



Architect Analyst



Project Manager



Developer



Tester



Installation

DESIGN

- Use built in mechanisms, i.e. authentication, authorisation and cryptography
- Decompose complexity or break things down into isolatable or more manageable parts
- Brainstorming of ideas or better ways of achieving the same result
- Always future proof
- Consider
 - Service oriented and Layered architectures
 - Loose coupling
 - High cohesion
 - One way dependencies
 - Abstract base classes
 - Interfaces to enforce contracts
- Consider in order of preference
 - Standard framework or SDK
 - BB&D toolbox
 - Application blocks or enterprise libraries
 - 3rd party tools
 - Custom development
- Contract

CORE

- Data is private and accessible only through properties
- Code is signed
- Consider logo compliance

WEB SERVICE

- Comply with Basic Profile 1.1
- Use doc/literal message format
- Use SoapParameterStyle.Bare

USER INTERFACE

- Not more than one row of tabs
- No tabs on the vertical
- Set Mouse Cursor to wait when busy
- Set tab Order of controls
- Control read left to right top to bottom
- Use Consistent Fonts, Colors and Sizes
- Remember screens designs does not equal paper
- Dock and Anchor controls appropriately
- Set Max Length On Fields
- Use DataBinding where ever possible
- Conform to Windows Guidelines for target platform i.e. Windows 2000 does not have style buttons
- Set Style to System and EnableVisualStyle
- Test with large fonts
- Use standard controls (Avoid 3rd Part Components)
- Decompose UI into re-usable user controls
- Provide Context Menu's for grids
- Validate Input

CONSTRUCTION

- Naming conventions
- XML Documentation conventions
- Defensive coding
 - Validate input
 - Minimise vulnerabilities
 - Report and deal with errors and exceptions
 - Log, trace and instrument all service entry and exit code
- Performance coding
 - Avoid managed destructors
 - Implement IDisposable pattern for class with heavy resources or destructors

RED FLAGS

- **Reflection** ... consider using toolbox service factory instead
- **Subclassing** ... more than 2 per day is a concern
- **Marshalling, Serialisation and Conversion** ... should be avoided by using standards based types, transports and tools

TESTING

- Every class has an associated test class
- Peer code review and testing
- Unit tests must:
 - test for success scenarios
 - test for failure scenarios
 - test for security violations

INSTALLATION

- Design and develop installs early in project lifecycle
- Ensure logo compliance
- Use standard tools

PROCESS

- Daily and automated build sequence
- Daily and automated test sequence
- Clearly define and agree on namespaces
 - Bbd.Toolbox.Frameworks

The poster shows a collection of "pointers" on what to watch out for in the relevant project phases. It does not show process!

