Information Systems Architecture
Stakeholders, Viewpoints, Perspectives

Eoin Woods
Barclays Global Investors
Content

- Defining Software Architecture
- Stakeholders
- The Software Architecture Problem
- Viewpoints to Guide Structure
- Perspectives to Guide Qualities
- Example Application
- Uses for Viewpoints and Perspectives
A common definition ...

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements the externally visible qualities of those elements, and the relationships among them.

Len Bass, Paul Clements and Rick Kazman (SEI)
Software Architecture in Practice, 2nd Edition
Defining Software Architecture

- Alternative definitions ...
  - The set of system design decisions that dictate the fundamental structure and properties of a system. Thus, the set of decisions that will cause the system to fail if made incorrectly.

- The set of design decisions that, if made wrongly, cause your project to be cancelled!
Role of Software Architecture

The bridge between requirements and design
Architecture & Requirements

- Requirements are an input to architecture
  - Requirements frame the architectural problem
  - Stakeholder needs and desires

- Architecture must influence requirements
  - “The art of the possible”
  - Stakeholder understanding of risk/cost
  - Stakeholder understanding of possibilities
Architecture & Design

- Architecture frames design
  - architecture is part of the design process
- Captures the system-wide decisions
  - what has to be consistent or constant
- Importance of role increases with scale
- Perfectly compatible with agile
  - even XP with the right approach
“Architecture, Design, Implementation”, Amnon Eden And Rick Kazman, ICSE 2003

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Global</th>
<th>Intentional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>Local</td>
<td>Intentional</td>
</tr>
<tr>
<td>Implementation</td>
<td>Local</td>
<td>Extensional</td>
</tr>
</tbody>
</table>

Intentional: infinitely many possible ways of satisfying the statement (i.e. constraint rather than instruction)  
Local: satisfied in design "d" => satisfied in all possible extensions of "d"
Quality Properties

- Non-functional characteristics (“-illities”)
  - Performance, Security, Availability, ...

- Often crucial to stakeholders
  - Slow functions don’t get used
  - Unavailable systems stop the business
  - Security problems cause headlines

- Yet often an after-thought
Quality Properties

- Addressing quality properties is a key architectural task
  - Understanding real stakeholder needs
  - Understanding what is possible
  - Making the key trade-offs to allow delivery
  - Avoiding expensive “retro-fit”
Content

- Defining Software Architecture
- Stakeholders
- The Software Architecture Problem
- Viewpoints to Guide Structure
- Perspectives to Guide Qualities
- Example Application
- Uses for Viewpoints and Perspectives
Stakeholders

- **Identifying Stakeholders**
  - People, Groups, Entities
  - Those who have an interest in or concerns about the realisation of the architecture

- **Importance of Stakeholders**
  - Architectures are built for stakeholders
  - Decisions must reflect stakeholder needs
  - Involving a wide stakeholder community increases your chances of success
Talking Point: Stakeholders

- Who cares whether your systems get built?
- Why do they care?
- Are they positively or negatively impacted?

Create a list of the stakeholders for your system
Stakeholders

- Attributes of a good stakeholder
  - Informed, to allow them to make good decisions
  - Committed, to the process and willing to make themselves available in a constructive manner, even if decisions are hard
  - Authorised, to make decisions
  - Representative, of their stakeholder group so that they present its views validly
Stakeholder Groups

- Acquirers pay for the system
- Assessors check it for compliance
- Communicators create documents and training
- Developers create it
- Maintainers evolve and fix the system
- Suppliers provide system components
- Support Staff help people to use the system
- System Administrators keep it running
- Testers verify that it works
- Users have to use the system directly
Content

- Defining Software Architecture
- Stakeholders
- The Software Architecture Problem
- Viewpoints to Guide Structure
- Perspectives to Guide Qualities
- Example Application
- Uses for Viewpoints and Perspectives
The Challenge

- Essential difficulties
  - Multi-dimensional problem
  - Highly complex mix of people and technology
  - Diverse stakeholder community to serve
  - Making trade-offs is essential but hard
  - Often no “right” answer
The Challenge

- Accidental difficulties
  - Little standardisation in description
  - Difficult to compare and discuss alternatives
  - Little standardisation in architectural activities
  - Little sharing of proven practice and known problems and their solutions
  - No framework for handling quality properties
Meeting the Challenge

- Organise the architectural design process
  - roles & activities, relationship to requirements & design

- Define the use of architecture artefacts
  - which models? when? why?

- Capture, classify and share knowledge
  - best practice, problems and pitfalls, proven solutions
Content

- Defining Software Architecture
- Stakeholders
- The Software Architecture Problem
- Viewpoints to Guide Structure
- Perspectives to Guide Qualities
- Example Application
- Uses for Viewpoints and Perspectives
Architectural Viewpoints

- Help to deal with architectural structure
- Decompose architectural description into views
  - each view addresses one aspect of the system
  - functional view, deployment view, ...
- Guide development of views via viewpoints
  - viewpoint contains proven practice, pitfalls, ...
  - each view defined by one viewpoint
- Organises the process and the artefacts
Architectural Viewpoints

- Well understood, widely applied, many exist
  - RUP/Kruchten “4+1” set (1995)
  - Siemens set (1999)
  - Rozanski & Woods set (2005)
  - Conceptual basis in IEEE 1471 (2000)
IEEE 1471 provides standard definitions

- A viewpoint is a collection of patterns, templates and conventions for constructing one type of view. It defines the stakeholders whose concerns are reflected in the viewpoint, and guidelines and principles and template models for constructing its views.

- A view is a representation of all or part of an architecture, from the perspective of one or more concerns which are held by one or more of its stakeholders.

Viewpoints and Views

Viewpoint defines View

Architectural Description 1..n
Example Viewpoint Set

- Functional Viewpoint
- Information Viewpoint
- Concurrency Viewpoint
- Development Viewpoint
- Deployment Viewpoint
- Operational Viewpoint

[Rozanski & Woods, 2005]
Example Viewpoint Set

- Core architectural structures
  - Functional
    - elements, connectors, interfaces, responsibilities, interactions
  - Information
    - entities, constraints, relationships, timeliness, usage, ownership
  - Concurrency
    - processes, threads, coordination, element to process mapping
Example Viewpoint Set

- **Building the system**
  - Development
    - layers, module structure, standard design, codeline

- **Moving towards deployment**
  - Deployment
    - hardware, network, software dependencies, process to node mapping
  - Operational
    - installation, migration, administration, support
Example Viewpoint Set

- Rozanski/Woods Viewpoint Set
- Aimed at large scale information systems
- Extension and refinement of the “4+1” set
  - renamed “Logical”, “Process” and “Physical”
  - added “Information” and “Operational”
- Standard content for viewpoints
  - applicability, concerns, models, stakeholders, problems & pitfalls, solutions, checklists
## Functional Viewpoint

<table>
<thead>
<tr>
<th>Focus</th>
<th>functional structure of the system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>design of runtime functional elements and their responsibilities, interfaces, and primary interactions</td>
</tr>
</tbody>
</table>
| Concerns           | - functional capabilities  
                     | - external interfaces  
                     | - internal structure  
                     | - design qualities |
| Models             | - functional structure model |
| Pitfalls           | - poorly defined interfaces / responsibilities  
                     | - infrastructure modelled as functional elements  
                     | - unintentionally overloaded view  
                     | - diagrams without element definitions  
                     | [...] |
Functional View Fragment

element interface and dependent elements using it

UML component represents an element

GUI Client

ClientActions {type=SOAP}
tagged values used to indicate interface characteristics if needed

Statistics Accessor

Statistics Store

ObservationsMgmt
StatsQuery
StatsUpdate

<<external >>
Bulk Loader

stereotype used to indicate external entity
### Information Viewpoint

<table>
<thead>
<tr>
<th>Focus</th>
<th>information structure, ownership and processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>design of storage, manipulation, management, and distribution of information</td>
</tr>
<tr>
<td>Concerns</td>
<td>▪ information structure and content</td>
</tr>
<tr>
<td></td>
<td>▪ information flow</td>
</tr>
<tr>
<td></td>
<td>▪ data ownership and quality</td>
</tr>
<tr>
<td></td>
<td>▪ timeliness, latency, and age</td>
</tr>
<tr>
<td></td>
<td>▪ references and mappings</td>
</tr>
<tr>
<td></td>
<td>▪ transaction management and recovery</td>
</tr>
<tr>
<td></td>
<td>▪ data volumes</td>
</tr>
<tr>
<td></td>
<td>▪ archives and data retention</td>
</tr>
<tr>
<td></td>
<td>▪ regulation</td>
</tr>
</tbody>
</table>
### Information Viewpoint (ii)

<table>
<thead>
<tr>
<th>Models</th>
<th>static data and metadata structure models</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>information flow models</td>
</tr>
<tr>
<td></td>
<td>information lifecycle models</td>
</tr>
<tr>
<td></td>
<td>data ownership and access models</td>
</tr>
<tr>
<td></td>
<td>volumetric models</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pitfalls</th>
<th>data incompatibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>poor data quality</td>
</tr>
<tr>
<td></td>
<td>unavoidable multiple updaters</td>
</tr>
<tr>
<td></td>
<td>key matching deficiencies</td>
</tr>
<tr>
<td></td>
<td>poor information latency</td>
</tr>
<tr>
<td></td>
<td>interface complexity</td>
</tr>
<tr>
<td></td>
<td>inadequate volumetrics</td>
</tr>
</tbody>
</table>
Information View Fragments

Deduction

*  
created  
published  

DerivedMeasure

Deduction

StatsSet

Variable

Observation
Information View Fragments (ii)

<table>
<thead>
<tr>
<th></th>
<th>Observation</th>
<th>Deduction</th>
<th>Derived Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistics Accessor</td>
<td>R</td>
<td>C,R,U,D</td>
<td>R</td>
</tr>
<tr>
<td>Statistics Calculator</td>
<td>-</td>
<td>-</td>
<td>C,U,D</td>
</tr>
<tr>
<td>Bulk Loader</td>
<td>C,U,D</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

![Bar chart showing data for 2005-01 to 2005-04](chart.png)

Legend:
- Observation
- Deduction
- Derived Measure
# Concurrency Viewpoint

<table>
<thead>
<tr>
<th>Focus</th>
<th>packaging elements into processes and threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>the concurrency structure, mapping functional elements to concurrency units to clearly identify the parts of the system that can execute concurrently, and how this is coordinated and controlled</td>
</tr>
<tr>
<td>Concerns</td>
<td>- task structure</td>
</tr>
<tr>
<td></td>
<td>- mapping of functional elements to tasks</td>
</tr>
<tr>
<td></td>
<td>- inter-process communication &amp; re-entrancy</td>
</tr>
<tr>
<td></td>
<td>- state management</td>
</tr>
<tr>
<td></td>
<td>- synchronization and integrity</td>
</tr>
<tr>
<td></td>
<td>- task startup, shutdown and recovery from failure</td>
</tr>
</tbody>
</table>
## Concurrency Viewpoint (ii)

<table>
<thead>
<tr>
<th>Models</th>
<th>system-level concurrency model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>system-level state model</td>
</tr>
<tr>
<td>Pitfalls</td>
<td>modelling of the wrong concurrency</td>
</tr>
<tr>
<td></td>
<td>excessive complexity</td>
</tr>
<tr>
<td></td>
<td>resource contention</td>
</tr>
<tr>
<td></td>
<td>deadlock and race conditions</td>
</tr>
</tbody>
</table>
# Development Viewpoint

<table>
<thead>
<tr>
<th>Focus</th>
<th>architectural constraints on the development process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>architectural design that supports and constraints the software development process</td>
</tr>
</tbody>
</table>
| Concerns | - module organization  
- codeline organization  
- common processing  
- standardization of design and testing  
- instrumentation |
| Models | - module structure models  
- common design models  
- codeline models |
| Pitfalls | - too much detail or lack of precision  
- overburdening the architectural description  
- uneven focus or lack of developer focus  
- problems with the specified environment |
## Deployment Viewpoint

<table>
<thead>
<tr>
<th><strong>Focus</strong></th>
<th>runtime environment structure and the distribution of software across it</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Content</strong></td>
<td>design of the environment into which the system will be deployed, including the system’s runtime dependencies</td>
</tr>
<tr>
<td><strong>Concerns</strong></td>
<td></td>
</tr>
</tbody>
</table>
|  | ▪ types of hardware required  
|  | ▪ specification and quantity of hardware required  
|  | ▪ third-party software requirements  
|  | ▪ technology compatibility  
|  | ▪ network requirements  
|  | ▪ network capacity required  
|  | ▪ physical constraints |
## Deployment Viewpoint (ii)

<table>
<thead>
<tr>
<th>Models</th>
<th>Pitfalls</th>
</tr>
</thead>
<tbody>
<tr>
<td>runtime platform models</td>
<td>unclear or inaccurate dependencies</td>
</tr>
<tr>
<td>network models</td>
<td>unproven technology</td>
</tr>
<tr>
<td>technology dependency models</td>
<td>lack of specialist technical knowledge</td>
</tr>
<tr>
<td></td>
<td>late consideration of the deployment environment</td>
</tr>
<tr>
<td>Deployment View Fragment (ii)</td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Client PC</strong></td>
<td>Windows XP SP1</td>
</tr>
<tr>
<td></td>
<td>Java JRE 1.4.2_06 or later</td>
</tr>
<tr>
<td></td>
<td>Internet Explorer 6.0 SP1</td>
</tr>
<tr>
<td><strong>Primary Server</strong></td>
<td>Windows 2003 server, w/sec patches</td>
</tr>
<tr>
<td></td>
<td>Java SDK 1.4.2_06 or later</td>
</tr>
<tr>
<td></td>
<td>Apache Tomcat 5.5.9 or later</td>
</tr>
<tr>
<td><strong>Database Server</strong></td>
<td>Solaris 9.0 w/Aug05 patch cluster</td>
</tr>
<tr>
<td></td>
<td>Oracle 9.2.0.2 Std Edition</td>
</tr>
<tr>
<td></td>
<td>10GB buffer cache, auto sized SGA</td>
</tr>
<tr>
<td></td>
<td>auto storage management, 2 table spaces</td>
</tr>
<tr>
<td></td>
<td>OEM 9.2.0.2 installed and working</td>
</tr>
</tbody>
</table>
### Operational Viewpoint

<table>
<thead>
<tr>
<th>Focus</th>
<th>System installation, migration, operation &amp; support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>Defines strategies for how the system will be operated, administered, and supported when it is running in its production environment</td>
</tr>
</tbody>
</table>
| Concerns               | - Installation and upgrade  
|                        | - Functional and data migration  
|                        | - Operational monitoring and control  
|                        | - Operational configuration management  
|                        | - Performance monitoring  
|                        | - Support responsibilities and procedures  
|                        | - Backup and restore |
## Operational Viewpoint (ii)

<table>
<thead>
<tr>
<th>Models</th>
<th>Pitfalls</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ installation models</td>
<td>▪ lack of engagement with the operational staff</td>
</tr>
<tr>
<td>▪ migration models</td>
<td>▪ lack of migration &amp; backout planning</td>
</tr>
<tr>
<td>▪ configuration management models</td>
<td>▪ insufficient migration window</td>
</tr>
<tr>
<td>▪ administration models</td>
<td>▪ missing management tools</td>
</tr>
<tr>
<td>▪ support and escalation models</td>
<td>▪ lack of integration into the production environment</td>
</tr>
<tr>
<td></td>
<td>▪ inadequate backup and recovery modelling</td>
</tr>
</tbody>
</table>
Operational View Content

- Installation Model
  - Installation groups
  - Dependencies and constraints
  - Backout strategy

- Operational CM Model
  - Configuration groups and dependencies
  - Configuration parameter sets
  - Operational control (switching between sets)
Operational View Content (ii)

- **Administration Model**
  - Monitoring and control facilities required and provided
  - Required routine operational procedures
  - Required operational action in case of error conditions
Talking Point: Views

- Which views would be useful for your systems?
- Who would be interested in each view?
- Which views wouldn’t be useful – why?

Draw up a list of which views you would use
Who would be interested in each?
Can you sketch fragments of one or two?
Viewpoints and Views Recap

- **Viewpoints**
  - A store of knowledge and experience
  - A guide to the architect
  - Templates to guide the process

- **Views**
  - A structure for description
  - A separation of concerns
  - Aid to stakeholder communication
Limitations of Viewpoints

- Quality properties are critical
  - most viewpoint sets don’t explicitly consider qualities

- Quality properties usually need cross-view consideration
  - viewpoints are relatively independent

- Viewpoint focus may lead to late consideration of quality properties
  - architects focus on desired structures not qualities
  - qualities are often expensive to add later
Content

- Defining Software Architecture
- Stakeholders
- The Software Architecture Problem
- Viewpoints to Guide Structure
- Perspectives to Guide Qualities
- Example Application
- Uses for Viewpoints and Perspectives
Talking Point: Qualities

- Which quality properties are crucial to your systems?
- Which views are going to be impacted by considering each such quality property?

Create a matrix showing which qualities are likely to impact which of your views.
### Possible Qualities to Consider

<table>
<thead>
<tr>
<th>Performance</th>
<th>Geographical distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability</td>
<td>Localisation</td>
</tr>
<tr>
<td>Security</td>
<td>Efficiency</td>
</tr>
<tr>
<td>Availability</td>
<td>Development constraints</td>
</tr>
<tr>
<td>Resilience</td>
<td>Time to deliver</td>
</tr>
<tr>
<td>Evolution</td>
<td>Reliability</td>
</tr>
<tr>
<td>Maintainability</td>
<td>Cost</td>
</tr>
<tr>
<td>Supportability</td>
<td>Safety</td>
</tr>
</tbody>
</table>
Dealing with Quality Properties

- A new concept could help
  - Allowing cross-view focus
  - Being quality rather than structure oriented
  - Providing similar organisation and guidance to a viewpoint but for a quality property
  - That can be used in tandem with viewpoints

- We call this new concept a “perspective”
  - or “architectural perspective” in full
An architectural **perspective** is a collection of **activities, checklists, tactics and guidelines** to guide the process of ensuring that a **system exhibits** a particular set of closely related **quality properties** that require consideration across a number of the system’s architectural views.

Rozanski and Woods, 2005
Architectural Perspectives

- A guide for dealing with quality properties
  - Guide the architect in achieving the required quality properties
  - Suggest changes to the existing views
  - Avoid possible redundancy between quality and structural views

- A new concept to use with viewpoints
  - Related to and extends SEI tactics work
  - Adds more context and advice to tactics
Adding Perspectives

- Viewpoint defines View
- View applied to Perspective results in Supporting Artefact

Architectural Description connects Viewpoint to View with 1..n relationship.
Architectural Perspectives

- A simple but effective idea
  - A store of knowledge and experience
  - A guide to the architect based on proven practice
  - Templates to guide the process

- Analogous to viewpoints but for quality properties, rather than structures

- Perspectives “applied” to views to assess qualities and guide changes needed
## Perspectives vs. Viewpoints

<table>
<thead>
<tr>
<th></th>
<th>Perspective</th>
<th>Viewpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Focus</strong></td>
<td>A quality property</td>
<td>A type of structure</td>
</tr>
<tr>
<td><strong>Result</strong></td>
<td>Changes to views</td>
<td>A view – model(s)</td>
</tr>
<tr>
<td></td>
<td>Supporting artefacts</td>
<td>A primary arch structure</td>
</tr>
<tr>
<td><strong>Guidance</strong></td>
<td>A process for application</td>
<td>Models to create</td>
</tr>
<tr>
<td></td>
<td>Advice based on practice</td>
<td>Advice based on practice</td>
</tr>
<tr>
<td><strong>Versus Views</strong></td>
<td>1 perspective : n views</td>
<td>1 viewpoint : 1 view</td>
</tr>
</tbody>
</table>
Perspectives and Views

- Security Perspective
- Accessibility Perspective
- Performance Perspective
- Location Perspective
- Availability Perspective
- Regulation Perspective
- Maintenance Perspective
- etc.

Functional View
Development View
Information View
Deployment View
Information View
Operational View

Stakeholders → Architecture
Architectural Perspectives

- Our initial core set for information systems
  - Performance and Scalability
  - Security
  - Availability and Resilience
  - Evolution
  - Also: Location, I18N, Usability, Regulation, ...

- Different sets in different domains
## Performance and Scalability

<table>
<thead>
<tr>
<th>Quality</th>
<th>ability of the system to predictably execute within its mandated performance profile and to handle increased processing volumes</th>
</tr>
</thead>
</table>
| Concerns | processing volume  
|          | response time  
|          | responsiveness  
|          | throughput  
|          | predictability |
| Tactics | optimize repeated processing  
|         | reduce contention via replication  
|         | prioritize processing  
|         | consolidate related workloads  
|         | distribute processing over time  
|         | [...] |
### Performance and Scalability (ii)

| Tactics (cont.) | • distribute processing over time  
|                | • minimize the use of shared resources  
|                | • partition and parallelize  
|                | • use asynchronous processing  
|                | • make design compromises |

| Pitfalls | • imprecise performance and scalability goals  
|          | • unrealistic models  
|          | • use of simple measures for complex cases  
|          | • inappropriate partitioning  
|          | • invalid environment and platform assumptions  
|          | • too much indirection  
|          | • concurrency-related contention  
|          | • careless allocation of resources  
|          | • disregard for network and in-process differences |
P & S Perspective Activities

1. Capture Performance Requirements

2. Create Performance Models

3A. Analyze Performance Model

3B. Practical Testing

4. Assess Against Requirements

5. Rework Architecture

[acceptable]
## Security

<table>
<thead>
<tr>
<th>Quality</th>
<th>ability of the system to reliably control, monitor, and audit who can perform actions on resources and to detect and recover from security failures</th>
</tr>
</thead>
</table>
| Concerns | - policies  
- threats  
- mechanisms  
- accountability  
- availability  
- detection and recovery |
| Tactics | - apply recognized security principles  
- authenticate the principals  
- authorize access  
- ensure information secrecy  
- ensure information integrity  
- [...] |
| Tactics (cont.) | vulnerability analysis  
|                | application of security technology |
| Pitfalls       | no clear requirements or models  
|                | complex security policies  
|                | unproven or ad-hoc security technologies  
|                | not designing for failure  
|                | lack of security administration facilities  
|                | technology-driven approach (or over-reliance)  
|                | failure to consider time sources  
|                | security as an afterthought  
|                | security embedded in the application code  
|                | piecemeal security  

Security Perspective Activities

1. Identify Sensitive Resources → 2. Define Security Policy

3. Identify Threats to the System


[unacceptable]

[acceptable]
# Availability and Resilience

<table>
<thead>
<tr>
<th>Quality</th>
<th>Ability of the system to be fully or partly operational as and when required and to effectively handle failures that could affect system availability</th>
</tr>
</thead>
</table>
| **Concerns**                                                            | classes of service  
|                                                                         | planned / unplanned downtime  
|                                                                         | mean time between failures & mean time to repair  
|                                                                         | disaster recovery  
|                                                                         | redundancy, clustering, failover  
| **Tactics**                                                             | select fault-tolerant hardware  
|                                                                         | use hardware clustering and load balancing  
|                                                                         | log transactions  
|                                                                         | apply software availability solutions  
|                                                                         | select or create fault-tolerant software  
|                                                                         | identify backup and disaster recovery solutions  

<table>
<thead>
<tr>
<th>Pitfalls</th>
</tr>
</thead>
<tbody>
<tr>
<td>single point of failure</td>
</tr>
<tr>
<td>overambitious availability requirements</td>
</tr>
<tr>
<td>ineffective error detection</td>
</tr>
<tr>
<td>overlooked global availability requirements</td>
</tr>
<tr>
<td>incompatible technologies</td>
</tr>
</tbody>
</table>
A & R Perspective Activities

1. Capture Availability Requirements
2. Produce Availability Schedule
3A. Estimate Platform Availability
3B. Estimate Functional Availability
4. Assess Against Requirements
5. Rework Architecture

[ finished ] [ not finished ]
## Evolution

<table>
<thead>
<tr>
<th>Quality</th>
<th>ability of the system to be flexible in the face of the change that all systems experience, balanced against the costs of providing such flexibility</th>
</tr>
</thead>
</table>
| Concerns | flexibility  
  extensibility  
  functional evolution  
  deployment evolution  
  integration evolution |
| Tactics | contain change  
  create flexible interfaces  
  apply change-oriented architectural styles  
  build variation points into the software  
  use standard extension points  
  achieve reliable change  
  preserve development environments [...] |
## Evolution (ii)

<table>
<thead>
<tr>
<th>Tactics (cont.)</th>
<th>configuration management</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>automated testing</td>
</tr>
<tr>
<td></td>
<td>build and release management</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pitfalls</th>
<th>prioritization of the wrong dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>changes that never happen</td>
</tr>
<tr>
<td></td>
<td>impact of evolution on critical quality properties</td>
</tr>
<tr>
<td></td>
<td>lost development environments</td>
</tr>
<tr>
<td></td>
<td>ad hoc release management</td>
</tr>
</tbody>
</table>
Evolution Perspective Activities

1. Characterise Evolution Needs

2. Assess Current Ease of Evolution

3. Consider Evolution Tradeoffs

4. Rework Architecture

[ finished ]

[ not finished ]
### Other Perspectives

<table>
<thead>
<tr>
<th>Perspective</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessibility</td>
<td>Can the system be used by people with disabilities?</td>
</tr>
<tr>
<td>Development Resource</td>
<td>Can the system be built within people, time and budget constraints?</td>
</tr>
<tr>
<td>Internationalisation</td>
<td>Is the system independent of language, country and culture?</td>
</tr>
<tr>
<td>Location</td>
<td>Will the system work, given its required geographical constraints?</td>
</tr>
<tr>
<td>Regulation</td>
<td>Does the system meet any required regulatory constraints?</td>
</tr>
<tr>
<td>Usability</td>
<td>Can people use the system effectively?</td>
</tr>
</tbody>
</table>
Talking Point: Perspectives

- Going back to your qualities/views matrix
- Which perspectives would help you to achieve your quality properties?
  - Are they in the primary or secondary sets?
  - Useful feedback for us if in secondary
- Where may you have conflicts between advice in different relevant perspectives?
Talking Point: Perspectives

- Decide on the most important qualities for the system you’ve been considering
  - Top 3

- For the most important (or interesting) property identify the likely impact of applying its perspective
  - How does achieving that quality affect the architecture?
Content

- Defining Software Architecture
- Stakeholders
- The Software Architecture Problem
- Viewpoints to Guide Structure
- Perspectives to Guide Qualities
- Example Application
- Uses for Viewpoints and Perspectives
Example Application

- Simple example of viewpoints and perspectives
- Used throughout the tutorial materials
- Statistics storage and processing system
  - Data loaded into the database
  - Derived measures calculated automatically
  - Statisticians view and report on the data
  - Deductions recorded and reviewed manually
Recap: Functional View

- **GUI Client**: element interface and dependent elements using it
- **Statistics Accessor**: UML component represents an element
  - ClientActions {type=SOAP}
  - Statistics Calculator
    - tagged values used to indicate interface characteristics if needed
  - Statistics Store
    - ObservationsMgmt
    - StatsQuery
    - StatsUpdate
    - BulkUpdate
  - <<external >>
    - Bulk Loader
    - stereotype used to indicate external entity
Recap: Concurrency View
Recap: Development View

- **Domain**
  - StatDate Library
  - Java Numerical Toolkit

- **Utility**
  - Apache Axis
  - Hibernate 2.1

- **Platform**
  - Java 1.4 Library
  - Oracle JDBC Driver 9.0
  - Servlet 2.2 API
Recap: Deployment View

Data Centre Resident

Primary Server
- Stats_Server
- Calculator

Client PC
- Memory >= 500MB, CPU >= 1.8GHz

Database Server
- DBMS_Process_Grp
- Loader
- Disk Array
  - Model = StorEdge 3510 FC, Capacity = 500GB

Database Server
- Model = SunFire V 440
  - Memory = 16GB, CPU = 2x1.6GHz
  - IO = FiberChannel
## Recap: Deployment View (ii)

<table>
<thead>
<tr>
<th>Component</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client PC</strong></td>
<td>- Windows XP SP1&lt;br&gt;- Java JRE 1.4.2_06 or later&lt;br&gt;- Internet Explorer 6.0 SP1</td>
</tr>
<tr>
<td><strong>Primary Server</strong></td>
<td>- Windows 2003 server, w/sec patches&lt;br&gt;- Java SDK 1.4.2_06 or later&lt;br&gt;- Apache Tomcat 5.5.9 or later</td>
</tr>
</tbody>
</table>
| **Database Server** | - Solaris 9.0 w/Aug05 patch cluster<br>- Oracle 9.2.0.2 Std Edition<br>  
  - 10GB buffer cache, auto sized SGA<br>  
  - auto storage management, 2 table spaces<br>- OEM 9.2.0.2 installed and working |
Recap: Operational View

- Omitted from slides for space reasons.
- Would include:
  - Operational CM approach
  - Monitoring and Control
  - Operational Needs
  - Installation / migration / backout strategies
Example: Applying Perspectives

- Performance and Scalability
  - Capture P & S Requirements
  - Create Performance Models
  - Analyse Models
  - Perform Practical Testing
  - Assess Against Requirements
  - Rework Architecture (apply tactics)

- What affect will this have on our system?
Example: Applying Perspectives

1. Capture Performance Requirements
2. Create Performance Models
3A. Analyze Performance Model
3B. Practical Testing
4. Assess Against Requirements
5. Rework Architecture

[acceptable]
Example: Applying Perspectives

<table>
<thead>
<tr>
<th>Measure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network A</td>
<td>100Mb</td>
</tr>
<tr>
<td>DB access (ro)</td>
<td>20ms</td>
</tr>
<tr>
<td>DB access (rw)</td>
<td>50ms</td>
</tr>
<tr>
<td>Single derived calc</td>
<td>1400ms</td>
</tr>
<tr>
<td>Client network</td>
<td>10Mb</td>
</tr>
<tr>
<td>Bulk load 100K</td>
<td>2500ms</td>
</tr>
<tr>
<td>Online load users</td>
<td>100</td>
</tr>
<tr>
<td>Memory per user</td>
<td>1MB</td>
</tr>
</tbody>
</table>

Calibration measures

Performance model
Example: Applying Perspectives

- Security
  - Identify Sensitive Resources
  - Define Security Policy
  - Identify Threats to the System
  - Design Security Implementation (apply tactics)
  - Assess Security Risks

- What affect will this have on our system?
Example: Applying Perspectives

1. Identify Sensitive Resources
2. Define Security Policy
3. Identify Threats to the System
4. Design Security Implementation
5. Assess Security Risks

[unacceptable]

[acceptable]
Example: Applying Perspectives

- Sensitive Resources
  - The data in the database

- Security Threats
  - Operators stealing backups
  - Administrators querying data, seeing names
  - Bribing investigating officers
  - Internal attack on the database via network
Example: Applying Perspectives

- **Security Countermeasures**
  - Backups: encrypt data in the database
    - How about performance?
    - Does this make availability (DR) harder?
  - Hiding names: use codes instead of names, protect the underlying names at a higher security level
    - More development complexity
    - Possible performance impact
Example: Applying Perspectives

- **Security Countermeasures**
  - Bribery: add audit trail for data access
    - Possible performance impact
    - More complexity
    - Protecting / using the audit trail
  - Network Attacks: harden database, firewalls, IDS
    - More deployment & administrative complexity
    - More hardware and operational cost
    - Operational impact if IDS trips
Example: Applying Perspectives

Impact on information view ...

```
DerivedMeasure

Deduction

StatsSet

Variable

Identifier Code

Observation

Isolate names
```
Example: Applying Perspectives

Impact on development view ...

- Controlled StatAccess Library
- StatDate Library
- Java Numerical Toolkit

Add audit when accessing data

- Apache Axis
- Hibernate 2.1
Example: Applying Perspectives

Impact on deployment view ...

- Added network model making network security clear
Example: Applying Perspectives

- Other Impact
  - Need IDS added to Development view
  - Database security
  - Need to capture impact on Operational view
  - Need to consider impact on availability
  - Need to re-work performance models to allow for database encryption, audit, ...

- Note the need to change many views to address security needs
Content

- Defining Software Architecture
- Stakeholders
- The Software Architecture Problem
- Viewpoints to Guide Structure
- Perspectives to Guide Qualities
- Example Application
- Uses for Viewpoints and Perspectives
Using Viewpoints & Perspectives

- A framework for organising work
- A store of knowledge
  - Document proven practice
  - Help standardise language and approach
  - Help to standardise languages and approaches
- Applicable at different career stages
  - Mentor novice architects
  - Guide working architects
  - Support expert architects
Using Viewpoints & Perspectives

- For Novice Architects
  - An introduction to each area of knowledge
  - A guide to what is important
  - A structure for the process
  - Definitions of standards and norms
  - Repository of proven practice and tactics
  - Pitfalls and solutions to avoid common errors
  - Checklist to ensure nothing is forgotten
Using Viewpoints & Perspectives

- For Working Architects
  - A reminder of what is important
  - A guide to new or rarely used areas of practice
  - Repository of proven practice and tactics
  - Pitfalls and solutions to avoid common errors
  - Checklist to ensure nothing is forgotten
Using Viewpoints & Perspectives

- For Expert Architects
  - A framework to allow knowledge sharing
  - An aid to tutoring and mentoring
  - Checklists to ensure nothing is forgotten
Summary

- Architecture Essential Difficulties
  - Multi-dimensional problem, no right answer
  - Stakeholder needs conflict
  - Complex mix of people and technology
  - Tradeoffs are inevitable

- Architecture Accidental Difficulties
  - Lack of standardisation (approach, artefacts)
  - Little sharing of knowledge and experience
Summary (ii)

- **Viewpoints and Views**
  - Views provide a convenient approach for effective architectural description
  - Viewpoints standardise views by defining their content
  - Viewpoints contain proven architectural knowledge for a particular domain
  - Viewpoints and views can address many accidental difficulties of software architecture
Summary (iii)

- Viewpoints for Information Systems
  - Functional
  - Information
  - Concurrency
  - Development
  - Deployment
  - Operational
Summary (iv)

- Perspectives
  - Viewpoints handle structure well, less convinced about quality properties
  - Perspectives provide similar guidance and knowledge sharing for quality properties
  - Perspectives suggest the design activities required to achieve a property
  - Perspectives provide proven practice, pitfalls, solutions and checklists to share experience
  - Applying perspectives modifies views
Summary (v)

- Perspectives for Information Systems
  - Availability and Resilience
  - Evolution
  - Performance and Scalability
  - Security
  - Others
    - Accessibility, Development Resource, Geographical Location, I18N, Regulation, Usability
Summary (iv)

- Using Viewpoints and Perspectives
  - Novices
    - Overview, guides, focuses attention
    - Proven practice, pitfalls, solutions and checklists
  - Working Architects
    - Reminder of existing knowledge
    - Aid in new areas
  - Experts
    - Mentoring and communication vehicle
    - Reminders of hard won lessons
To Learn More

Software Systems Architecture
Working With Stakeholders Using Viewpoints and Perspectives

Nick Rozanski & Eoin Woods
Addison Wesley, 2005

http://www.viewpoints-and-perspectives.info
Eoin Woods
contact@eoinwoods.info
www.eoinwoods.info

Comments and Questions?

Nick Rozanski
nick@rozanski.com
www.nick.rozanski.com