Why do we care about Software Architecture?

• Because we want to
  ▪ be a dominant player in our industry/market
  ▪ deal with organizational or technical complexity
  ▪ enable something that is not possible/feasible today
  ▪ establish a shared technology foundation for a product line
  ▪ be in business in 5 years

  ▪ want a product, system or family of applications to have qualities or system characteristics such as a high level of integration, evolvability, understandability
Software Architecture
Components and Relationships

Conceptual Architecture
- Abstract, system-wide view
- Basis for communication

Logical Architecture
- “Blueprint”: Precise, unambiguous, actionable
- Basis for supplier/client contract
Software Architecture Components and Location

Execution Architecture
- Configuration of components at run-time
- Basis for system tuning during design

Software Architecture System-Level Concerns

<table>
<thead>
<tr>
<th>Principles</th>
<th>Style</th>
<th>Key Mechanisms and Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principle Name</td>
<td>Description</td>
<td>e.g., component interconnection mechanisms</td>
</tr>
<tr>
<td>Rationale/Benefits</td>
<td>Implications</td>
<td>Meta-Architecture</td>
</tr>
<tr>
<td>Counterargument</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Meta-Architecture
- Guiding principles and strategies
- Basis for system decomposition and composition
Architecting How To
Guiding Principles and Strategies

<table>
<thead>
<tr>
<th>Principle Name</th>
<th>Give the principle a catchy name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Statement of the principle.</td>
</tr>
<tr>
<td>Rationale/Benefits</td>
<td>Describe the reasoning behind the principle. Where applicable, provide traceability to business or architectural objectives.</td>
</tr>
<tr>
<td>Implications</td>
<td>Identify implications such as actions that need to be undertaken, and constraints implied by the principle.</td>
</tr>
<tr>
<td>Counterargument</td>
<td>Describe the reasonable counter to this principle.</td>
</tr>
</tbody>
</table>

Architecting How To
Identify Components (initial cut)

<table>
<thead>
<tr>
<th>Clerk</th>
<th>Dialog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance Books</td>
<td>Receive Payments</td>
</tr>
<tr>
<td>Enter</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Give the component an easy-to-remember name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responsibilities</td>
<td>List the responsibilities assigned to the component</td>
</tr>
<tr>
<td>Collaborators</td>
<td>List of other components this component depends on for (delegated) services [out-ports]</td>
</tr>
<tr>
<td>Rationale</td>
<td>State the rationale for allocating responsibilities to this component. Provide traceability to functional requirements and qualities or meta-architecture.</td>
</tr>
<tr>
<td>Issues and Notes</td>
<td>List assumptions, constraints, unknowns, etc.</td>
</tr>
</tbody>
</table>
Architecting How To Model System Behavior

Key principle: Form follows Function
- Assign responsibilities to components to accomplish required services taking into account system qualities
- Key tool: Collaboration Diagrams

Architecting How To Document Interfaces

I/F Element | Description
--- | ---
Interface name | A unique identifier for the interface
Exceptions | The name and data content for each operation's exceptions
Properties | The name and type of each property
Operations | The name of each operation, together with the input and output parameters and exceptions
Operation descriptions | Description of each operation using
  - informal description or
  - pre/post condition template
  - example showing typical calling usage (optional)
Protocol (optional) | Constraints on the order in which operations may be called (Statechart)
Service Level (optional) | Non-functional requirements to be met by the services provided by the interface (operations)
Notes and Issues | List of components using I/F
                   | List of issues to be resolved
Architecting How To
Allocate Components to Processes

Designates a process
Asynchronous message

Node on which process resides
Communicates across Beans messaging service

Key tool: Collaboration Diagrams

Diagram from Booch et al, 1999

Architecting How To
Functional Requirements

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Validate User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Customer</td>
</tr>
</tbody>
</table>
| Steps           | 1. The system prompts the customer for a PIN number
|                 | 2. Customer enters the PIN number
|                 | 3. The Customer commits the entry
|                 | 4. The system checks the PIN to see if it is valid
|                 | 5. If valid, system acknowledges the entry
| Variations      | The Customer can cancel at any time, thus restarting the use case. No changes are made to the Customer’s account.
|                 | The Customer can clear the PIN any time before committing it and re-enter the PIN.
|                 | If the Customer enters an invalid PIN, the use case restarts. If this happens 3 times in a row, the system cancels the transaction.

Software Architecture Action Guide 4/6/00
Slide 11

Software Architecture Action Guide 4/6/00
Slide 12

Copyright 2000 Bredemeyer Consulting
http://www.bredemeyer.com
## Architecting How To

### Non-Functional Requirements

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Use case identifier and reference number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Goal to be achieved by use case and sources for requirement</td>
</tr>
<tr>
<td>Actors</td>
<td>List of actors involved in use case</td>
</tr>
<tr>
<td>Assumptions</td>
<td>Conditions that must be true for use case to terminate successfully</td>
</tr>
<tr>
<td>Steps</td>
<td>Interactions between actors and system that are necessary to achieve goal</td>
</tr>
<tr>
<td>Variations (optional)</td>
<td>Any variations in the steps of a use case</td>
</tr>
<tr>
<td>Non-Functional</td>
<td>List of non-functional requirements that the use case must meet</td>
</tr>
<tr>
<td></td>
<td>The nonfunctional requirements are listed in the form: <code>&lt;keyword&gt; : &lt; requirement&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Non-functional keywords include, but are not limited to Performance, Reliability, Fault Tolerance, Frequency, and Priority. Each requirement is expressed in natural language or an appropriate formalism.</td>
</tr>
<tr>
<td>Issues</td>
<td>List of issues that remain to be resolved</td>
</tr>
</tbody>
</table>

---

## Architecting How To

### Architecture Context

- **Context Map**
  - Graphical History
  - Based on Grove Graphics Guides
Architecting How To
Sequence

Requirements

- System context
- Stakeholder goals
- Use Cases
- Qualities
- Concurrency

Structuring

- Guiding principles and strategies
- Components and responsibilities
- Interfaces
- Mapping to processes and nodes

Architecting How To
Validation

<table>
<thead>
<tr>
<th>Scenario ID</th>
<th>Scenario Description</th>
<th>Change Req’d? Y/N</th>
<th>Description of Changes Required</th>
</tr>
</thead>
</table>

Key tool: **Impact Assessment Table**
Architecting How To
Process Overview

- Init/Commit
- Architectural Requirements
- System Structuring
- Architecture Validation
- Deployment

Role of the Architect

- Envision
- Listen
- Champion
- Conceptualize
- Model Design
- Translate
- Unpack ‘ilities’
- Spot trends
- Prioritize

- Requirements and Scoping
- System Structuring
- Critique
- Prototype
- Review/Assess
- Consult
- Educate
- Police

- Lead
- Deployment