A Case Study in Product Line Architectures

Dewayne E Perry
ENS 623A
Office Hours: T/Th 11:00-12:00
perry @ ece.utexas.edu
www.ece.utexas.edu/~perry/education/382v-s06/

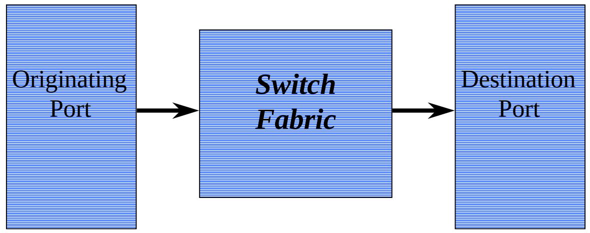
Context

- A snapshot during the architectural process for this product line (ie, not THE final product line architecture)
- Basic requirements
  - Cover a large class of diverse instances in the same application domain
  - Support dynamic reconfiguration
- Simplification of non-relevant issues
- Product Line Domain
  - Network Communication Product
  - Real time, embedded system
  - HW event driven
  - High reliability, high integrity
  - Fault-tolerant, fault-recoverable
  - Hardened - to operate in a variety of environments

Access Boxes

- Current State
  - Custom built to customer specification
  - Hard-wired hardware
  - Hard/hand-coded software
  - To evolve: build new hardware and software
- Target State of the Product Line
  - Dynamic reconfiguration of both HW and SW
  - Hardware
    - common interfaces
    - plug compatible components
  - Software
    - generic architecture
    - common platform
    - plug and play

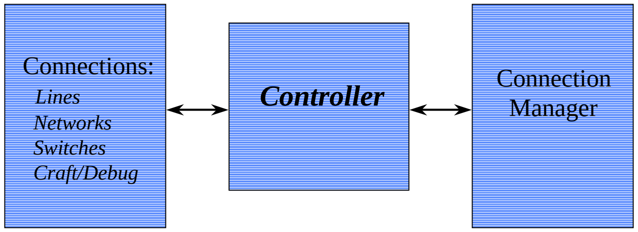
Basic Abstraction: *Connection*



Basic Abstraction: Connections

- Variety of connections from
  - relatively static to
  - dynamic, simple to complex
- Variety of connection machines from
  - simple one board, centralized systems to
  - multiple board, distributed systems

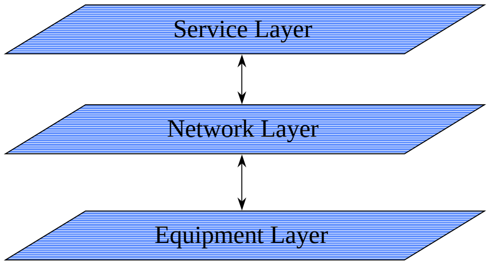
Basic HW/SW System

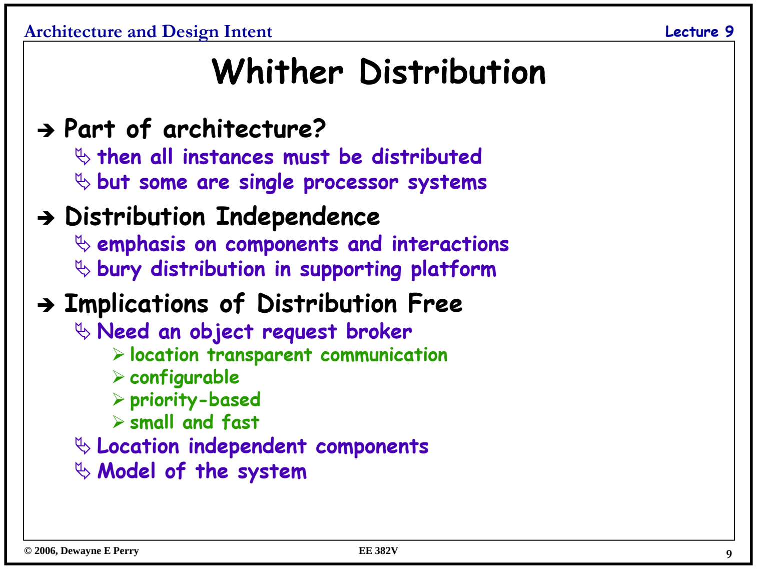


Basic System

- Devices of various sorts that are used for connections to various kinds of network components
- Controllers for those devices
- A connection manage to establish and remove connections

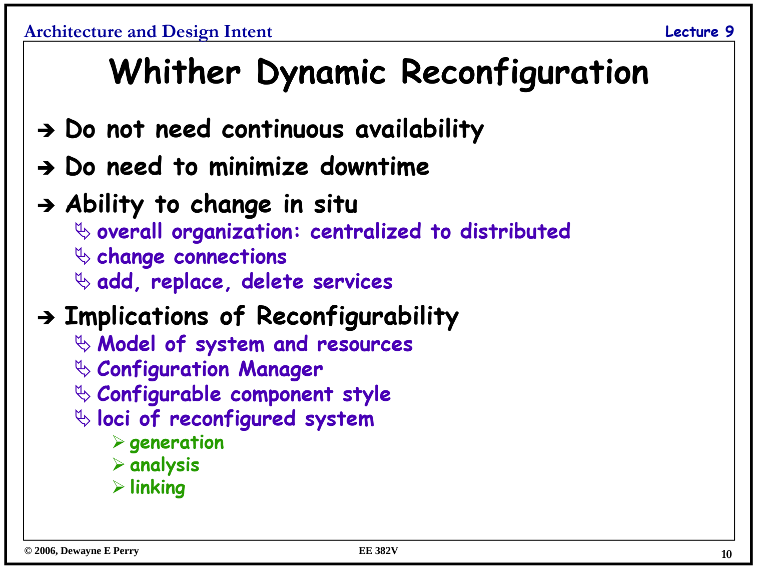
Typical Architecture





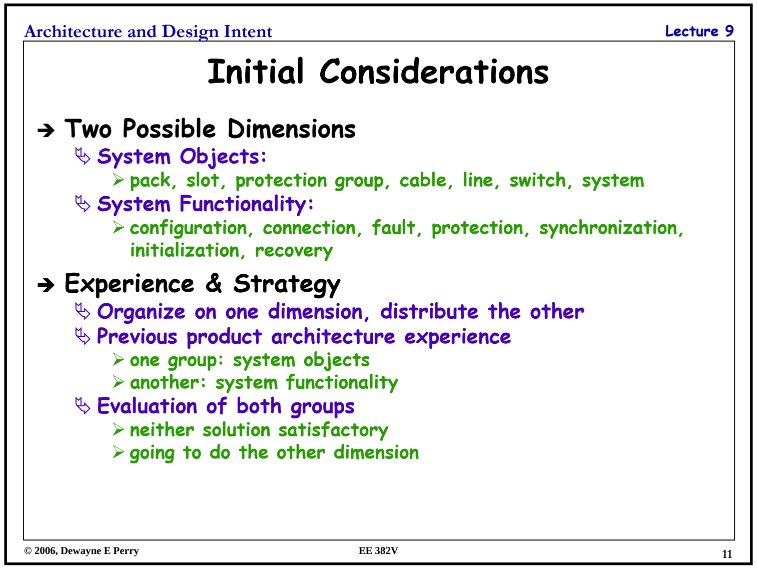
Whither Distribution

- Part of architecture?
  - then all instances must be distributed
  - but some are single processor systems
- Distribution Independence
  - emphasis on components and interactions
  - bury distribution in supporting platform
- Implications of Distribution Free
  - Need an object request broker
    - location transparent communication
    - configurable
    - priority-based
    - small and fast
  - Location independent components
  - Model of the system



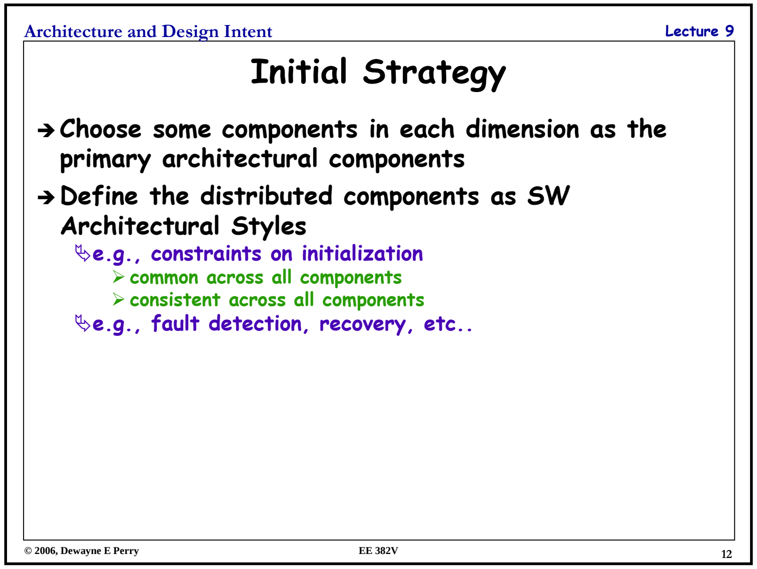
Whither Dynamic Reconfiguration

- Do not need continuous availability
- Do need to minimize downtime
- Ability to change in situ
  - overall organization: centralized to distributed
  - change connections
  - add, replace, delete services
- Implications of Reconfigurability
  - Model of system and resources
  - Configuration Manager
  - Configurable component style
  - loci of reconfigured system
    - generation
    - analysis
    - linking



Initial Considerations

- Two Possible Dimensions
  - System Objects:
    - pack, slot, protection group, cable, line, switch, system
  - System Functionality:
    - configuration, connection, fault, protection, synchronization, initialization, recovery
- Experience & Strategy
  - Organize on one dimension, distribute the other
  - Previous product architecture experience
    - one group: system objects
    - another: system functionality
  - Evaluation of both groups
    - neither solution satisfactory
    - going to do the other dimension



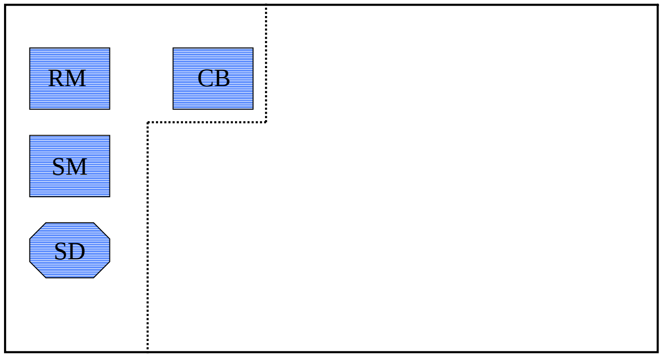
Initial Strategy

- Choose some components in each dimension as the primary architectural components
- Define the distributed components as SW Architectural Styles
  - e.g., constraints on initialization
    - common across all components
    - consistent across all components
  - e.g., fault detection, recovery, etc..

Distr/Reconfig Components

- CB - Command Broker
- SM - System Model
- SD - System Data
- RM - Reconfigure Manager

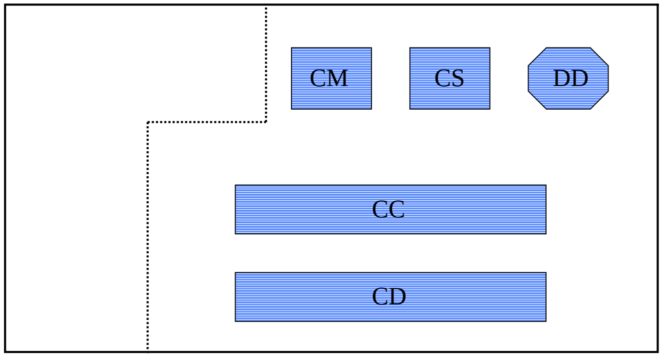
Distr/Reconfig Components



Domain-Specific Components

- CM - Connection Manager
- IM - Integrity Manager
- CS - Connection Services
- CC - Connection Controllers
- CD - Connection Devices (HW)

Domain-Specific Components



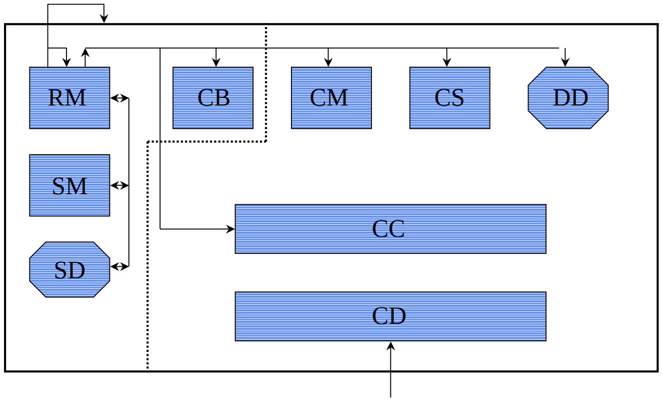
Distribution Components

- System Model/Data (SM/SD)
  - Logical Model
  - Logical to Physical Mapping
  - Priority/Timing constraints
- Command Broker (CB)
  - Operation invocation
  - Operation scheduling

Reconfiguration Components

- Reconfiguration Generation (RG)
  - Outside the fielded system
  - Component generation
  - Completeness/consistency analysis
  - Configuration minimality
- Reconfiguration Manager (RM)
  - Termination of components
  - SM, SD, Component update
  - Registration/Linking
  - Initialization
  - Reflection to be able to replace self

Configuration Connections



Reconfiguration Connections

- RM to self - in case of RM replacement
- RM to entire configuration
- RM to individual components
  - termination first, preserve data
  - reconfigure model and provisioning
  - reconfigure components
- Integrity constraints on connections

Style for Reconfigurable Components

- Location independent
- Initialize:
  - start/restart, rebuild dynamic data, allocate resources, initialize operation
- Finalize:
  - preserve dynamic data, release resources, terminate operation

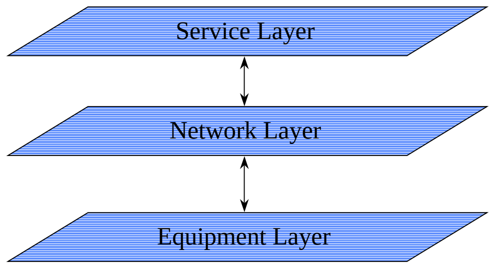
Reconfiguration Generation (RG)

- Problem: maintaining a minimum configuration in the access/transport boxes
  - typically limited space
  - avoid clutter of unused software components
  - minimize reconfiguration time and expense
- Minimal Reconfiguration Solution
  - AED is the set of architectural elements and their dependencies
  - CC is the current architectural element configuration
  - D(X) is the transitive closure of X in AED
  - ADD(AE) = D(AE) - D(CC)
  - DELETE(AE) = D(AE) - D(CC - AE)
  - Do ADDs first

DS Architectural Structure

- CM/CS - use typical architecture for decomposition/layering
  - service layer
  - network layer
  - equipment layer
- IM/CC - distribute using styles

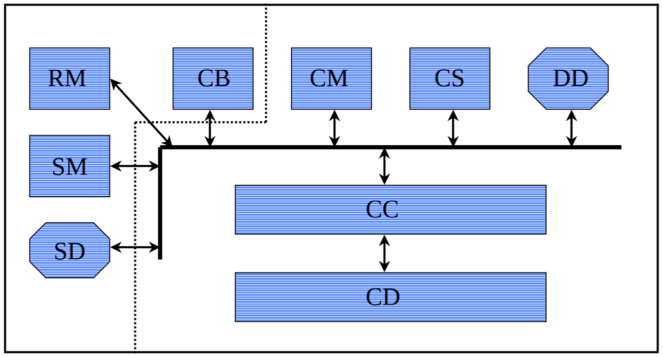
DS Component Decomposition



DS Architectural Connections

- Software bus for
  - Control of interactions
  - access to dynamic and system data
- Performance constraints
- Reliability constraints

Architectural Connections



IM Exception Handling Style

- Recover when can, else reconfigure around fault
- Isolate fault without impacting other components
- Avoid false dispatches
- Provide mechanisms for inhibiting any action
- Do not leave working components unavailable
- Enable working in the presents of faults
- Recover from single faults
- Protect against rolling recoveries
- collect, log appropriate information
- map exceptions to faults
- enable sequencing of recovery actions

Summary

- Techniques for distribution-free and dynamically reconfigurable architecture
  - Data-driven
  - Late dynamic binding
  - Reflection
- Techniques for Domain Specific Organization
  - Primary components - architectural elements
  - secondary components - architectural styles
  - classes of interactions
    - different connectors
    - with different constraints