


Introduction to Software Architectures


hh 1



Goals of Presentation

- Define key terms
 - Architecture
 - Component
 - Connector
 - Configuration/Topology (Styles)
- Understand ways architectures can be used to help SW development


CS578: Software Architecture (2) January 18, 2005



What's A Software Architecture?

- IEEE Standard Glossary of Software Engineering Terminology [IEEE Std 610.12-1990]
 - Organizational structure of a system or component
- IEEE Recommended Practice for Architectural Description of Software-Intensive Systems [IEEE Std 1471-2000]
 - Fundamental organization of a system embodied in
 - Its components
 - Relationships among the components
 - Relationships to the environment (!) Code? Design?
 - Principles guiding its design & evolution (!)

CS578: Software Architecture (3) January 18, 2005



What's A Software Architecture? (cont.)

- "Foundations for the Study of Software Architectures" [Perry & Wolf 92]


Software Architecture =
 { Elements, Form, Rationale }

↑ ↑ ↑

What How Why

Analogy: words, grammar, meaning

CS578: Software Architecture (4) January 18, 2005

 Center For Software Engineering


What's A Software Architecture? (cont.)

☐ **Software Architecture =**
{ Elements, Form, Rationale }

- Highest level concept of a system in its environment
 - Description of elements from which systems are built
 - Interactions among those elements
 - Patterns that guide their composition
 - Constraints on these patterns

☐ **Provides basis for Software Engineering**

CS578: Software Architecture (5) January 18, 2005


 Center For Software Engineering

What's A Software Architecture? (cont.)

☐ *Software Architectures* [Shaw & Garlan 96] defines for a system

- Computation components
 - Clients
 - Servers
 - Databases
 - Filters
 - Layers
- Interactions among components
 - Subprogram calls
 - Shared data
 - Client–server
 - DB–accessing protocols
 - Asynchronous event multicast
 - Piped streams
 - etc.

CS578: Software Architecture (6) January 18, 2005


 Center For Software Engineering

What's A Software Architecture? (cont.)

☐ **Shaw & Garlan (cont.)**

- [A level of design that] involves
 - Description of elements from which systems are built
 - Interactions among those elements
 - Patterns that guide their composition
 - Constraints on these patterns

CS578: Software Architecture (7) January 18, 2005

 Center For Software Engineering


What's A Software Architecture? (cont.)

☐ **Kruchten**

- Design & implementation of high-level structure of software
- Involves
 - Abstraction (Analogy: city map versus city)
 - Decomposition
 - Composition
 - Style
 - Aesthetics

Don't try to understand everything – try to understand what is important!


CS578: Software Architecture (8) January 18, 2005



What's A Software Architecture? (cont.)

- RUP 2002
 - Highest level concept of a system in its environment
 - Organization or structure of system's significant components
 - Interacting through interfaces
 - With components composed of successively smaller components & interfaces


CS578: Software Architecture (9) January 18, 2005



Architectures Are Used To

- Specify solution to problem
 - Designing a system that meets requirements
 - Making trade-offs to satisfy requirements
- Analyze solution
 - Understanding system-level concerns (properties) & goal & -ilities
- Manage software complexity
 - e.g. canonical solutions


CS578: Software Architecture (10) January 18, 2005



Focus Of Software Architectures

- System structure
 - Components + interactions + rules of composition
+ rules of behavior
 - Correspondence between requirements & implementation
 - Context in Environment


CS578: Software Architecture (11) January 18, 2005



Focus Of SW Architectures (cont.)

- Framework for understanding system-level concerns
 - Global rates of flow
 - Communication patterns
 - Execution control structure
 - Scalability
 - Paths of system evolution
 - Capacity
 - Throughput
 - Consistency
 - Component compatibility

CS578: Software Architecture (12) January 18, 2005


 Center For Software Engineering

Why Architectures?

- Key to reducing development costs
 - Explicit system structure (divide and conquer)
 - Component-based development philosophy
- Separation of concerns
 - Computation from communication (!)
 - Architecture from implementation (!)
- Natural evolution of design abstractions for large, complex systems
 - Structure & interaction details **overshadow** choice of algorithms & data structures

More work but less cost?

CS578: Software Architecture (13) January 18, 2005


 Center For Software Engineering

Confused about Terminology?

- Connector
 - Communication, dependency, relationship, link, call, interaction, ...
- Component`
 - Functionality, class, system, subsystem, legacy system, client, server, filter ...
- Style
 - Topology, pattern, composition rules, form, ...

Different terms with similar meaning – but many subtleties


CS578: Software Architecture (14) January 18, 2005

 Center For Software Engineering

Benefits Of Explicit Architectures

- Framework for satisfying requirements
- Technical Basis for
 - Design
 - Consistency analysis
 - Dependency analysis
 - Reuse
- Managerial basis for
 - Cost estimation
 - Process management


CS578: Software Architecture (15) January 18, 2005

 Center For Software Engineering

Key Architectural Concepts

- Component
 - Unit used as a part of some system
 - Locus of computation and state
- Connector
 - Element that models
 - Interactions among components
 - Rules that govern those interactions
- Configuration
 - Connected graph of components & connectors which describes architectural structure


CS578: Software Architecture (16) January 18, 2005



What's A Component?

- ❑ Merriam-Webster's On-line Collegiate Dictionary
 - A constituent part: ingredient
 - Analogy: baking (what, how much, order, when, treatment)
- ❑ Software Architectures [Shaw & Garlan 96]
 - Loci of computation & state
 - clients
 - databases
 - filters
 - servers
 - layers
 - ADTs
 - Has an interface specification that defines its properties
 - how do I have to ask for it?


CS578: Software Architecture (17) January 18, 2005



What's A Component?

- ❑ "Foundations for the Study of Software Architectures" [Perry & Wolf 92]
 - A unit of computation or a data store
 - Perry & Wolf's processing & data elements
- ❑ *Component Software: Beyond Object-Oriented* [Szyperski 97]
 - A unit of
 - independent deployment
 - third-party composition
 - Has no persistent state


CS578: Software Architecture (18) January 18, 2005



What's A Component? (cont.)

- ❑ UML v1.4
 - Modular, deployable, & replaceable part of system that
 - Encapsulates implementation
 - Exposes set of interfaces
 - Specified by one or more classifiers that reside on it
 - May be implemented by one or more artifacts
 - e.g., binary, executable, or script files
- ❑ RUP 2002
 - A non-trivial, nearly independent, & replaceable part of system that fulfills clear function in context of well-defined architecture
 - Conforms to & provides physical realization of a set of interfaces

CS578: Software Architecture (19) January 18, 2005



What's A Component? (cont.)

- ❑ Instructor's Definition:
 - A unit used as a part of some system
 - Essentially same as Perry & Wolf
- ❑ Form of unit depends on
 - Abstraction & architectural style employed
- ❑ Unit used as a component of a system should provide capabilities (behavior and/or information) that are of use to the system
 - If unit doesn't, it should not be a component of this system
- ❑ A component is expected to work with other components to satisfy requirements of system

CS578: Software Architecture (20) January 18, 2005

Center For Software Engineering

Component Goals

- ❑ Frequent system goals include
 - Easy maintenance
 - Easy replacement of parts
 - Easy optimization
 - High reliability
 - High security
- ❑ To achieve such goals
 - Components' interfaces should be well defined
 - Implementation of components should be hidden
 - Dependence on other components should be minimal

CS578: Software Architecture (21) January 18, 2005

Center For Software Engineering

Simple & Composite Component

- ❑ Components may be simple or composite
 - Parts of "Simple" components are different level of abstraction
 - Heart & Cells (cell is not a heart, also used to build other organs)
 - Screw & Molecules/Atoms
 - Class & Data & Operations
 - "Composite" component has parts that are the same level of abstraction
 - Body & Organ (e.g. heart)
 - Car & Engine;
 - Process & Thread
 - Based on simple and complex components
 - "System of Systems" (what is a system? what is a component?)

CS578: Software Architecture (22) January 18, 2005

Center For Software Engineering

Connectors

- ❑ A connector is an architectural element that models
 - Interactions among components
 - Rules that govern those interactions
- ❑ Simple interactions
 - Procedure calls
 - Shared variable access
- ❑ Complex & semantically rich interactions
 - Network protocol
 - Client-server protocols
 - what rule governs this interaction?
 - What does the server know about the client? Restricted communication!
 - Database access protocols
 - Asynchronous event multicast
 - Piped data streams

CS578: Software Architecture (23) January 18, 2005

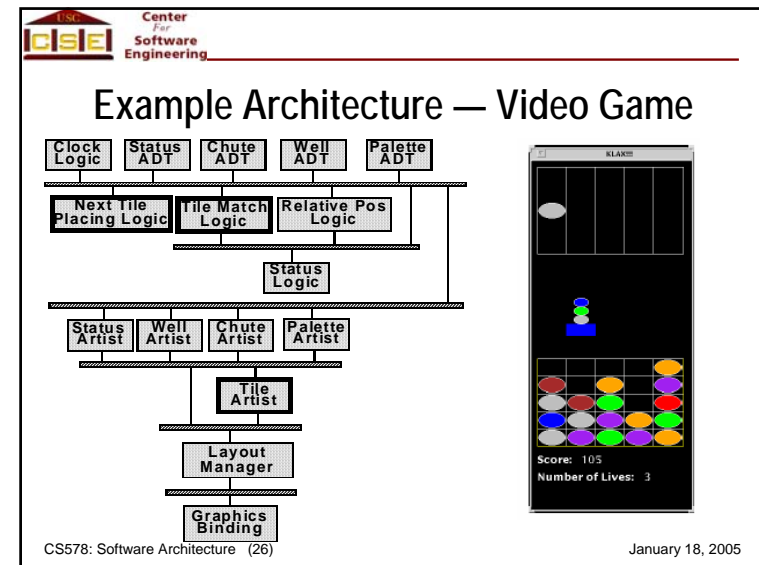
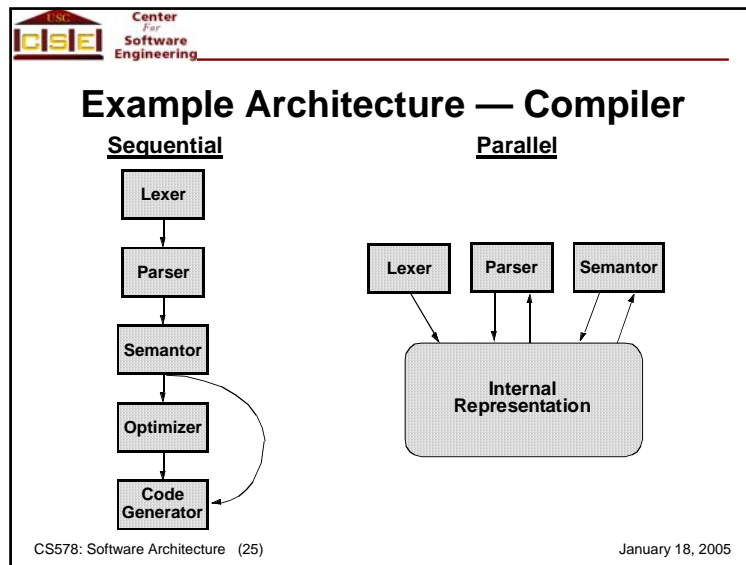
Center For Software Engineering

Configurations/Topologies

- ❑ Architectural configuration or topology
 - Describes architectural structure
 - Proper connectivity
 - Concurrent & distributed properties
 - Adherence to design heuristics & style rules
 - client-server has a well-defined configuration (server-server connector)
 - Represented by connected graph of components & connectors
- ❑ Each composite component has a configuration

The diagram illustrates a component C that is connected to three other components: A, B, and D. Component C is shown as a box with a dashed border, and its internal structure is detailed in a larger box to its right. Inside this larger box, C is represented as a sub-component with a solid border, containing seven smaller sub-components labeled C1 through C7. C1 and C2 are at the top, C3 and C4 are in the middle, and C5, C6, and C7 are at the bottom. Lines connect component C to A, B, and D, and lines also connect C to its internal sub-components C1 through C7.

CS578: Software Architecture (24) January 18, 2005



- Center For Software Engineering
- ### Analogies To Software Architecture
- Hardware architecture
 - Small number of design elements
 - Scale by replication of (canonical) design elements
 - Network architecture
 - Focus on topology
 - Only few topologies considered
 - e.g. Star, ring, grid
 - Building architecture
 - Multiple views
 - Styles
- CS578: Software Architecture (27) January 18, 2005

- Center For Software Engineering
- ### Current Treatment of Software Architectures
- Understood at level of intuition, anecdote, & folklore
 - Informal descriptions
 - boxes & lines
 - informal prose
 - Semantically rich vocabulary that conveys a lot
 - RPC
 - client-server
 - pipe and filter
 - layered
 - distributed
 - OO
 - Is this level of informality really a critical problem?
- CS578: Software Architecture (28) January 18, 2005

Center For Software Engineering

What Are Software Architectures Used for?

- Representation
- Design Process Support
 - Analysis
 - Static
 - Dynamic
 - Evolution
 - Specification-Time
 - Execution-Time
 - Refinement
- Traceability
- Simulation/Executability

CS578: Software Architecture (29) January 18, 2005

Center For Software Engineering

Representation

- Principal problems
 - Aid stakeholder communication & understanding
- Desired solutions
 - Multiple perspectives
- Achievable via
 - Graphical notations
 - Explicit configuration modeling
 - Additional views: control flow, data flow, process, resource utilization

CS578: Software Architecture (30) January 18, 2005

Center For Software Engineering

Design Process Support

- Principal problems
 - Design collaborating collection of components for large, distributed, heterogeneous systems
- Desired solutions
 - Multiple perspectives
 - Design guidance & rationale
- Achievable via
 - Active support for specification
 - Proactive vs. Reactive
 - Non-intrusive vs. Intrusive


CS578: Software Architecture (31) January 18, 2005

Center For Software Engineering

Static Analysis

- Principal problems
 - Evaluate system properties early
 - Reduce number & cost of errors
 - Architecture is analyzed without executing it
- Desired solutions
 - Internal consistency
 - Concurrent & distributed properties
 - Design heuristics & style rules
- Achievable via
 - Parsers, compilers, model checkers
 - Schedulability & resource utilization analyzers
 - Design critics
 - Mismatch analysis


CS578: Software Architecture (32) January 18, 2005



Dynamic Analysis

- Principal problems
 - Same as static analysis
 - Architecture is analyzed during execution
 - How do you execute an architecture?
- Desired solutions
 - Testing & debugging
 - Assertion checking
 - Specification & checking of important runtime properties
- Achievable via
 - Scenarios
 - Simulation
 - Event visualization & filtering


CS578: Software Architecture (33) January 18, 2005



Specification-Time Evolution

- Principal problems
 - Evolution of design elements, systems, and system families
- Desired solutions
 - Architectural equivalent of subtyping/refinement
 - Architectural configuration management
 - Incremental specification system families
- Achievable via
 - Heterogeneous, flexible subtyping mechanisms
 - Explicit & flexible connectors
 - Explicit specification of application family


CS578: Software Architecture (34) January 18, 2005



Execution-Time Evolution

- Principal problems
 - Same as specification-time evolution
 - Must be accomplished during system execution
- Desirable solutions
 - Replication, insertion, removal, reconnection, migration
 - Planned or unplanned
 - Constraint satisfaction
- Achievable via
 - Constrained & unconstrained (“pure”) dynamism
 - Conditional configuration
 - Replication
 - Analysis of architecture during system modification


CS578: Software Architecture (35) January 18, 2005



Refinement

- Principal problems
 - Bridge gap between informal diagrams & programming languages
- Desired solutions
 - Specify architectures at different abstraction levels
 - Correct & consistent refinement across levels
- Achievable via
 - Correctness-preserving mappings
 - Comparative simulations


CS578: Software Architecture (36) January 18, 2005

 Center For Software Engineering

Traceability

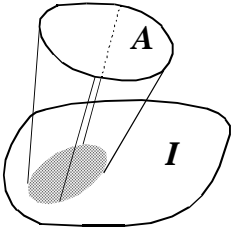
- ❑ Principal problems
 - Multiple abstraction levels + multiple perspectives
- ❑ Desired solutions
 - Traceability across architectural cross-sections
- ❑ Achievable via
 - Well established relationships among architectural perspectives
 - Mapping from requirements to architecture

CS578: Software Architecture (37) January 18, 2005


 Center For Software Engineering

Simulation/Executability

- ❑ Principal problems
 - Checking dynamic properties requires running system
 - Early prototypes needed to demonstrate features to stakeholders
- ❑ Desired solutions
 - Construct simulations
 - Systematic support for system generation
- ❑ Achievable via
 - Simulating event sequences
 - Restricting implementation space




CS578: Software Architecture (38) January 18, 2005

 Center For Software Engineering

Summary

- ❑ Software Architecture =
 - { Elements, Form, Rationale }
 - Highest level concept of a system in its environment
 - Description of elements from which systems are built
 - Interactions among those elements
 - Patterns that guide their composition
 - Constraints on these patterns


CS578: Software Architecture (39) January 18, 2005

 Center For Software Engineering

Summary (cont.)

- ❑ Architectures support
 - Communication
 - Understanding
 - Analysis
 - Evolution
 - Refinement
 - Traceability

CS578: Software Architecture (40) January 18, 2005



Summary (cont.)

- ☐ Three canonical building blocks
 - Component
 - Unit used as part of some system
 - Locus of computation & state
 - Connector
 - Element that models
 - Interactions among components
 - Rules that govern those interactions
 - Configuration
 - Connected graph of components & connectors which describes architectural structure

CS578: Software Architecture (41) January 18, 2005