Application Architecture Guide 2.0 Project Overview

"How to put the Legos together"

J.D. Meier
Alex Homer
Jason Taylor
Prashant Bansode
Lonnie Wall
Rob Boucher
Akshay Bogawat

09/25/08

patterns & practices
Proven practices for predictable results

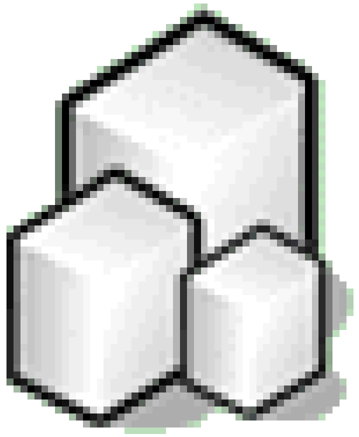
Elevator Pitches

“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.



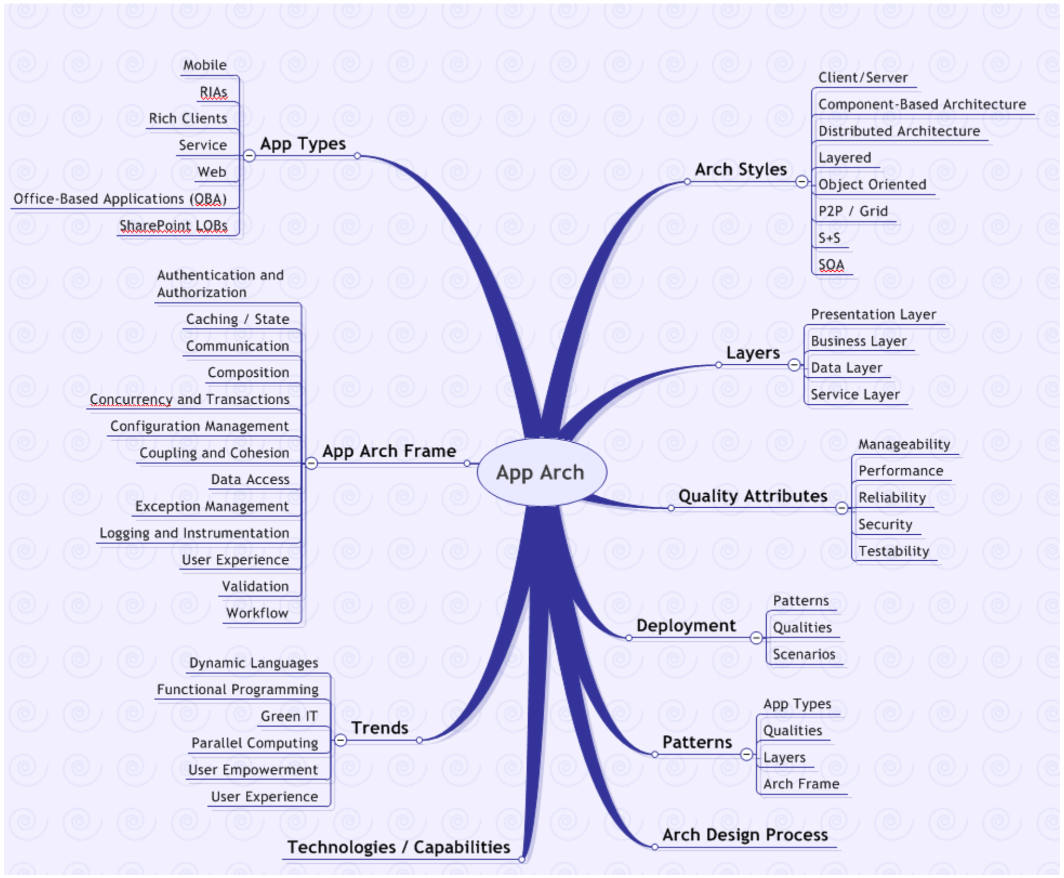
Key Features of the Guide

- **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

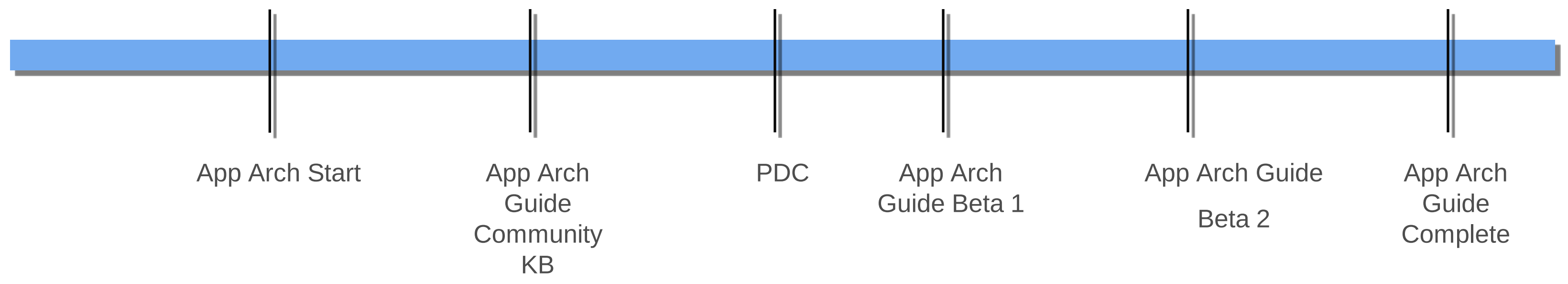
- Help you choose the right architecture for your application.
- Help you choose the right technologies
- Help you make more effective choices for key engineering decisions.
- Help you map appropriate strategies and patterns.
- Help you map relevant patterns & practices solution assets.

Topology Map



Execution

Schedule



App Arch	Date
Start	08/01/08
Codeplex KB	08/30/08
App Arch Guide Beta 1	11/15/08
App Arch Guide Beta 2	12/15/08
App Arch Guide Final (PDF)	01/15/09

At a Glance

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

Rhythm

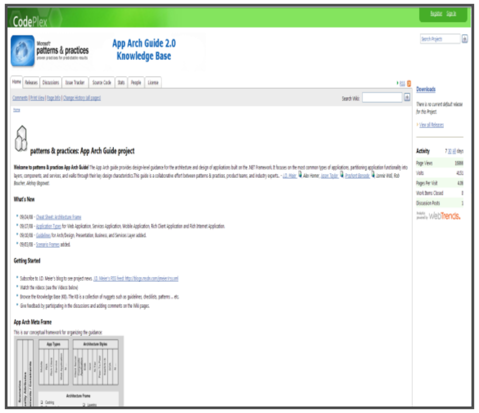
- 2 Week Releases (Drafts / Modules)

Approach

- Time-boxed results
- Incremental value
- Separation of risk (project focus)

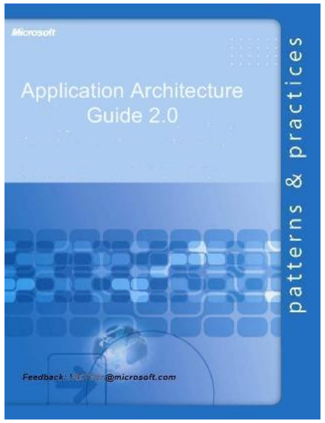
From KB to Guide to MSDN

KB (CodePlex)


CodePlex
patterns & practices
App Arch Guide 2.0 Knowledge Base
patterns & practices: App Arch Guide project
What's New
Getting Started
App Arch Meta Frame

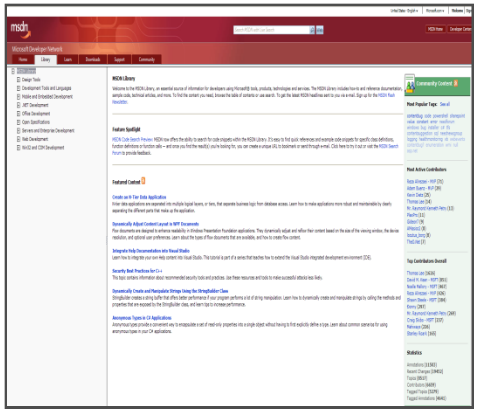
- •Reusable nuggets
- •Community KB
- •Customer Test / Vette

Guide



- •Story
- •Frames
- •Principles, patterns, practices

MSDN



- •Full catalog
- •Guide
- •Fit and finish

Guide / KB

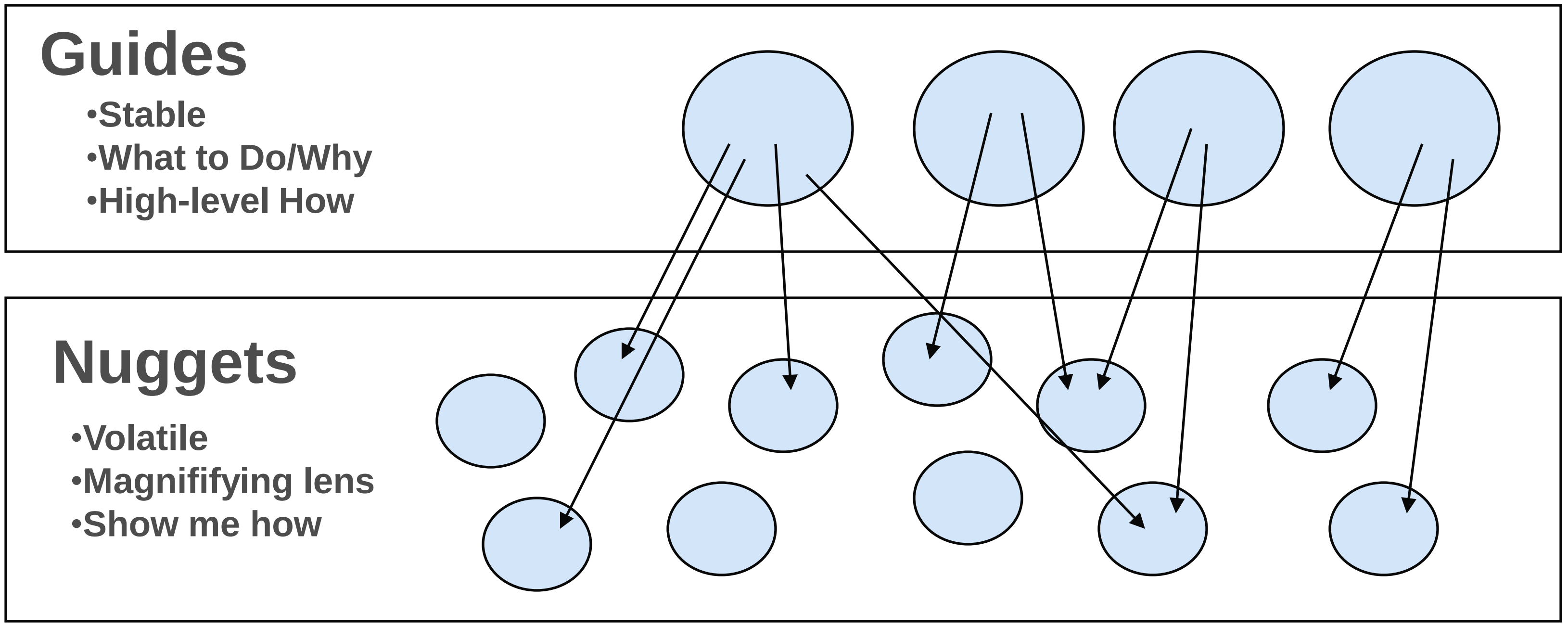
Guide

- 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

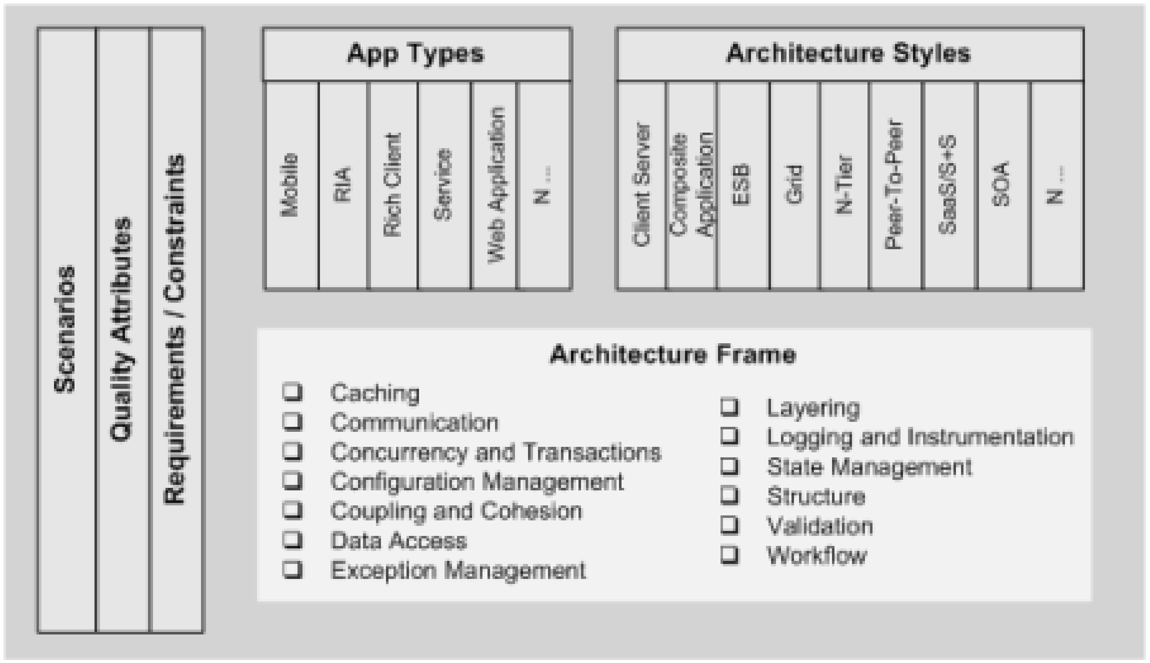
- Stable
- What to Do/Why
- High-level How

Nuggets

- Volatile
- Magnififying lens
- Show me how

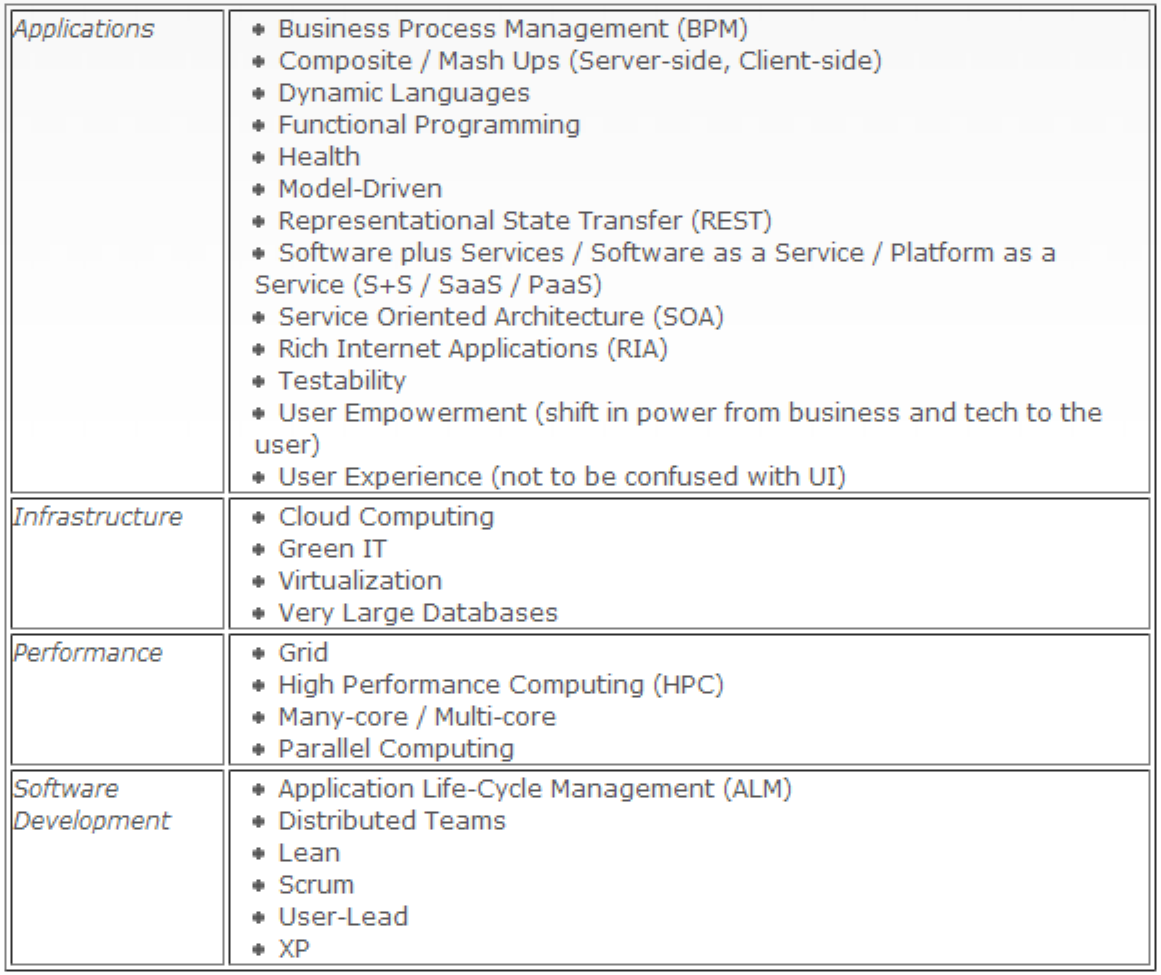
Appendix

Conceptual Framework



Key Trends

Key Trends / Hot Spots



Applications	• Business Process Management (BPM)
	• Composite / Mash Ups (Server-side, Client-side)
	• Dynamic Languages
	• Functional Programming
	• Health
	• Model-Driven
	• Representational State Transfer (REST)
	• Software plus Services / Software as a Service / Platform as a Service (S+S / SaaS / PaaS)
	• Service Oriented Architecture (SOA)
	• Rich Internet Applications (RIA)
	• Testability
	• User Empowerment (shift in power from business and tech to the user)
	• User Experience (not to be confused with UI)
Infrastructure	• Cloud Computing
	• Green IT
	• Virtualization
	• Very Large Databases
Performance	• Grid
	• High Performance Computing (HPC)
	• Many-core / Multi-core
	• Parallel Computing
Software Development	• Application Life-Cycle Management (ALM)
	• Distributed Teams
	• Lean
	• Scrum
	• User-Lead
	• XP

Arch Styles

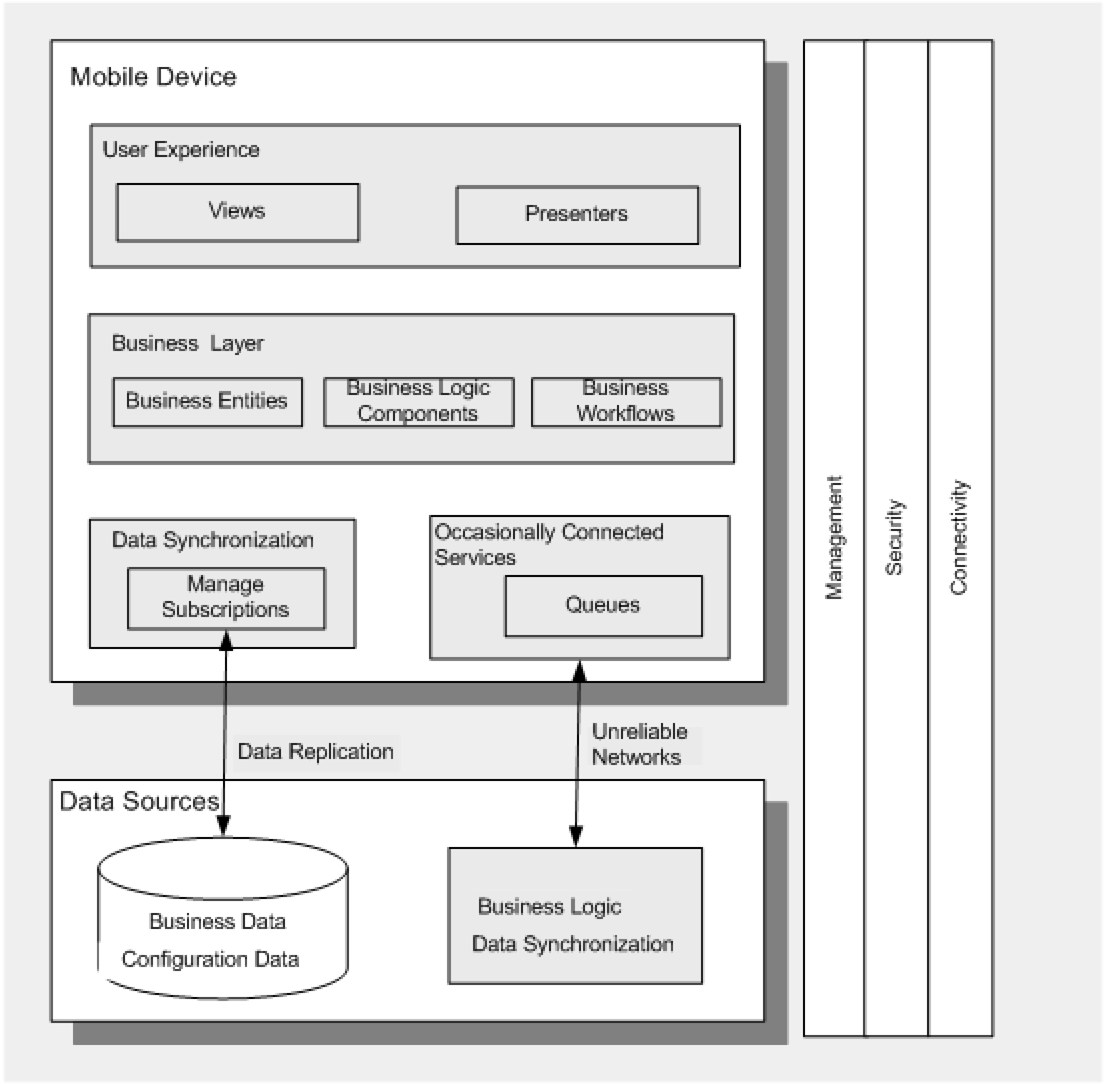
Arch Styles

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

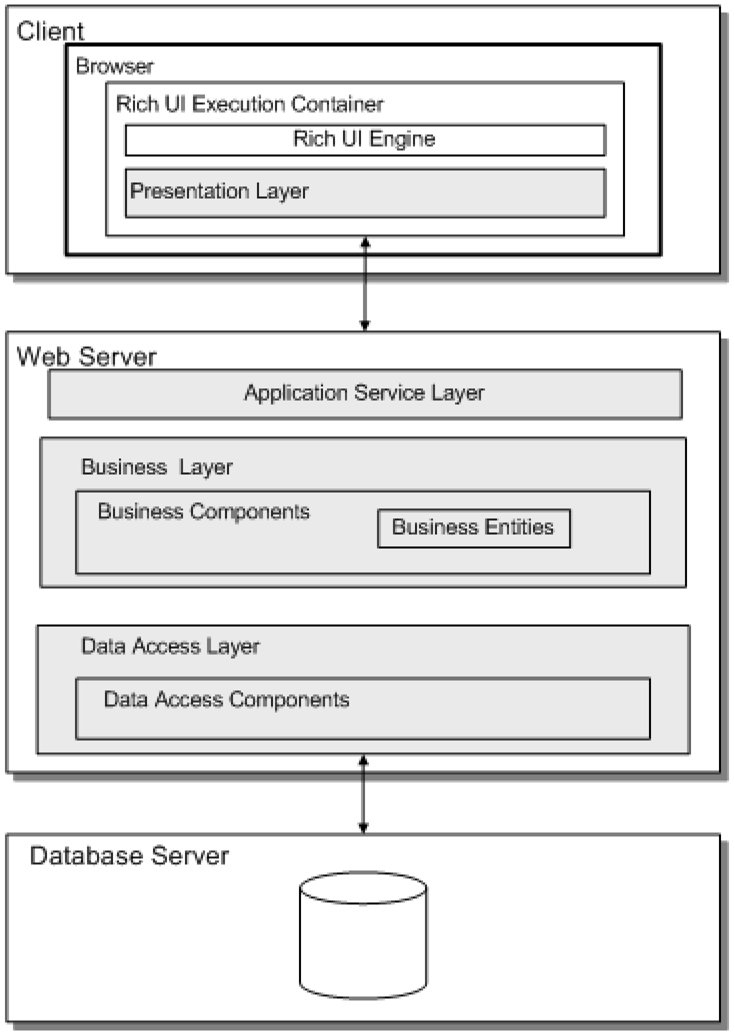
App Types (Archetypes)

Not Exhaustive, Just Illustrative

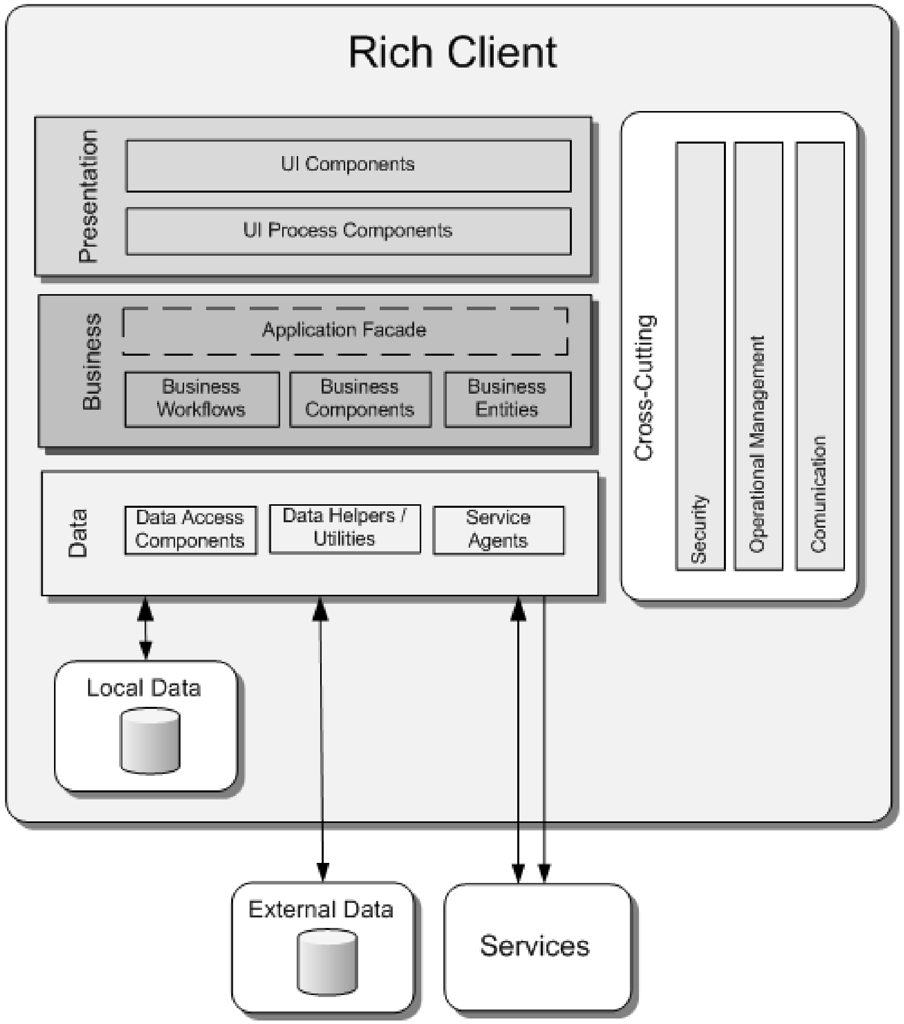
Mobile


Mobile Device
User Experience
Views
Presenters
Business Layer
Business Entities
Business Logic Components
Business Workflows
Data Synchronization
Manage Subscriptions
Occasionally Connected Services
Queues
Data Replication
Unreliable Networks
Data Sources
Business Data
Configuration Data
Business Logic
Data Synchronization
Management
Security
Connectivity

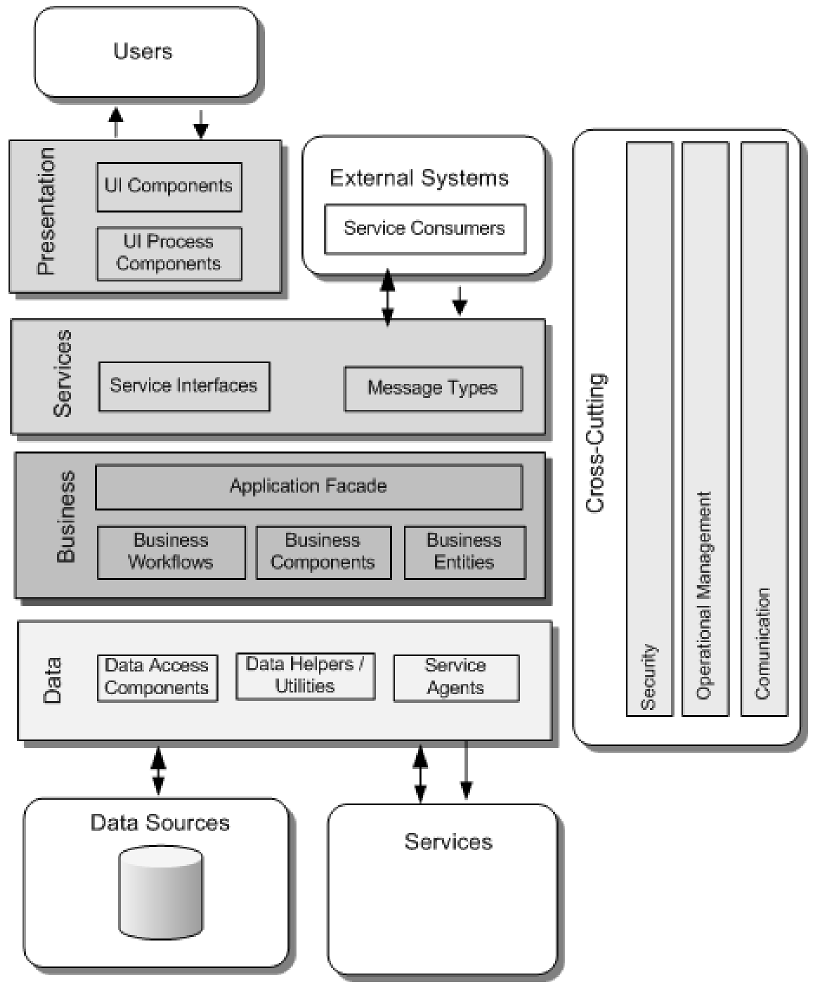
Rich Internet Application (RIA)



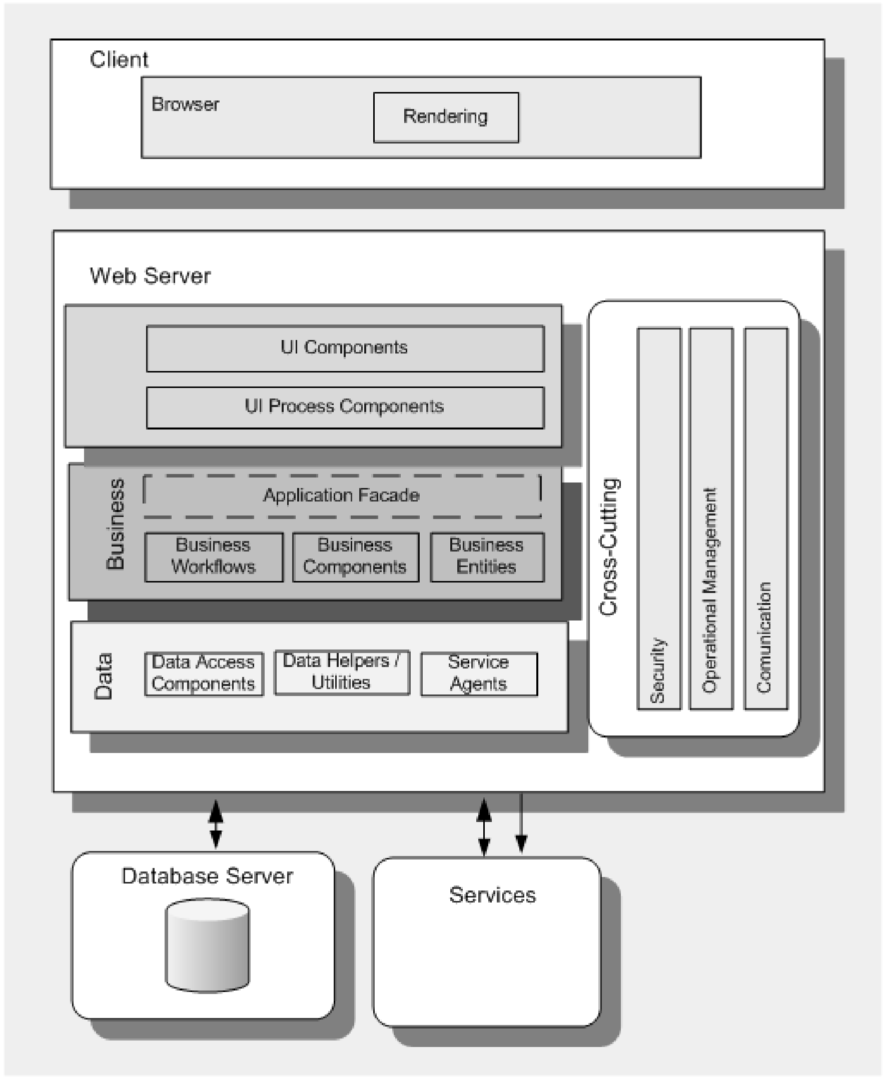
Rich Client


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
Comunication
Local Data
External Data
Services

Service

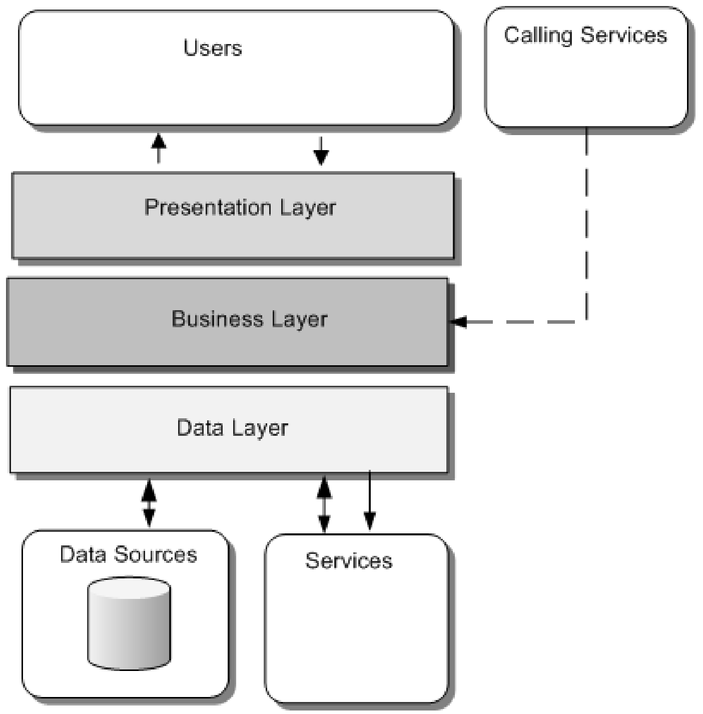

Users
Presentation
UI Components
UI Process Components
External Systems
Service Consumers
Cross-Cutting
Security
Operational Management
Comunication
Services
Service Interfaces
Message Types
Business
Application Facade
Business Workflows
Business Components
Business Entities
Data
Data Access Components
Data Helpers / Utilities
Service Agents
Data Sources
Services

Web Application

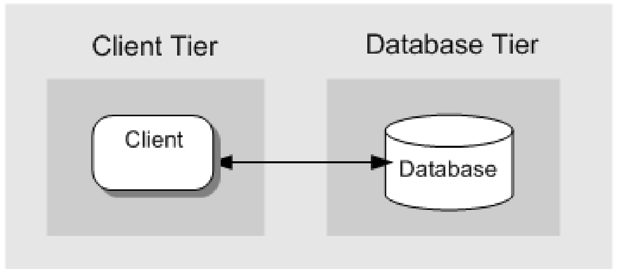


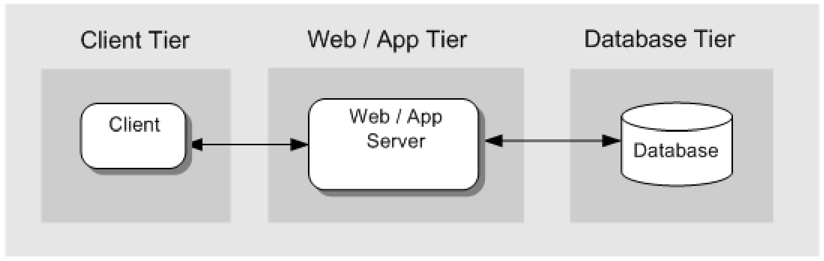
Layers, Components, Tiers

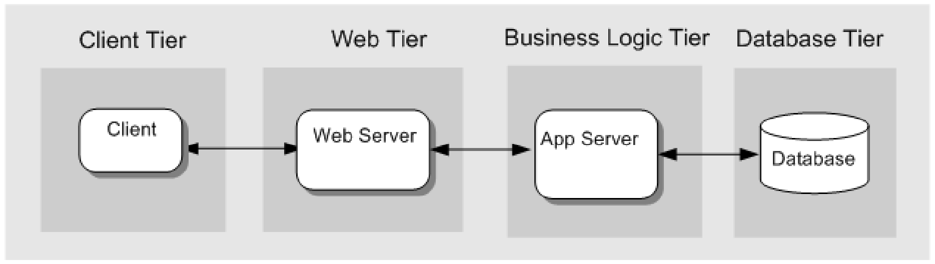
Presentation, Business, Data



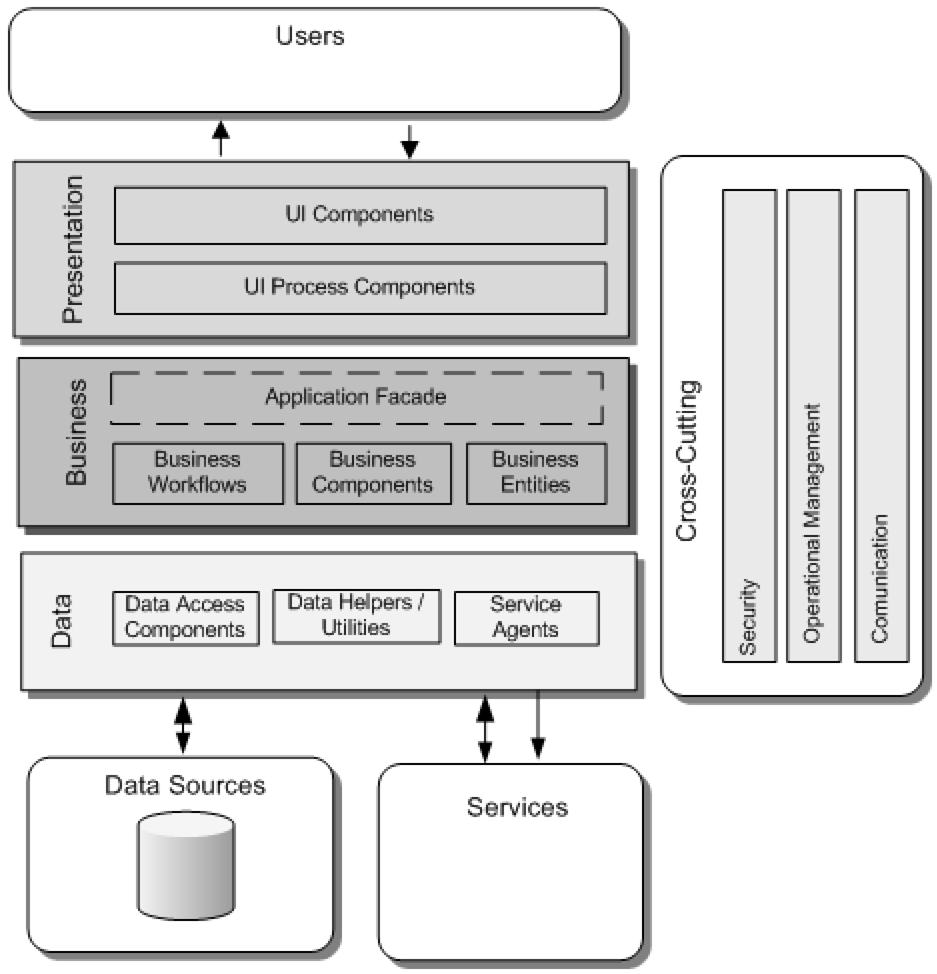
Tiers (2-Tier, 3-Tier, N-Tier)



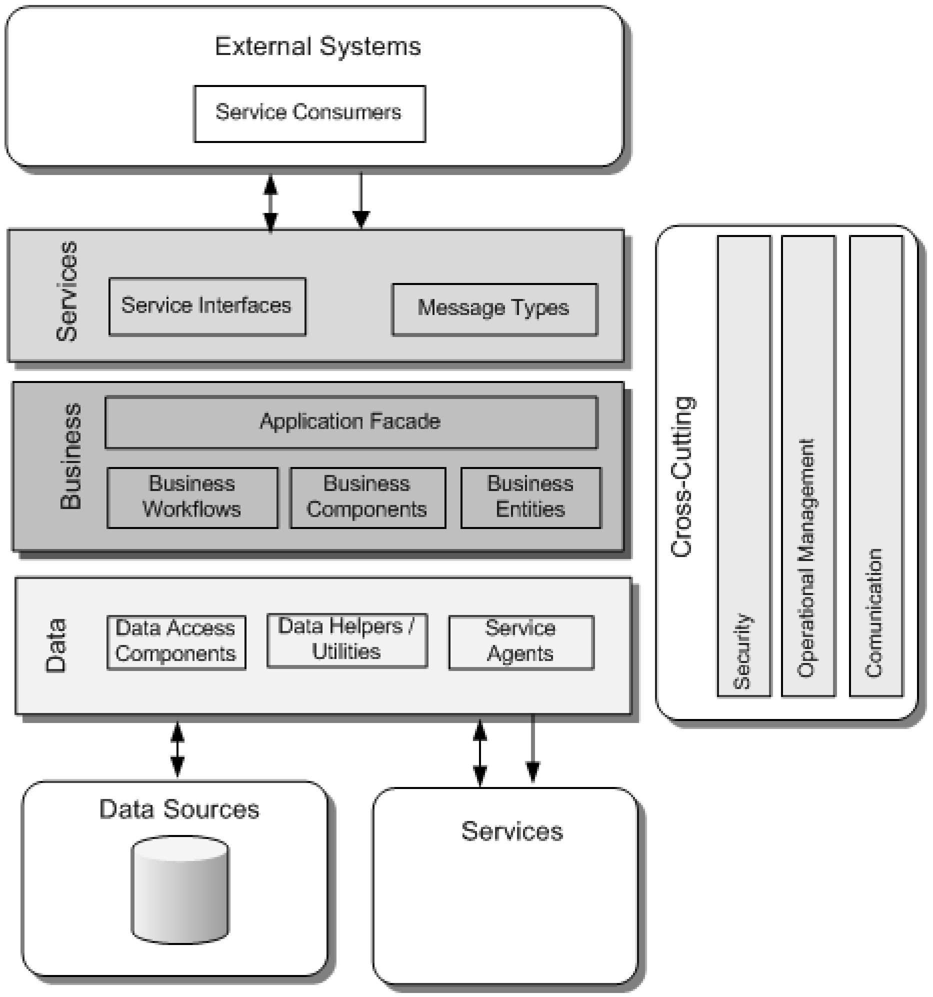




Layers / Components



Services Layer

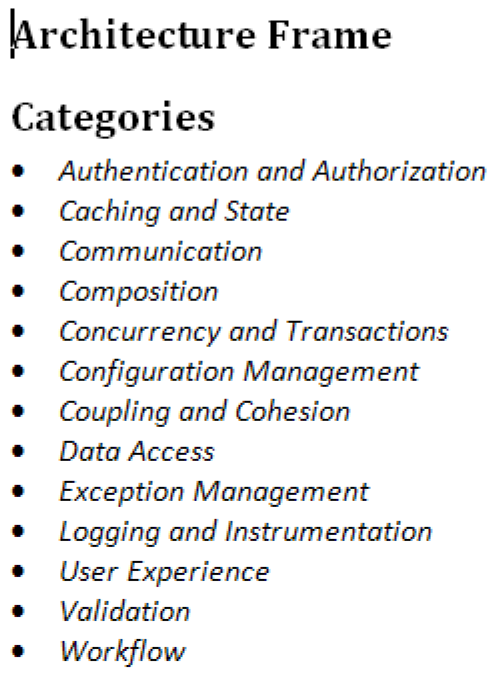


patterns & practices

proven practices for predictable results

Arch Frame

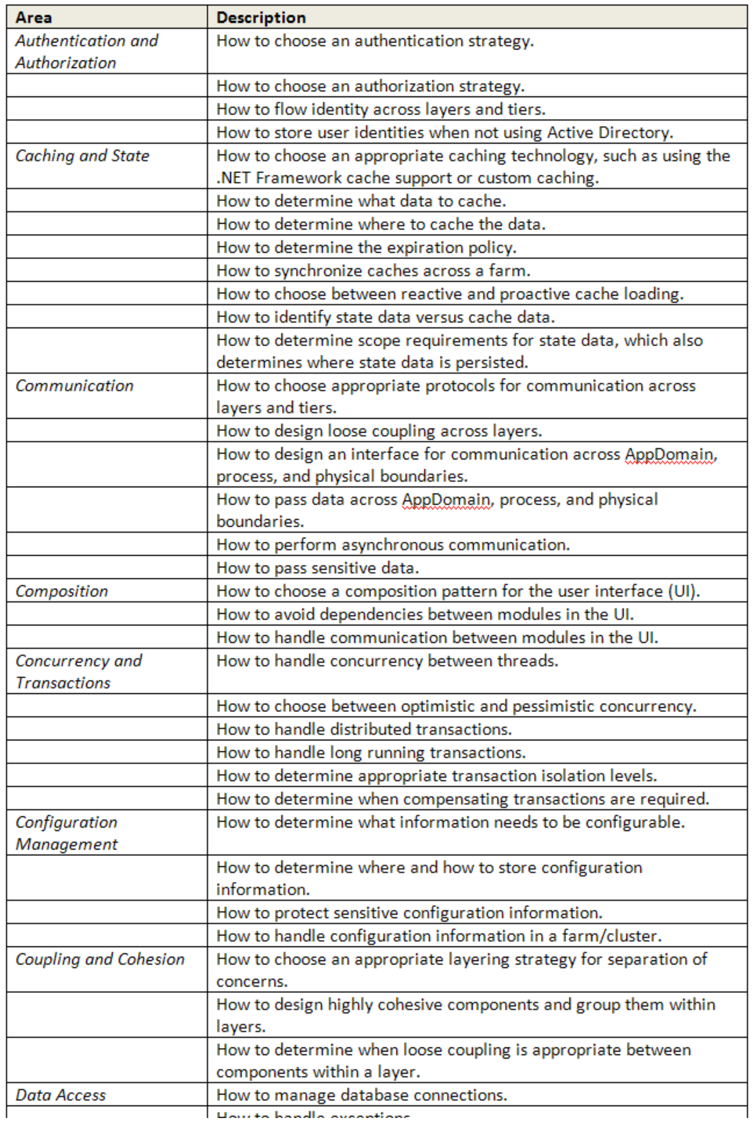
Arch Frame



Architecture Frame

Categories

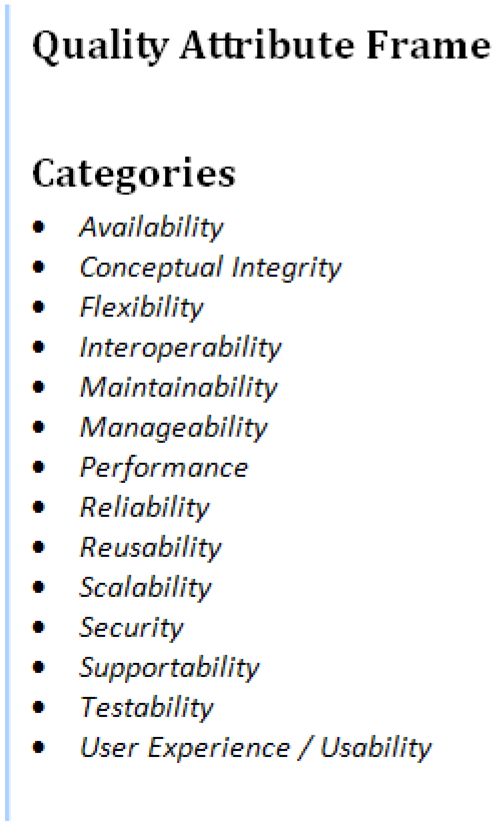
- *Authentication and Authorization*
- *Caching and State*
- *Communication*
- *Composition*
- *Concurrency and Transactions*
- *Configuration Management*
- *Coupling and Cohesion*
- *Data Access*
- *Exception Management*
- *Logging and Instrumentation*
- *User Experience*
- *Validation*
- *Workflow*



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

Quality Attributes

Quality Attribute Frame

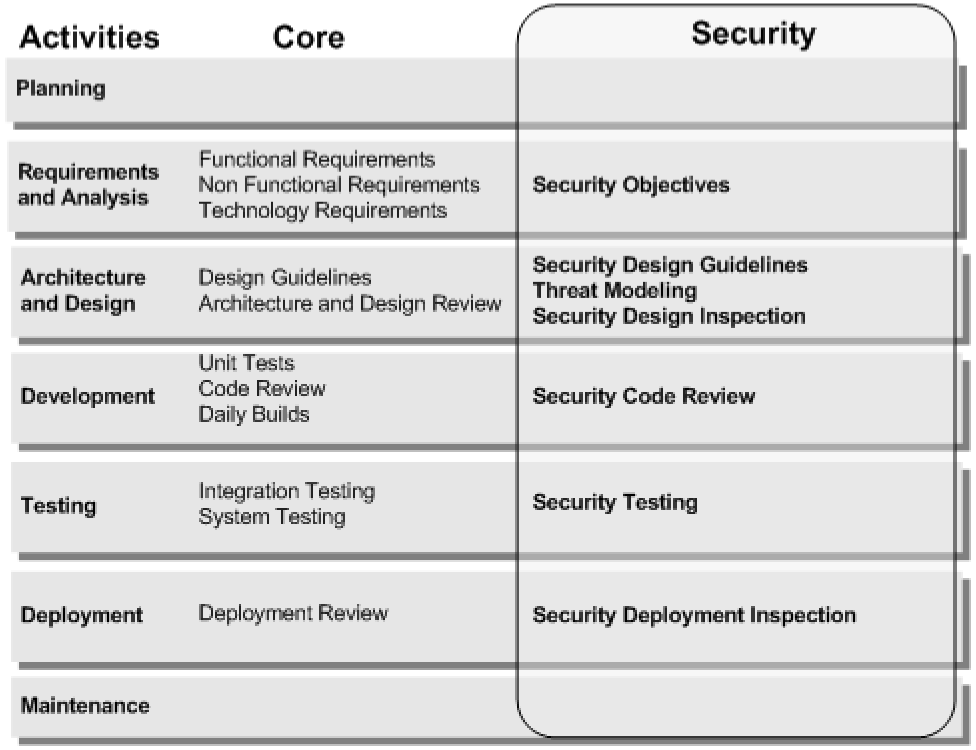


Quality Attribute Frame

Categories

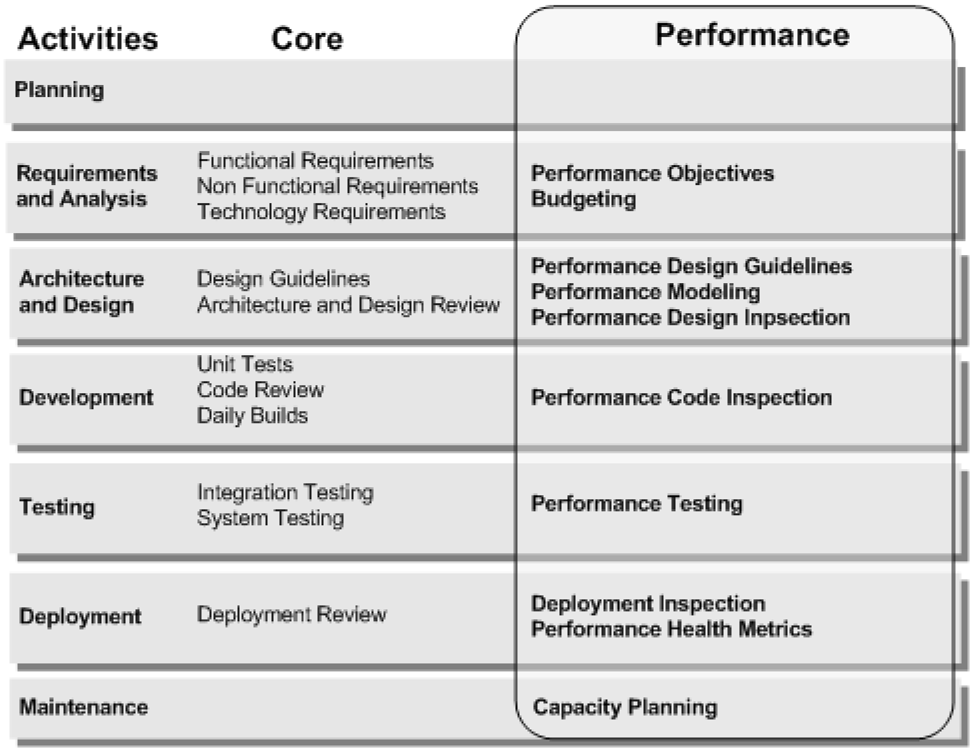
- *Availability*
- *Conceptual Integrity*
- *Flexibility*
- *Interoperability*
- *Maintainability*
- *Manageability*
- *Performance*
- *Reliability*
- *Reusability*
- *Scalability*
- *Security*
- *Supportability*
- *Testability*
- *User Experience / Usability*

patterns & practices Security Engineering



Activities	Core	Security
Planning		
Requirements and Analysis	Functional Requirements Non Functional Requirements Technology Requirements	Security Objectives
Architecture and Design	Design Guidelines Architecture and Design Review	Security Design Guidelines Threat Modeling Security Design Inspection
Development	Unit Tests Code Review Daily Builds	Security Code Review
Testing	Integration Testing System Testing	Security Testing
Deployment	Deployment Review	Security Deployment Inspection
Maintenance		

patterns & practices Performance Engineering

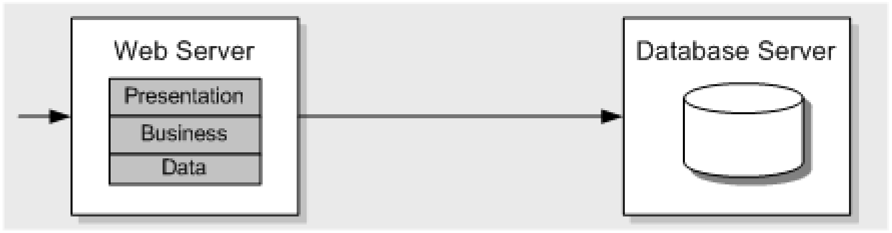


Activities	Core	Performance
Planning		
Requirements and Analysis	Functional Requirements Non Functional Requirements Technology Requirements	Performance Objectives Budgeting
Architecture and Design	Design Guidelines Architecture and Design Review	Performance Design Guidelines Performance Modeling Performance Design Inpsection
Development	Unit Tests Code Review Daily Builds	Performance Code Inspection
Testing	Integration Testing System Testing	Performance Testing
Deployment	Deployment Review	Deployment Inspection Performance Health Metrics
Maintenance		Capacity Planning

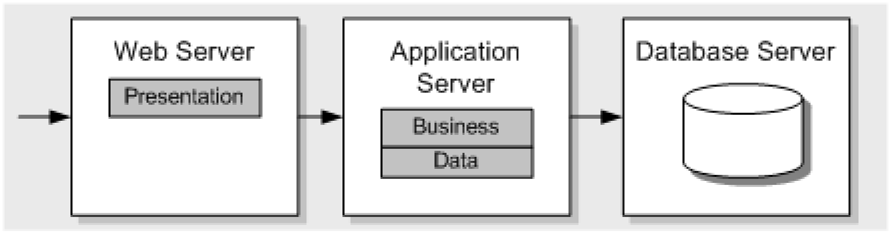
Deployment Patterns

Deployment Patterns

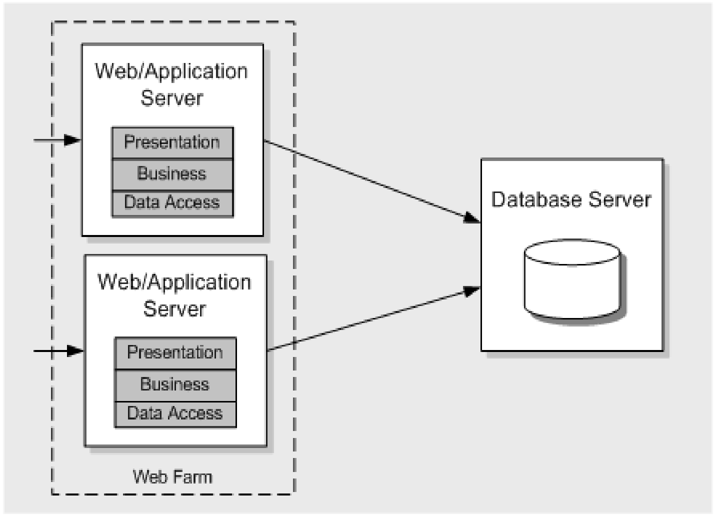
Non-Distributed



Distributed



Web/App Farms



The End

Follow Along At …

- App Arch Guide 2.0 Project Site (CodePlex) – http://www.codeplex.com/AppArch
- J.D. Meier's Blog – http://blogs.msdn.com/jmeier
- Patterns & practices Home – http://msdn.com/practices