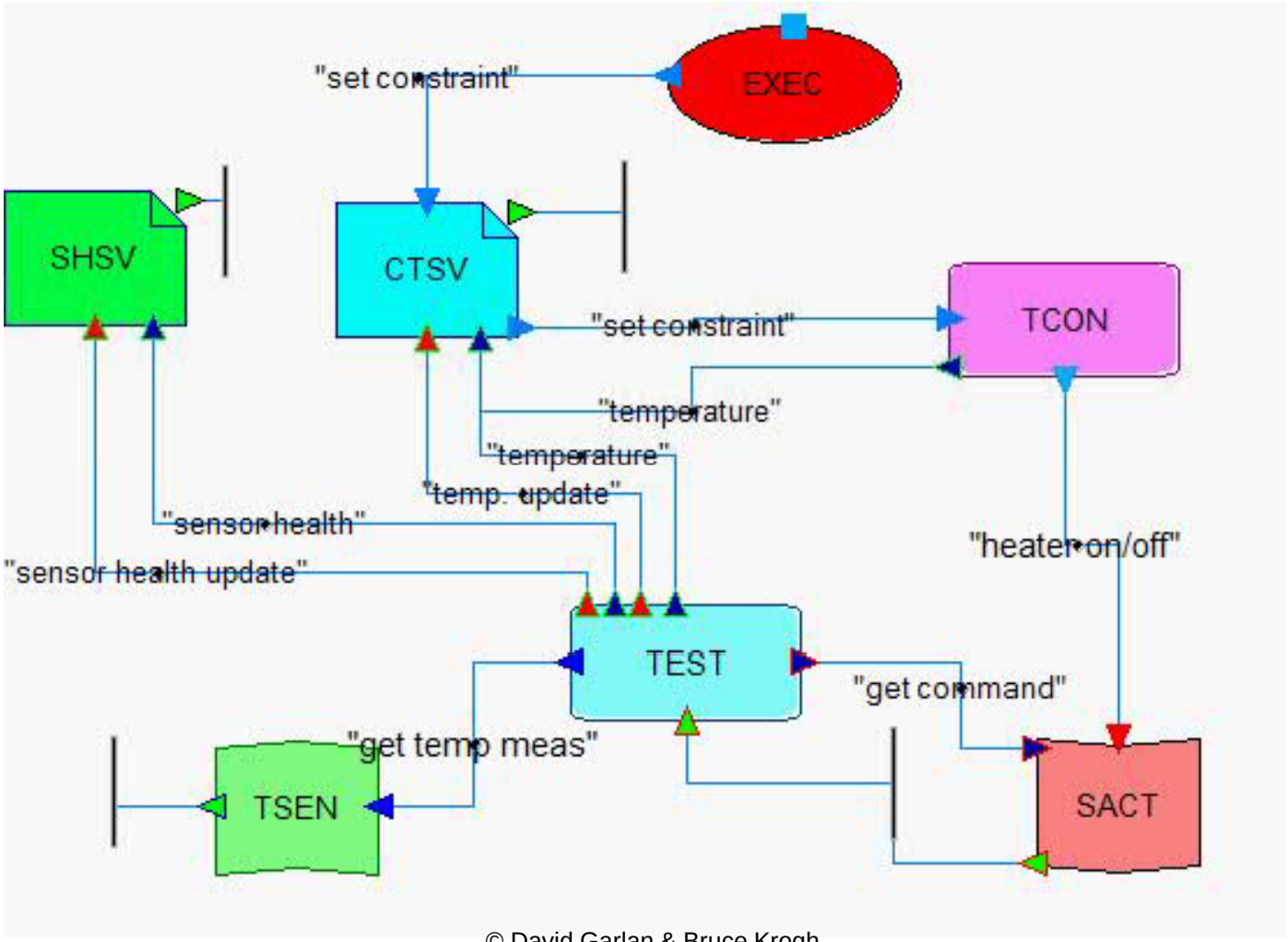
Example: NASA Mission Data Systems (MDS)



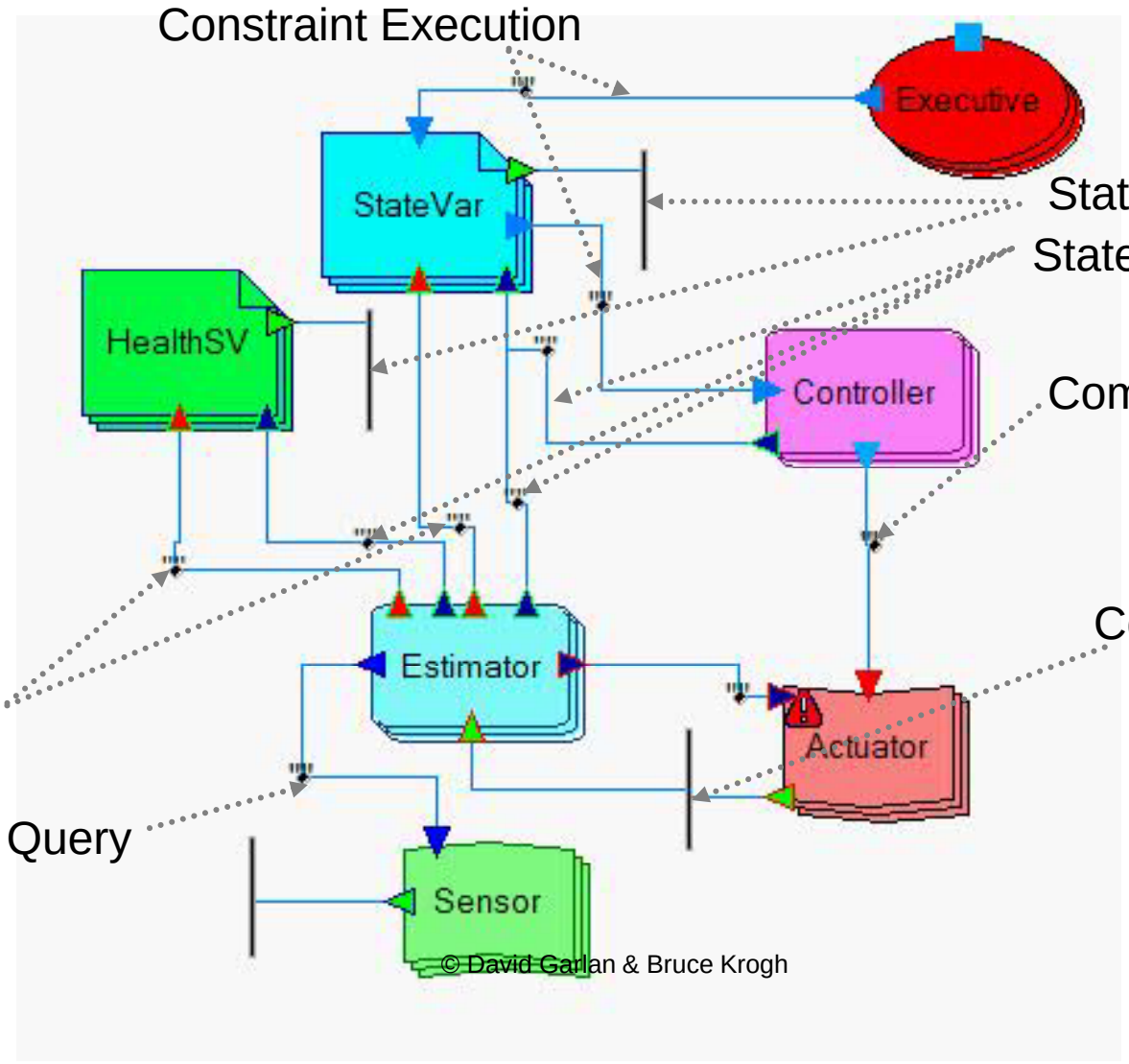
- MDS defines an architectural framework for a family of NASA space systems
  - System of architectural component types
  - Rules on how they can be connected
  - Run-time infrastructure for executing MDS systems
  - Reusable code base
- Checking/ensuring conformance to MDS is an important and hard problem
  - Many rules, many components, complex topology
  - Mapping between architectural design and code is non-trivial


AcmeStudio - TempControlSystem.acme - AcmeStudio
File Edit View Navigate Search Project Run Design Family Window Help
Navigator
RainbowMDS
docs
families
images
Mds
tests
Rocky7.acme
TempControlSystem.acme
Topology.acme
*TempControlSystem.acme
"set constraint"
EXEC
SHSV
CTSV
"set constraint"
TCON
"temperature"
"temperature"
"temp. update"
"sensor health"
"sensor health update"
"heater on/off"
TEST
"get command"
"get temp meas"
anMSL.TEST
TSEN
SACT
Global Typespace
Component
MDSFam
ActuatorT
ControllerT
EstimatorT
ExecutableT
ExecutiveT
HealthStateVarT
SchedulerT
SensorT
StateVarT
CommandNotifConnT
CommandQueryConnT
CommandSubmitConnT
ConstraintExecConnT
ExecuteConnT
MeasurementNotifConnT
MeasurementQueryConnT
PointToPointConnT
PubSubConnT
Acme Source
Family - MDSFam
System - anMSL
Outline
anMSL
EXEC
TCON
CTSV
TEST
SHSV
SACT
TSEN
conn01
stUpdConn1
stNotifConn2
stUpdConn2
conn03
cmdNotifConn1
conn0
cmdSubConn1
stNotifConn1
conn05
Element View
Name: TEST
Description
Properties
Rules
Structure
Representations
Errors
Types
Rule 2.2: An estimator that is not event driven can have no command notification ports
Rule 3.1: An estimator that needs no command evidence can have no command query ports
Rule 4.1: An estimator that requires no measurement data can have no measurement query ports
Rule 5.2: An estimator that is not event driven can have no measurement notification ports
Rule 7.2: Each estimator state update port must be connected to only one state variable.
Estimators can only have ports of type StateUpdateReqrPortT, StateQueryReqPortT, CommandQueryReqrPortT, Co...
invariant Forall p in self.ports | ☐ Exists t in {CommandNotifProvPortT, CommandQueryReqrPortT, MeasurementQu...
Element View
Tasks

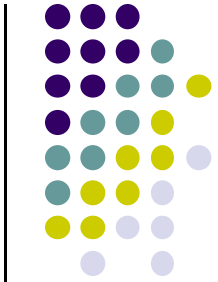
Temperature Control System



The MDS Style

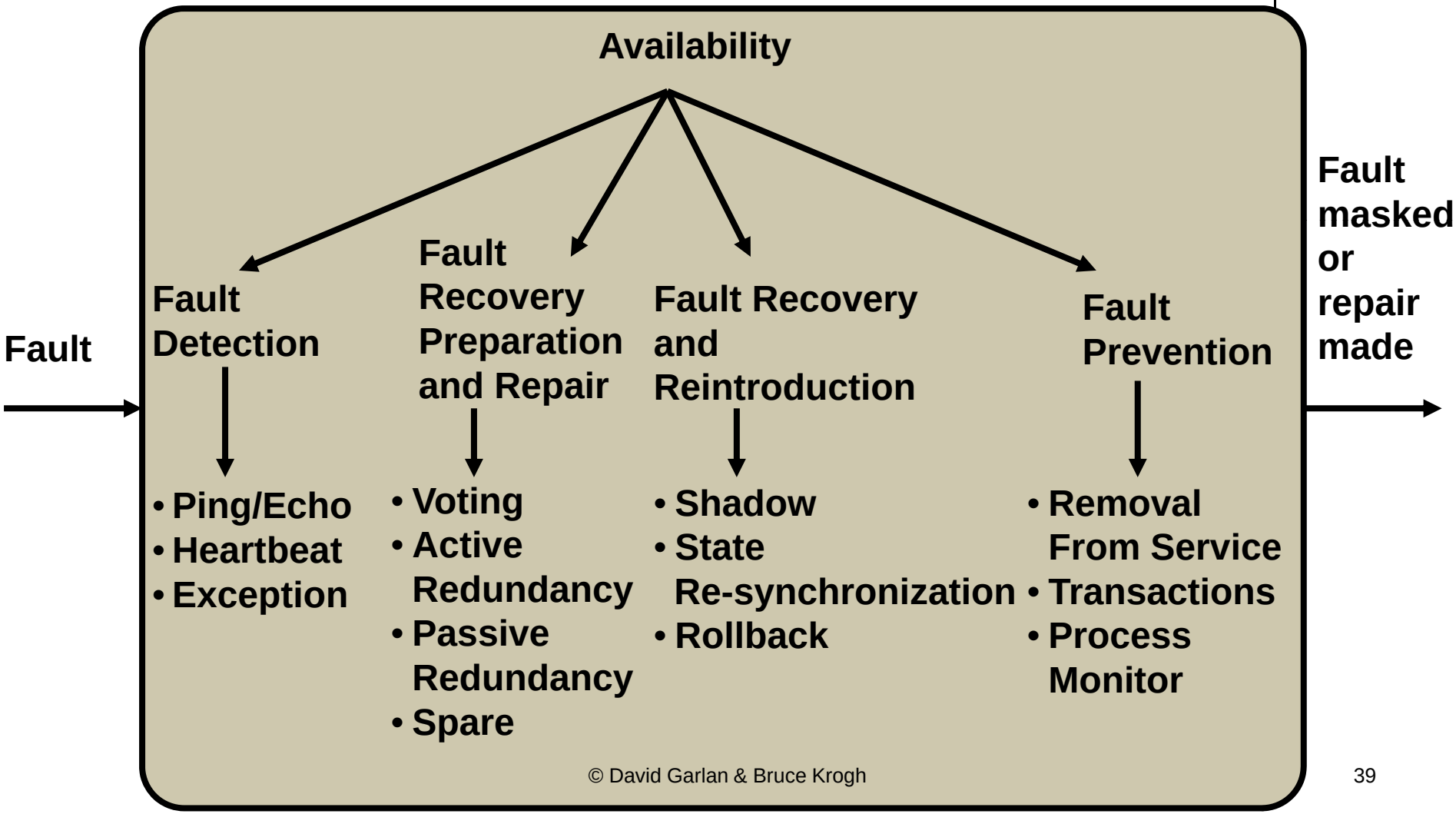


Tactics

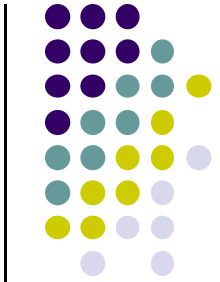


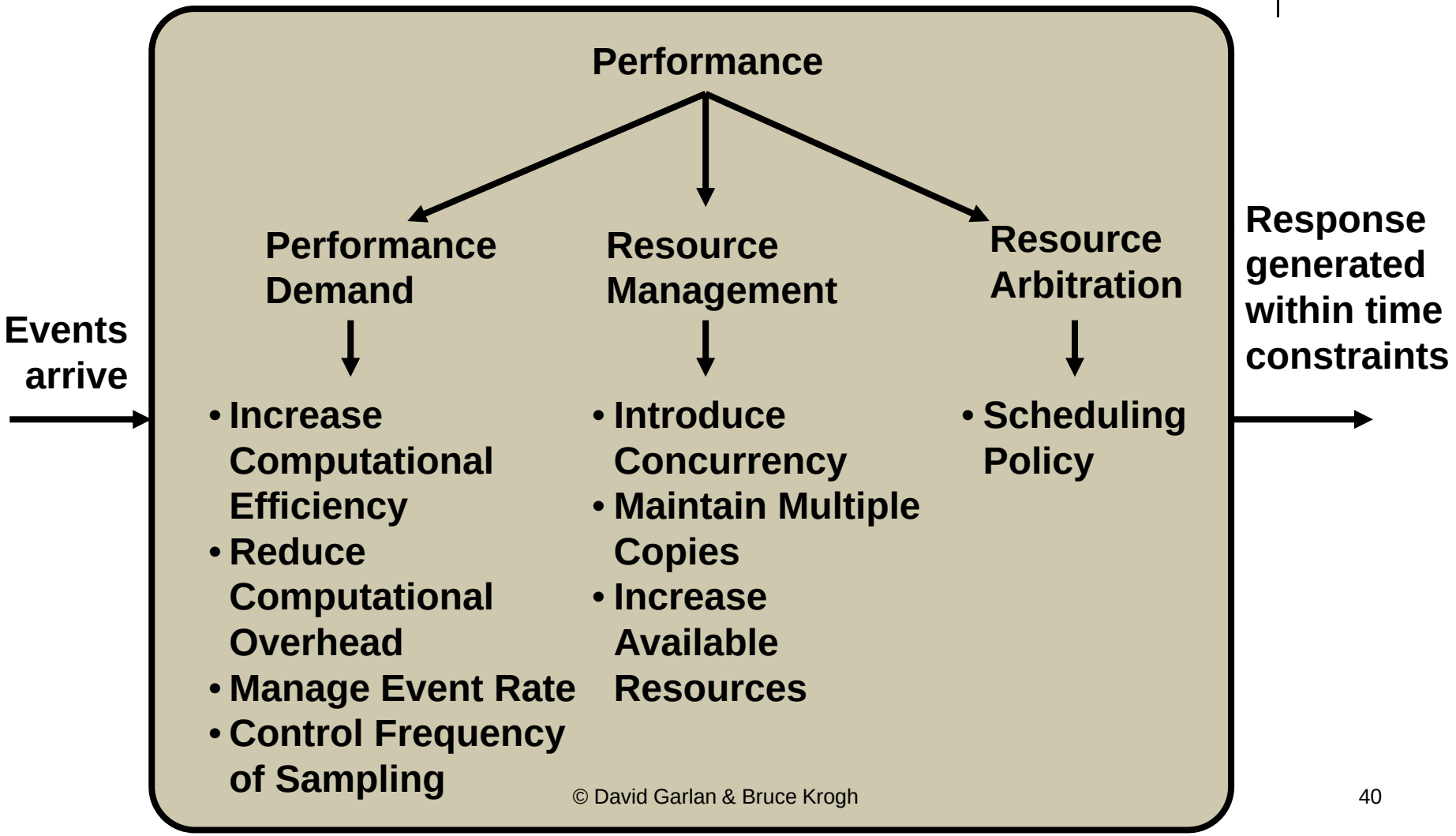
- A *tactic* is a design decision that refines a high level style/pattern and is influential in the control of a quality attribute response.
- Tactics complement and refine patterns that make up the architecture.

Availability Tactics

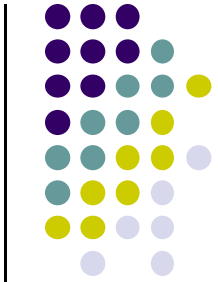


Performance Tactics



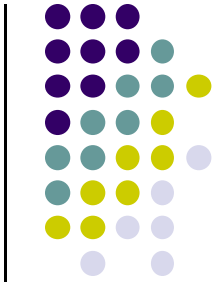


Outline

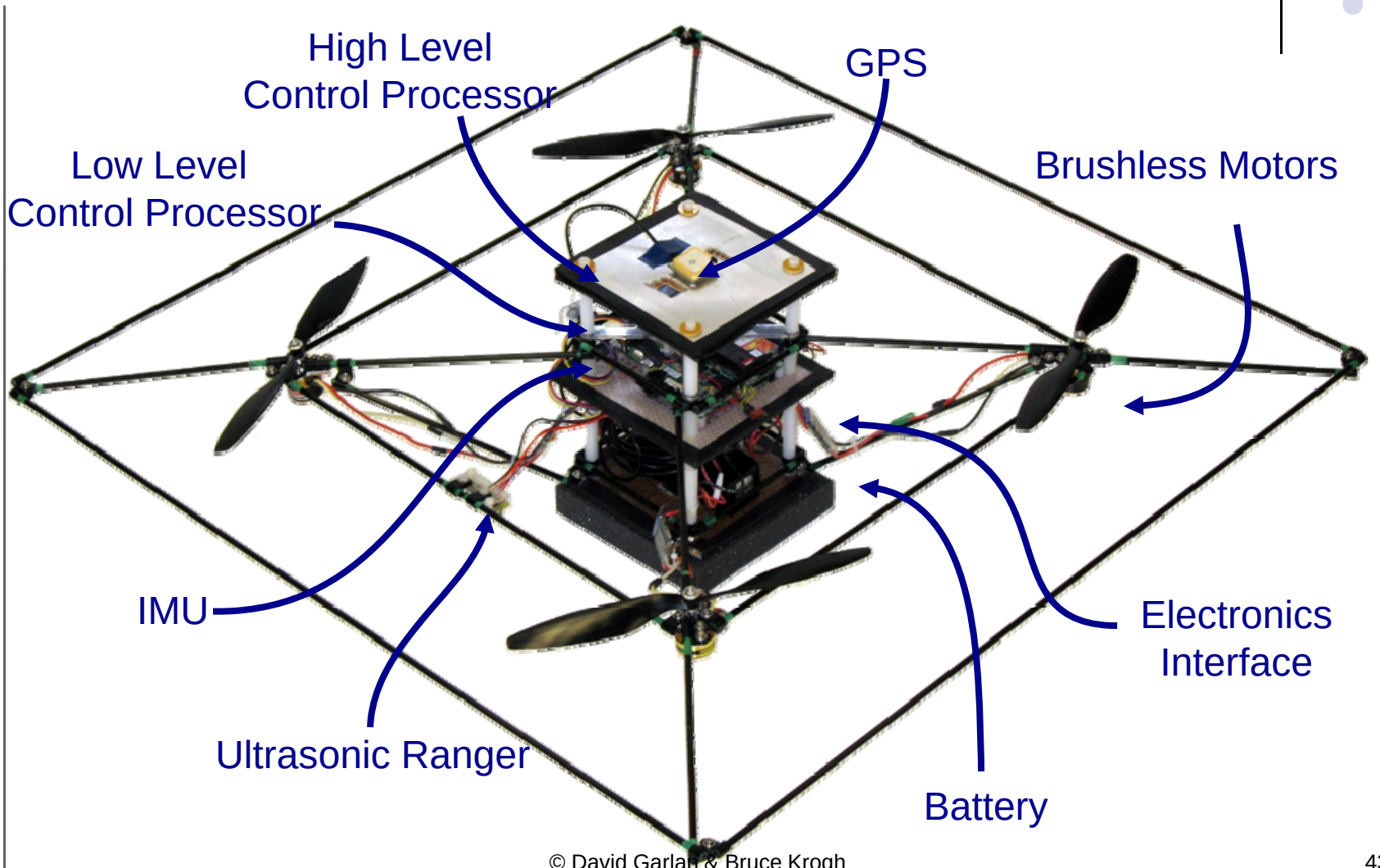


- What is Software Architecture?
  - Definition & Intent
  - Architecture Representations as Models
  - Benefits
- Basic concepts of software architecture
  - Views, Styles, and Tactics
- CPS Architectures
  - Adding physical elements to the models
  - Reconciling multiple views

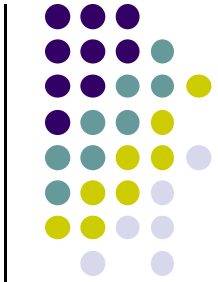
Extending Architecture to CPS



- Three main problems
  - Extending architecture models to support both cyber and physical elements (and their interactions)
  - Incorporating existing modeling techniques
  - Reconciling multiple views

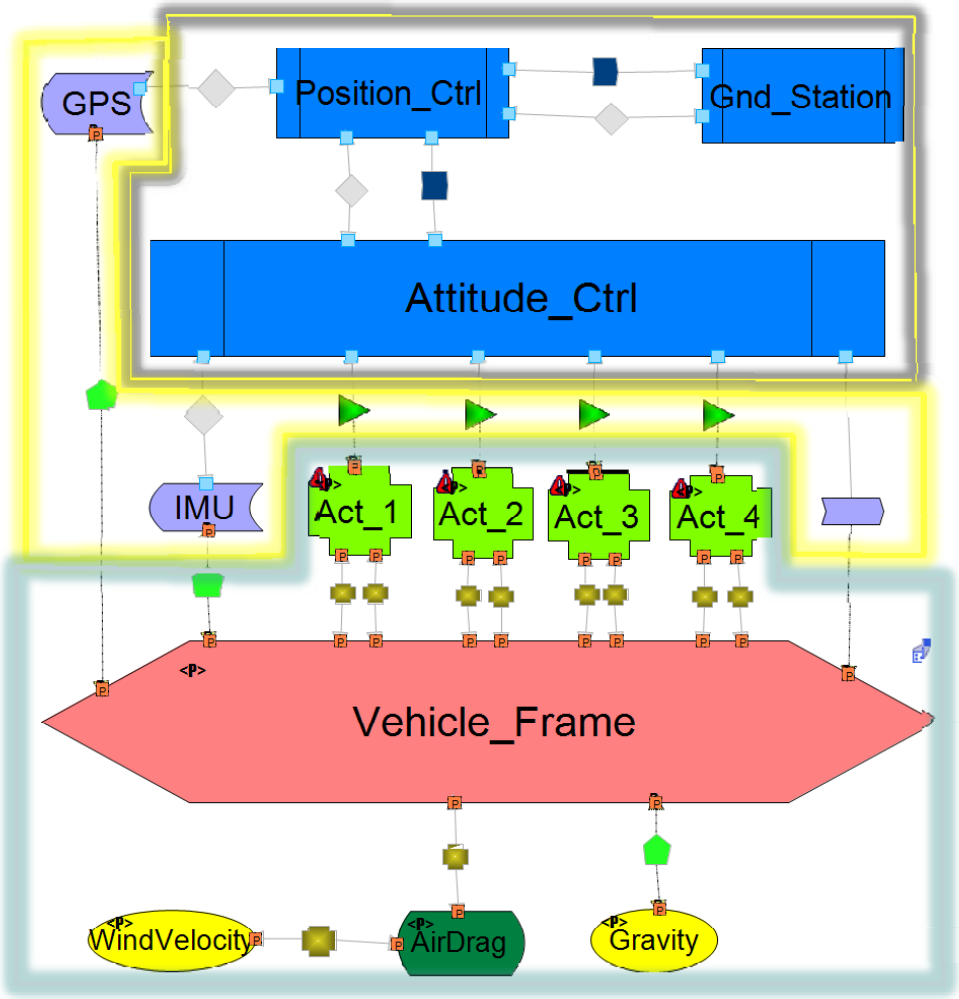


Extended with Physical Elements



- Include physical system as a set of interacting components with shared variables/coupled constraints
  - Components: Physical elements (mechanical, electrical, thermal, environmental,…)
  - Connectors: Physical interactions (conservation laws, energy flows, …)
  - Behavior: Dynamic behavior of elements (DAEs, LHA, …)
- Bridging elements link physical elements to cyber elements

Quadrotor Architectural Model

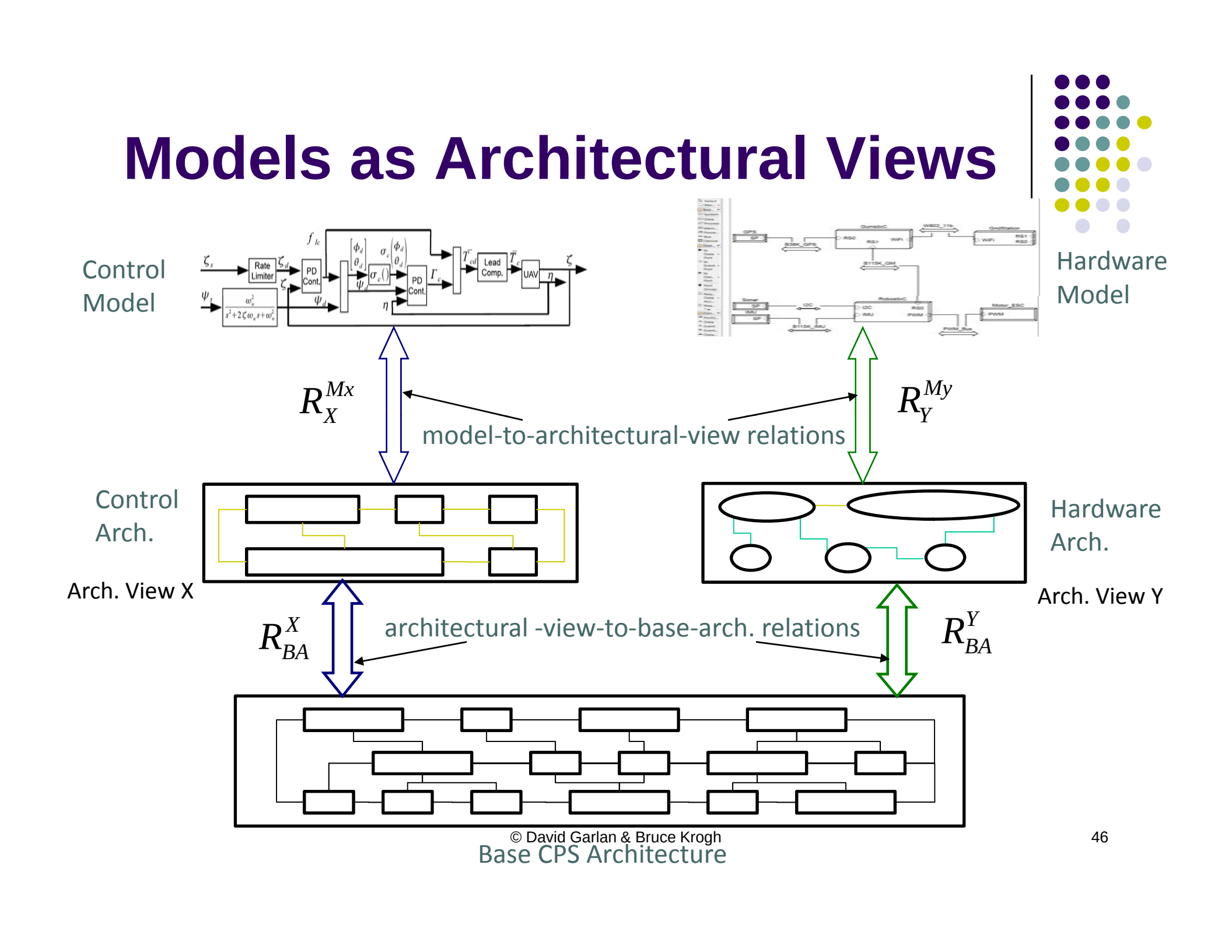


Cyber elements

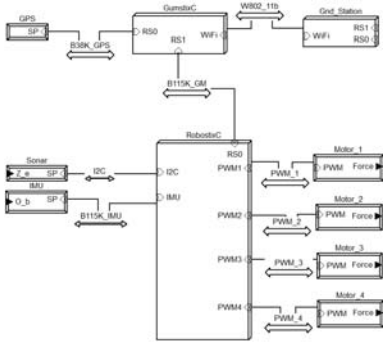
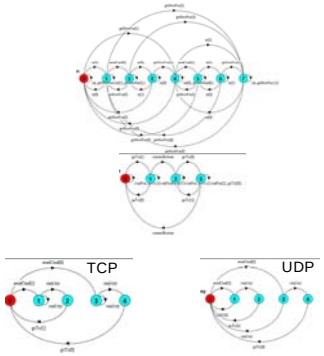
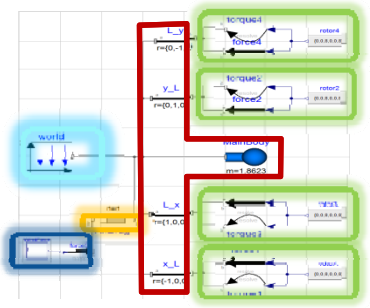
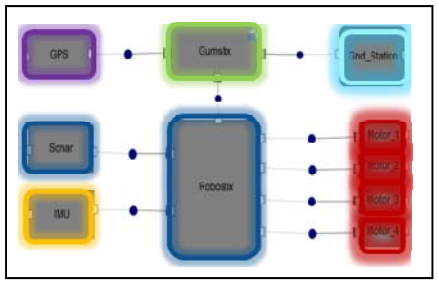
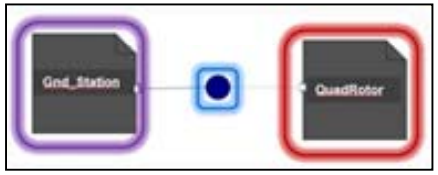
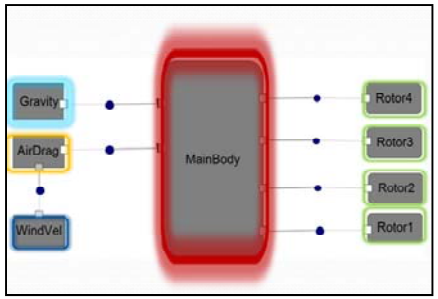
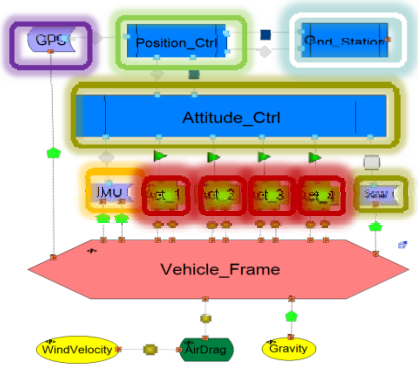
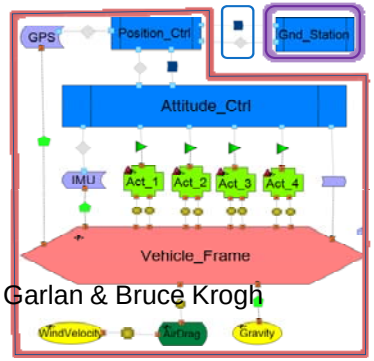
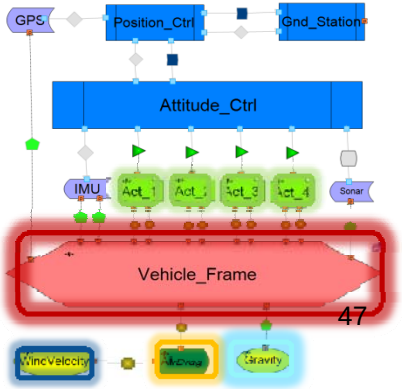
Bridging elements

Physical elements

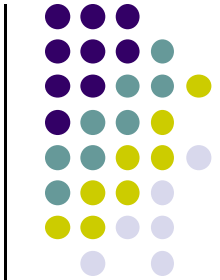
Models as Architectural Views



STARMAC Architectural Views

	Hardware (AADL)	Software (FSP)	Physical (Modelica)
Model			
Arch. View			
Base Arch.			

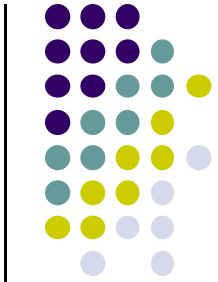
References - 1



General Architecture

- Shaw, M.; Garlan, D. *Software Architecture: Perspectives on an Emerging Discipline*, Upper Saddle River, NJ: Prentice Hall, 1996
- Bass, L.; Clements, P. & Kazman, R. *Software Architecture in Practice, Second Edition.* Boston, MA: Addison-Wesley, 2003.
- Lattanze, A. *Architecting Software Intensive Systems: A Practitioners Guide*, New York, NY: Taylor and Francis, 2008
- Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Stal, M. *Pattern Oriented Software Architecture: A System of Patterns.* West Sussex England: John Wiley Ltd., 1996.
- Clements, P. et al., *Documenting Software Architecture: Views and Beyond, Second Edition,* Addison Wesley, 2011.

References - 2



Conferences

- Working International Conference on Software Architecture (WICSA)
- European Conference on Software Architecture (ECSA)

Software Architecture at CMU

- http://www.cs.cmu.edu/~able then click on publications

CPS Architecture Research at CMU

- http://www.cs.cmu.edu/~able/research/scyphys.html