Application Architecture Guide 2.0
Project Overview

“How to put the Legos together”
Elevator Pitches

“How to put the Legos together” …

“… Microsoft playbook for app architecture”
Vision

- **A story that customers get.** A story around MS for how to put the platform Legos together from an application architecture standpoint.

- **Platform playbook for building apps.** A thin guide that frames out the application architecture space and maps relevant principles, patterns, and practices for application types, layers, quality attributes and technologies.

- **KB of App Arch Nuggets.** A thick, browsable knowledge base (KB) of guidelines, how tos, checklists, patterns, videos … etc.

- **Successful Customers.** Solution Architects, developer leads, and developers are confident and competent building applications on the .NET platform. Customers using J2EE / competitive platforms can build effective solutions on the .NET platform.
Key Features of the Guide

- **Canonical app frame** - describes at a meta-level, the tiers and layers that an architect should consider. Each tier/layer will be described in terms of its focus, function, capabilities, common design patterns and technologies.

- **App Types** - 5-7 canonical application archetypes to illustrate common application types. Each archetype will be described in terms of the target scenarios, technologies, patterns and infrastructure it contains. Each archetype will be mapped to the canonical app frame. They are illustrative of common app types and not comprehensive or definitive.

- **Arch Frame** - a common set of categories for hot spots for key engineering decisions.

- **Quality Attributes** - a set of qualities/abilities that shape your application architecture: performance, security, scalability, manageability, deployment, communication, etc.

- **Principles, patterns and practices** - using the frames as backdrops, the guide will overlay relevant principles, patterns, and practices.

- **Technologies and capabilities** - a description/overview of the Microsoft custom app dev platform and the main technologies and capabilities within it.
Key Scenarios for the Guide

- Help you choose the right architecture for your application.
- Help you choose the right technologies
- Help you make more effective choices for key engineering decisions.
- Help you map appropriate strategies and patterns.
- Help you map relevant patterns & practices solution assets.
Execution
At a Glance

- **START:** 07/15/08
- **END:** 01/15/09

Rhythm

- 2 Week Releases (Drafts / Modules)

Approach

- Time-boxed results
- Incremental value
- Separation of risk (project focus)
From KB to Guide to MSDN

KB (CodePlex)
- Reusable nuggets
- Community KB
- Customer Test / Vette

Guide
- Story
- Frames
- Principles, patterns, practices

MSDN
- Full catalog
- Guide
- Fit and finish
Guide / KB

Guide

- Chapters
- Frames
- Principles
- Patterns
- Guidelines

KB / Nuggets

- App Patterns
- App Scenarios
- Cheat Sheets
- Checklists
- Code Examples

- Explained
- Guidelines
- How Tos
- Patterns
- Videos
Modules (Conceptual Model)

Guides
- Stable
- What to Do/Why
- High-level How

Nuggets
- Volatile
- Magnifying lens
- Show me how
Appendix
Conceptual Framework

![Conceptual Framework Diagram]

**App Types**
- Mobile
- RIA
- Rich Client
- Service
- Web Application
- N ...

**Architecture Styles**
- Client Server
- Composite Application
- ESB
- Grid
- N-Tier
- Peer-To-Peer
- SaaS/S+S
- SOA
- N ...

**Architecture Frame**
- Caching
- Communication
- Concurrency and Transactions
- Configuration Management
- Coupling and Cohesion
- Data Access
- Exception Management
- Layering
- Logging and Instrumentation
- State Management
- Structure
- Validation
- Workflow
Key Trends
Key Trends / Hot Spots

<table>
<thead>
<tr>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Business Process Management (BPM)</td>
</tr>
<tr>
<td>• Composite / Mash Ups (Server-side, Client-side)</td>
</tr>
<tr>
<td>• Dynamic Languages</td>
</tr>
<tr>
<td>• Functional Programming</td>
</tr>
<tr>
<td>• Health</td>
</tr>
<tr>
<td>• Model-Driven</td>
</tr>
<tr>
<td>• Representational State Transfer (REST)</td>
</tr>
<tr>
<td>• Software plus Services / Software as a Service / Platform as a Service (S+S / SaaS / PaaS)</td>
</tr>
<tr>
<td>• Service Oriented Architecture (SOA)</td>
</tr>
<tr>
<td>• Rich Internet Applications (RIA)</td>
</tr>
<tr>
<td>• Testability</td>
</tr>
<tr>
<td>• User Empowerment (shift in power from business and tech to the user)</td>
</tr>
<tr>
<td>• User Experience (not to be confused with UI)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Infrastructure</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Cloud Computing</td>
</tr>
<tr>
<td>• Green IT</td>
</tr>
<tr>
<td>• Virtualization</td>
</tr>
<tr>
<td>• Very Large Databases</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Grid</td>
</tr>
<tr>
<td>• High Performance Computing (HPC)</td>
</tr>
<tr>
<td>• Many-core / Multi-core</td>
</tr>
<tr>
<td>• Parallel Computing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Application Life-Cycle Management (ALM)</td>
</tr>
<tr>
<td>• Distributed Teams</td>
</tr>
<tr>
<td>• Lean</td>
</tr>
<tr>
<td>• Scrum</td>
</tr>
<tr>
<td>• User-Lead</td>
</tr>
<tr>
<td>• XP</td>
</tr>
</tbody>
</table>
Arch Styles
Arch Styles

- Client/Server
- Component-Based Architecture
- Data Centered
- Distributed Architecture
- Layered Architecture
- Object Oriented
- P2P/Grid
- S+S / SaaS / PaaS
- SOA
- REST
App Types (Archetypes)

Not Exhaustive, Just Illustrative
Mobile
Rich Internet Application (RIA)
Rich Client

- **Presentation**
  - UI Components
  - UI Process Components

- **Business**
  - Application Facade
  - Business Workflows
  - Business Components
  - Business Entities

- **Data**
  - Data Access Components
  - Data Helpers / Utilities
  - Service Agents

- **Cross-Cutting**
  - Security
  - Operational Management
  - Communication

- **Local Data**
  - External Data
  - Services
Layers, Components, Tiers
Presentation, Business, Data
Tiers (2-Tier, 3-Tier, N-Tier)
Layers / Components

- Presentation
  - UI Components
  - UI Process Components

- Business
  - Application Facade
    - Business Workflows
    - Business Components
    - Business Entities

- Data
  - Data Access Components
  - Data Helpers / Utilities
  - Service Agents

- Cross-Cutting
  - Security
  - Operational Management
  - Comunication

Data Sources

Services
Services Layer

- External Systems
  - Service Consumers
- Services
  - Service Interfaces
  - Message Types
- Business
  - Application Facade
    - Business Workflows
    - Business Components
    - Business Entities
- Data
  - Data Access Components
  - Data Helpers / Utilities
  - Service Agents
- Cross-Cutting
  - Security
  - Operational Management
  - Communication
Arch Frame
# Architecture Frame

## Categories
- Authentication and Authorization
- Caching and State
- Communication
- Composition
- Concurrency and Transactions
- Configuration Management
- Coupling and Cohesion
- Data Access
- Exception Management
- Logging and Instrumentation
- User Experience
- Validation
- Workflow

## Areas

<table>
<thead>
<tr>
<th>Areas</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication and Authorization</td>
<td>How to choose an authentication strategy.</td>
</tr>
<tr>
<td></td>
<td>How to choose an authorization strategy.</td>
</tr>
<tr>
<td></td>
<td>How to flow identity across layers and tiers.</td>
</tr>
<tr>
<td></td>
<td>How to store user identities when not using Active Directory.</td>
</tr>
<tr>
<td>Caching and State</td>
<td>How to choose an appropriate caching technology, such as using the .NET Framework cache support or custom caching.</td>
</tr>
<tr>
<td></td>
<td>How to determine what data to cache.</td>
</tr>
<tr>
<td></td>
<td>How to determine where to cache the data.</td>
</tr>
<tr>
<td></td>
<td>How to determine the expiration policy.</td>
</tr>
<tr>
<td></td>
<td>How to synchronize caches across a farm.</td>
</tr>
<tr>
<td></td>
<td>How to choose between reactive and proactive cache loading.</td>
</tr>
<tr>
<td></td>
<td>How to identify state data versus cache data.</td>
</tr>
<tr>
<td>Communication</td>
<td>How to determine scope requirements for state data, which also determines where state data is persisted.</td>
</tr>
<tr>
<td></td>
<td>How to choose appropriate protocols for communication across layers and tiers.</td>
</tr>
<tr>
<td></td>
<td>How to design loose coupling across layers.</td>
</tr>
<tr>
<td></td>
<td>How to design an interface for communication across AppDomain, process, and physical boundaries.</td>
</tr>
<tr>
<td></td>
<td>How to pass data across AppDomain, process, and physical boundaries.</td>
</tr>
<tr>
<td></td>
<td>How to perform asynchronous communication.</td>
</tr>
<tr>
<td></td>
<td>How to pass sensitive data.</td>
</tr>
<tr>
<td>Composition</td>
<td>How to choose a composition pattern for the user interface (UI).</td>
</tr>
<tr>
<td></td>
<td>How to avoid dependencies between modules in the UI.</td>
</tr>
<tr>
<td></td>
<td>How to handle communication between modules in the UI.</td>
</tr>
<tr>
<td>Concurrency and Transactions</td>
<td>How to handle concurrency between threads.</td>
</tr>
<tr>
<td></td>
<td>How to choose between optimistic and pessimistic concurrency.</td>
</tr>
<tr>
<td></td>
<td>How to handle distributed transactions.</td>
</tr>
<tr>
<td></td>
<td>How to handle long running transactions.</td>
</tr>
<tr>
<td></td>
<td>How to determine appropriate transaction isolation levels.</td>
</tr>
<tr>
<td></td>
<td>How to determine when compensating transactions are required.</td>
</tr>
<tr>
<td>Configuration Management</td>
<td>How to determine what information needs to be configurable.</td>
</tr>
<tr>
<td></td>
<td>How to determine where and how to store configuration information.</td>
</tr>
<tr>
<td></td>
<td>How to protect sensitive configuration information.</td>
</tr>
<tr>
<td></td>
<td>How to handle configuration information in a farm/cluster.</td>
</tr>
<tr>
<td>Coupling and Cohesion</td>
<td>How to choose an appropriate layering strategy for separation of concerns.</td>
</tr>
<tr>
<td></td>
<td>How to design highly cohesive components and group them within layers.</td>
</tr>
<tr>
<td></td>
<td>How to determine when loose coupling is appropriate between components within a layer.</td>
</tr>
<tr>
<td>Data Access</td>
<td>How to manage database connections.</td>
</tr>
</tbody>
</table>
Quality Attributes
Quality Attribute Frame

Categories

- Availability
- Conceptual Integrity
- Flexibility
- Interoperability
- Maintainability
- Manageability
- Performance
- Reliability
- Reusability
- Scalability
- Security
- Supportability
- Testability
- User Experience / Usability
<table>
<thead>
<tr>
<th>Activities</th>
<th>Core</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td></td>
<td>Security Objectives</td>
</tr>
<tr>
<td>Requirements and Analysis</td>
<td>Functional Requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Non Functional Requirements</td>
<td>Security Design Guidelines</td>
</tr>
<tr>
<td></td>
<td>Technology Requirements</td>
<td>Threat Modeling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Security Design Inspection</td>
</tr>
<tr>
<td>Architecture and Design</td>
<td>Design Guidelines</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Architecture and Design Review</td>
<td></td>
</tr>
<tr>
<td>Development</td>
<td>Unit Tests</td>
<td>Security Code Review</td>
</tr>
<tr>
<td></td>
<td>Code Review</td>
<td>Security Code Review</td>
</tr>
<tr>
<td></td>
<td>Daily Builds</td>
<td>Security Code Review</td>
</tr>
<tr>
<td>Testing</td>
<td>Integration Testing</td>
<td>Security Testing</td>
</tr>
<tr>
<td></td>
<td>System Testing</td>
<td>Security Testing</td>
</tr>
<tr>
<td>Deployment</td>
<td>Deployment Review</td>
<td>Security Deployment Inspection</td>
</tr>
<tr>
<td>Maintenance</td>
<td></td>
<td>Security Deployment Inspection</td>
</tr>
</tbody>
</table>

patterns & practices Security Engineering
<table>
<thead>
<tr>
<th>Activities</th>
<th>Core</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td></td>
<td>Performance Objectives</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Budgeting</td>
</tr>
<tr>
<td>Requirements and Analysis</td>
<td>Functional Requirements</td>
<td>Performance Design Guidelines</td>
</tr>
<tr>
<td></td>
<td>Non Functional Requirements</td>
<td>Performance Modeling</td>
</tr>
<tr>
<td></td>
<td>Technology Requirements</td>
<td>Performance Design Inspection</td>
</tr>
<tr>
<td>Architecture and Design</td>
<td>Design Guidelines</td>
<td>Performance Code Inspection</td>
</tr>
<tr>
<td></td>
<td>Architecture and Design Review</td>
<td>Performance Testing</td>
</tr>
<tr>
<td>Development</td>
<td>Unit Tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Code Review</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Daily Builds</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>Integration Testing</td>
<td>Deployment Inspection</td>
</tr>
<tr>
<td></td>
<td>System Testing</td>
<td>Performance Health Metrics</td>
</tr>
<tr>
<td>Deployment</td>
<td>Deployment Review</td>
<td></td>
</tr>
<tr>
<td>Maintenance</td>
<td></td>
<td>Capacity Planning</td>
</tr>
</tbody>
</table>
Deployment Patterns
Deployment Patterns

Non-Distributed

Web Server
Presentation
Business
Data

Database Server

Distributed

Web Server
Presentation

Application Server
Business
Data

Database Server

Web/App Farms

Web/Application Server
Presentation
Business
Data Access

Web Farm

Database Server
The End
Follow Along At …

- App Arch Guide 2.0 Project Site (CodePlex) – http://www.codeplex.com/AppArch
- Patterns & practices Home – http://msdn.com/practices