Who Am I?

- Research Associate, Center for Software Engineering
- 20 years industrial teaching & consulting on object-oriented methods, software engineering, & programming languages
- Consultant on definition of Architecture Design Language (ADL) for real-time, safety-critical systems
  - Based on Unified Modeling Language (“UML”) and Honeywell’s MetaH
  - To be proposed as standard of Society of Automotive Engineers (SAE)
- Created Colbert Object-Oriented Software Development method (“OOSD”)
  - Noted for strength in real-time software development
    - NASA Langley Research Center used for software engineering process manual
- MBASE developer
- Founded Absolute Software Co., Inc. in 1986
Goal of Presentation

- Understand how to perform System Analysis
  - Using
    - MBASE
    - Object-oriented techniques
    - RUP
    - Rational Rose
- Understand how to document analysis

Outline

- Key Concepts
- Process Overview
- Example Project Description
- Process by Example
What's A Software Architecture?

- Perry & Wolf
  - Software Architecture = { Elements, Form, Rationale }

- IEEE Std 1471-2000
  - Fundamental organization of a system embodied in
    - Its components
    - Relationships among the components
    - Relationships to the environment
    - Principles guiding its design & evolution

- 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

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 even multicast
  - Piped streams
  - etc.
What’s a Component?

- **Software Architectures** [Shaw & Garlan 96]
  - Loci of computation & state
  - Has an interface specification that defines its properties

Purposes of Architecture Analysis & Design

- To transform requirements into design of system
- To evolve robust architecture for system
- To adapt design to match implementation environment
  - Designing it for performance
    - RUP 2001
Object, Component, System, People, Organization, ….

Object, Component, System, People, Organization, … (cont.)
What You Need To Describe At Any Level?

- Structure
  - External Perspective (Specification/Interface)
  - Internal Perspective (Implementation)
- Responsibilities
  - its state
  - its dynamic behavior
  - the operations it can perform
  - the requests it makes of other objects
    - Each component/object collaborates (works) with other components/objects in performance of its responsibilities
- Qualities
  - Quantitative (e.g., size, speed)
  - Qualitative (e.g., “user friendly”, re-usable)
MBASE Describes **Current Organization** in OCD Section 3.3

- Structure
- Responsibilities
  - its state
  - its dynamic behavior
  - the operations it can perform
  - the requests it makes of other objects
- Qualities
  - Quantitative
  - Qualitative

- Sec 3.3.1
- Sec 3.3.2
- Sec 3.3.3
- Sec 3.3.4
- Sec 3.2

MBASE Describes **Changed Organization** in OCD Section 4

- Structure
- Responsibilities
  - its state
  - its dynamic behavior
  - the operations it can perform
  - the requests it makes of other objects
- Qualities
  - Quantitative
  - Qualitative

- Sec 4.3.1
- Sec 4.3.2
- Sec 4.3.3
- Sec 4.3.4
- Sec 4.2
- Sec 4.4

Also captures external view of System for Managers & Domain experts
### MBASE Describes System

**External View in SSAD Section 2**

<table>
<thead>
<tr>
<th>Structure</th>
<th>Responsibilities</th>
<th>Qualities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>its state</td>
<td>Quantitative</td>
</tr>
<tr>
<td></td>
<td>its dynamic behavior</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the operations it can perform</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the requests it makes of other objects</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sec 2.1</td>
<td>Sec 2.4</td>
</tr>
<tr>
<td></td>
<td>Sec 2.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sec 2.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sec 2.5</td>
<td></td>
</tr>
</tbody>
</table>

**Qualities**
- Quantitative
- Qualitative

### MBASE Describes System Internal

**(Architectural) View in SSAD Section 3**

<table>
<thead>
<tr>
<th>Structure</th>
<th>Responsibilities</th>
<th>Qualities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>its state</td>
<td>Quantitative</td>
</tr>
<tr>
<td></td>
<td>its dynamic behavior</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the operations it can perform</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the requests it makes of other objects</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sec 3.1</td>
<td>Sec 3.4</td>
</tr>
<tr>
<td></td>
<td>Sec 3.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sec 3.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sec 3.5</td>
<td></td>
</tr>
</tbody>
</table>

**Qualities**
- Quantitative
- Qualitative

Independent of implementation details
MBASE Describes System Internal (Implementation) View in SSAD Section 4

- Structure
- Responsibilities
  - its state
  - its dynamic behavior
  - the operations it can perform
  - the requests it makes of other objects
- Qualities
  - Quantitative
  - Qualitative

- Sec 4.1 & 4.5
- Sec 4.2
- Sec 4.3
- Independent of implementation details
- Sec 4.4

Analysis & Design Approach

- Architecture–centric
  - Define stable architecture early
- Use–case Driven
  - Use–case describe capabilities/usage of system
  - Use–case help identify objects & operations needed
- Iterative & Incremental
- Risk Driven
Architecture Design & Analysis Goals

- Describe what we want to create
  - Not how it is done
- Model high-level architectural of system
  - Formalize & refine system features described in domain description
    - Capabilities
    - Artifacts
    - Processes
    - Levels of Services
    - Rules
- Validate that architecture achieves required system
  - Behavior
  - Goals

Architecture Analysis Audience

- Description is for all constituents of project
- Analysis is for Domain Experts
  - i.e. high level leaders who
    - Understand domain
    - Know what they want,
    - Have authority to make decisions
- Designers
  - Who design & implementation details (“how”)
Outline

- Key Concepts
- Process Overview
- Example Project Description
- Process by Example
Outline

- Key Concepts
- Process Overview
- Example Project Description
- Process by Example

OCD 2.1 System Capability Description

- Target customer: Information Service Division (ISD) in University of Southern California
- The system is proposed and implemented for faculty, students, researchers and librarians at USC who need to identify where a particular periodical title is indexed, the date of coverage and whether it is available in full-text.
- The Full-text Title Database is a web-based full-text journal title searchable database.
- That system retrieves and consolidates journal title information from different vendors and provides it to general public users such as faculty, students, researchers and librarians.
- Unlike the Jake project at Yale University and full-text journal and newspaper search engine at Indiana University, our product is designed dedicated to USC which is accessible globally via World Wide Web and provides full-text journal title information to general public especially USC community. The Jake project of Yale University and the journal and newspaper search engine at Indiana University provide similar functionality. However, the former includes some information in those vendors’ databases that USC does not have access to and the later only contains journal title information from databases that Indiana University has access to which is inadequate.
OCD 2.1.1 Benefits Realized

- The database we are going to build will let the general public users (faculty, students, researchers, librarians, etc.) easy to find the full-text journal title information they need. They can do a search in our database instead of searching each vendor’s database USC has access to one after another. Thus it will much reduce the full-text journal searching time in the research. Since it’s an easy to use and powerful system, more and more users are willing to use our system. It leads to more frequent and efficient utilization of resources of ISD, and more productive research work by faculty, students and researchers at USC. And it will lead to better ISD service and a higher academic reputation for USC.

- The database we are going to build will provide a web-base interface to system administrator, so ISD can easily find a person who has some basic computer knowledge to maintain the whole system, and doesn’t have to provide much training to him/her or additional resources. Therefore ISD can save budget and time on a system which is easy to maintain like this proposed system.

OCD 2.1.2 Results Chain

- Construct a full-text journal title searchable database
- Collect and consolidate full-text journal title & location information into a single database
- Easy to search & locate full-text journal titles
- Faculty, students, researchers and librarians are willing to use the system
- More frequent and efficient utilization of ISD resources
- More users' feedback
- ISD will improve its quality of services if it gets more users' feedback
- Better ISD services
- More published research papers
- More productive research
- Less time to search full-text journals
- A university with more published papers has higher academic reputation.
- Higher academic reputation for USC
OCD 2.2 Key Stakeholders

- Software/IT system’s users: Faculty, Researchers, Students, and Librarians at USC.
  - The relationship with result chain: The users will be the ones who are going to interact with our system through user interface. If they are willing to use our system, they will get the benefit which is to “fill a serious gap in the tools available and result in more efficient research for researchers and students”.
- Customers: ISD
  - The relationship with result chain: Whom we are going to develop the system for. This is also the one who is going to inspect the result chain.
- Developers: CS577a Team & CS577b Team for Full-text Title Database
  - Developers are responsible for design, construct and implement the software system. They are also responsible for helping the customer and provide some training to the system administrator and software maintainer during the transition phase of the system.
  - Relationship with result chain: We are the people who develop the result chain.
- System Administrator: Greg Fleming
  - Greg Fleming will be the one who is assigned by ISD and responsible for updating the system database. He needs to observe the vendors’ databases. Once vendors update their databases, our unified database should also be updated.
  - Relationship with result chain: He is responsible for ensuring the system to provide the updated information to users so that they can actually get the expected benefits.
- Software Maintainer: ISD employee
  - ISD will assign a person to maintain the system. He/she is responsible for maintaining the software code and crash recovery of the system.
  - Relationship with result chain: He/she is responsible for ensuring the system stability and availability so that the expected benefits can be realized.
- Data Source: Vendors
  - Vendors are the commercial organizations which provide databases containing the full-text journals and subscription service to others.
  - Relationship with result chain: The initiative of the result chain. We assume the vendors’ databases are searchable. Our unified Fulltext Title Database will be generated from their databases.

OCD 2.3 System Boundary & Environment

[Diagram showing the system boundary and environment with relationships between Library User, Librarian, Researcher, Student, Faculty, ISD System Administrator, ISD System Maintainer, Vendor Database, Full-Text Title Database System, and other stakeholders.]
OCD 3.2 Organization Goals

- **OG-1**: Fast Information Access
  - Provide a faster way to access information
- **OG-2**: Reduced workload
  - Allowing the librarians to focus their time & energy on those patrons that really need help
- **OG-3**: User friendly environment
- **OG-4**: Enhance the library collection
  - Enhance the library collection by subscribing to vendors’ full-text journal databases
- **OG-5**: Ensure the Integrity of Information
  - Provide a way to ensure the integrity of the information and protect the contents of the system
- **OG-6**: Make the Resource Available to Distributed Users
  - Users may be locate at any place and want to access the resources

---

OCD 4.3 System Capabilities

- **Library User**
  - Search for Full-text Journal Title
- **System Administrator**
  - Maintain Journal Data
  - Maintain Vendor Data Profile
  - Set Maintenance Password

[Diagram showing the system capabilities]
OCD 4.5.1 Proposed Organization Structure

Classifiers

Library User

Full-Text Title Database System

System Administrator

Full-Text Title Database

Typical Configuration of Instances

For this system,

- There’s really only 1 configuration
- Doesn’t add much information
OCD 4.5.3 Proposed Processes
Search for Full-Text Journal Title

Other activities not shown
Outline

- Key Concepts
- Process Overview
- Example Project Description
- Process by Example

Architecture Analysis & Design

- Sub-activities not ordered
  - i.e. can start anywhere

- Why not?
  - Iterative Process
Define Architecture

- **Purpose:**
  - Define candidate architecture for system
  - Based on experience gained from similar systems or in similar problem domains
  - Identify SW & HW Components
  - Understand
    - Hardware execution environment
    - Allocation of components to hardware

- **Inputs:**
  - Proposed System
  - Proposed Entity Model
  - Behavior Model
  - Architectural Patterns
  - L.O.S Requirements
  - Evolution Requirements

Define Architecture Artifacts

- **Hardware Component Classifier Model**
  - Identify hardware component classes
  - Identify connections between components

- **Software Component Classifier Model**
  - Identify hardware component classes
  - Identify connections between components

- **System Deployment Model**
  - Describe configuration(s) of hardware component s
  - Allocate software components to hardware components
Define Architecture
Hardware Component Classifier Model

- **Goal**
  - Understand execution environment that SW will run on

- **Model describes**
  - Classification of hardware components
    - UML calls “node classes”
  - Classification of connections between components

- **Artifacts**
  - 1 or more Hardware Classifier Diagrams
  - Descriptions of each hardware component & connector classifier

---

Define Architecture
Hardware Component Classifier Diagram for Full-Text Title Database System

```
<<node>> Workstation
   +client 0..*

<<connector>> Network
   +server 1

<<node>> Unix Server
```
Define Architecture
Considerations for E-Business Systems

- Existing network logical & physical design
- Existing databases & database design
- Existing Web environment
  - Servers, firewalls etc.
- Existing server environment
  - Configuration, software versions, planned upgrades
- Existing standards
  - Network, naming, protocols etc.

Define Architecture
Software Component Model

- Common components
  - Clients
  - Servers
  - Databases
  - Filters

- Identify connections
  - i.e. which components need to communicate
Define Architecture
Software Component Model
(cont.)

- Look for Components
  - Loci of computation & state
  - Modular, deployable, & replaceable part of system
    - i.e. can be independently
      - Ordered, configured, or delivered
      - Developed, as long as the interfaces remain unchanged
      - Deployed across a set of distributed computational nodes
      - Changed without breaking other parts of the systems
  - Units which can provide restricted security over key resources
  - Existing products or external systems in design

- Hints from System External Structure (Context) Model
  - Look to Actors
    - Separate functionality used by two different actors
    - Each actor may independently change their requirements
Define Architecture
Software Component Model (cont.)

- Hints from System Artifacts & Information Model
  - Look to user interface
    - If relatively independent of entity classes
      - i.e. interfaces & entities can & will change independently
        - Create components which are horizontally integrated
          - Component(s) for related user interface boundary classes
          - Component(s) for related entity classes
    - If tightly coupled with entity classes
      - i.e. change in one triggers a change in other
        - Create components which are vertically integrated
          - Component(s) for related boundary & entity classes
  - Look for optional groups of business entities
    - i.e. Group of entities that
      - Represent optional behavior
      - May be removed, upgraded, or replaced with alternatives
  - Look for coupling & cohesion between classes
    - i.e. highly coupled or cohesive classes collaborate to provide some set of services
    - Organize highly coupled classes into component
    - Separating classes along lines of weak coupling
      - May be able to eliminate weak coupling by splitting classes into smaller classes with more cohesive responsibilities
Define Architecture
Software Component Model
(cont.)

Hints from System Artifacts & Information Model (cont.)

- Look at substitution
  - If there are several levels of service specified for a particular capability
    - i.e. high, medium and low availability
  - Represent each service level as separate subsystem
  - Each of which will realize the same set of interfaces

- Subsystems are substitutable for one another

Define Architecture
Software Component Model
(cont.)

- Look at distribution
  - Multiple instances of particular component can execute on different nodes
  - Single instance of component can’t be split across nodes
  - If component behavior must be split across nodes
    - Decompose component into smaller components with more restricted functionality
  - Create component for functionality that must reside on each node
Define Architecture
Software Component Model
(cont.)
- Based on past experiences
  - General
  - Domain-specific books & papers
  - Technology-specific
    - During design
- Decision
  - Client-Server Model

Define Architecture
Software Component Classifier Model For Full-text Title Database System, Top–Level

```
<<Component>>
Library_User
<<Component>>
System_Administrator
<<Component>>
Server
```
Define Architecture
Software Component Classifier Model For
Full-text Title Database System,
Server Component

- Note: using a style that allows nested component classifiers

Define Architecture
Deployment Model

- Goal
  - Understand
    - Execution environment
    - Distribution of software components across environment

- Model describes
  - Configuration(s) of component instances
    - “components”
  - Allocation of software components to hardware

- Artifacts
  - 1 or more Collaboration Diagram
Define Architecture

Deployment Diagram for Full-Text Title Database System

Define Analysis Classes

- **Purpose:**
  - Create model of the information classes needed to represent
  - Artifacts & information required by system
  - Forms defined in current prototype
  - Information communicated between components
  - Control behavior specific to one or a few processes performed by architectural units
    - i.e. “recipes” that define how to do something

- **Inputs:**
  - System Context Model
  - System Artifacts & Information Model
  - Architecture Behavior

- **Artifacts:**
  - Updated Component Model
  - Analysis Class Model
Define Analyze Classes
Guide to Creating Class Model

- Create draft Analysis Class Model
  - Create diagram(s) that shows preliminary
    - Classes
    - Relations
      - Associations
      - Dependencies?
      - Generalizations

- Warning: don’t get carried away defining class

Define Analyze Classes
Guide to Creating Class Model

- First-cut class identification
  - Define how entities in System Artifact & Information Model will be represented
    - Often 1 class per entity
    - Stereotype <<entity>>
    - Define attributes for class based on attributes of entities
    - If state of some instances need to be maintained between execution
      - Set Persistence property to True
Define Analyze Classes
Guide to Creating Class Model (cont.)

First-cut class identification (cont.)

- For each window/screen/form in user interface
  - Create a class
  - Stereotype <<boundary>>

- Review use-case description for nouns
  - Determine whether describes class or object
    - If object determine its class
  - If classes found that are not in model
    - Add it
    - Determine stereotype

Define Analyze Classes
Guide to Creating Class Model (cont.)

First-cut class identification (cont.)

- Create class to coordinate work of use-case
  - If logic is not particularly related to
    - user interface issues (boundary objects)
    - data engineering issues (entity objects)
  - Stereotype <<control>>
Define Analyze Classes
Guide to Creating Class Model (cont.)

- Identify preliminary relations
  - Associations
    - Relations between concepts that indicates some
      - Meaningful connection
      - Interesting connection
    - Associations worth noting & including on Conceptual Model
      - Imply knowledge of relationship that needs to be preserved
      - Derived from Common Associations list
      - Compelling or useful in context of requirements
  - High priority associations
    - Invariably useful in conceptual model
      - A is physically or logical part of B
      - A is physically or logically contained in/on B
      - A is recorded in B
    - Named based on
      - Matching the pattern “Class Name – Verb Phrase – Class Name”
      - Creating readable & meaningful sequence in model context
  - Dependencies
    - Knowledge of other class required, but no permanent association
      - Hint: think of classes of parameters or temporary objects need in an operation
  - Generalizations
    - From special class to general class

- Repeat: don’t get carried away
Define Analyze Classes
Enterprise Classification Model for Full-Text Title Database System

Define Analyze Classes
Business-Analysis Class Mapping for Full-Text Title Database System
Analyze Behavior

**Purpose:**
- Understand how system behavior is realized by components & information classes

**Inputs:**
- Draft Software Component Model
- System Behavior Model
- Draft Analysis Class Model

**Artifacts:**
- Sequence Diagram for each use–cases in Behavior Model
- Updated Analysis Class Model
- Updated Component Model

Analyze Behavior (1st Build)
Search and Locate Journals

1. User requests search
2. System displays search page
3. User enters search criteria
4. System queries database asking for journals that match user’s search criteria
5. Result := Search(request)
6. result_set := Create(journals matching criteria)
7. result := Create(result)
8. System displays journal list in search results page or Displays error page that asks the user to search again.

User requests search
System displays search page
User enters search criteria
System queries database asking for journals that match user’s search criteria
System displays journal list in search results page or Displays error page that asks the user to search again.
System Analysis
What Have We Done?

- Developed a preliminary
  - Architecture model for our system
  - Behavior Model of system
  - Classes model

Architecture Analysis & Design
What Have Left To Do?

- Complete descriptions of Behavior
- Refine Architecture
- Refine Analysis Classes
- Demonstrate feasibility of our architecture
- Review
Architecture Analysis & Design

Conclusions

- We now have seen
  - How to perform Architecture Analysis & Design
    - Using
      - MBASE
      - Object-oriented techniques
      - RUP
      - Rational Rose
  - How to document analysis

- Time for you to trying on your problem
  - Then you’ll really understand!