Refactoring, Serialization, and Version Hell

Michael Stiefel

co-author Application Development Using C# and .NET

www.reliablesoftware.com
development@reliablesoftware.com

© Copyright Reliable Software, Inc.

Goal: Reproducible Deployment

- Avoid “DLL Hell”
- Every installation has the same result
- Every uninstall is a complete removal
- Model: Install/Uninstall of a Compact Disc
- Allow side-by-side execution
- Test What You Ship
- Deterministic

© Copyright Reliable Software, Inc.

The Problem

- Deterministic Deployment requires Unique Identifiers (Strong Names)
- Strongly Named Assemblies must reference Strongly Named Assemblies
- Types are Assembly Relative
- Assemblies reference an assembly with a particular identity.

© Copyright Reliable Software, Inc.

Assemblies Need Unique Identifiers

- Application “Bill of Materials”
- Determine What is Missing
- *No Unique Id Example*
  - Not deterministic

© Copyright Reliable Software, Inc.

Unique Ids Require Strong Names

- Cryptographically Unique Name
  - public / private key
- Incorporate Version
  - allow versioning of assemblies
- Incorporate Culture
- *Name-version-culture-public key*
- Placed in Assembly Manifest

© Copyright Reliable Software, Inc.

Assemblies Referenced By Unique Id

- Public Key or Public Key Token of referenced assembly
  - Public Key Token is lowest 8 bytes of SHA1 hash of public key
  - more compact than public key
- Placed in referencing assembly at compile time
- *Unique Id Example*
  - deterministic

© Copyright Reliable Software, Inc.

Note: Technically Optional

"*A conforming implementation of the CLI need not perform this validation, but it is permitted to do so, and it may refuse to load an assembly for which the validation fails. A conforming implementation of the CLI may also refuse to permit access to an assembly unless the assembly reference contains either the public key or the public key token. A conforming implementation of the CLI shall make the same access decision independent of whether a public key or a token is used.*"

Common Language Infrastructure
Partition II: Metadata Definition and Semantics
Section 6.3

© Copyright Reliable Software, Inc.

Determinism Must Propagate

- Strongly named assemblies must call into strongly named assemblies
  - Keeps determinism from the perspective of the caller
  - *Strongly Named Reference Example*

© Copyright Reliable Software, Inc.

Type is Assembly Relative

- Different Names Changes the Type
  - *Almost Identical Example*
- Versioning an Assembly Changes the Type
  - *Types Example*

© Copyright Reliable Software, Inc.

Implications …

© Copyright Reliable Software, Inc.

Version Compatibility

- Specified by version binding policy in the application's configuration file.
- Policy allows administrators to configure applications without recompilation.
  - Security Policy
  - Version Binding Policy
  - Remoting Configuration

© Copyright Reliable Software, Inc.

Version Policy

- Can indicate version equivalences for application

```
…
 <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
   <publisherPolicy apply = “yes” />
   <dependentAssembly>
     <assemblyIdentity name="Customer" publicKeyToken="f95c9298a3bfb06b" />
     <bindingRedirect oldVersion="1.0.0.0-1.2.0.0" newVersion="1.2.0.0" />
   </dependentAssembly>
</assemblyBinding>
 …
```

- *Version Policy Example*

© Copyright Reliable Software, Inc.

Publisher Policy

- Can specify equivalent versions
  - for major.minor version
- Can be turned off in application configuration file for entire application
- Insure critical fixes get to all applications
- Not for upgrades
- *Publisher Policy example*

© Copyright Reliable Software, Inc.

Machine Policy

- Machine.config can have version policy set by administrators
- Policy resolution order: app, publisher, admin

© Copyright Reliable Software, Inc.

Policy Hell

- Policy complicates a programmer's life.
- Policy simplifies an administrator's life.
  - Eases application replacement
  - Precise bug fix releases
- Version Hell is an example of Policy Hell.
- Get used to it!
  - The world does not revolve around programmers.
  - Would be easier with better tool support for rebinding or source code control integration.

© Copyright Reliable Software, Inc.

Assembly Granularity

More precise releases

Emergency fixes

Security fixes

Easier to Find problems

Quicker development build cycle

vs.

Management of many assemblies

© Copyright Reliable Software, Inc.

Dependencies

- Within Assembly Dependencies OK
- Minimize Dependencies Among Assemblies
- Consider Assembly Dependencies as well as Class Dependencies

© Copyright Reliable Software, Inc.

Refactoring

"*the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure.*"

Martin Fowler
Refactoring: Improving the Design of Existing Code

© Copyright Reliable Software, Inc.

Logical vs. Physical Design

- Distinguish Logical and Physical Design
- Logical design = class design
  - Keep software evolvable
- Physical design = class distribution among assemblies
  - Allow more precise upgrades
  - Distributing classes among assemblies is not enough
- *Physical Design Example*

© Copyright Reliable Software, Inc.

Logical Dependencies

- Interfaces help decouple logical dependences.
- Design Patterns help decouple logical dependences.
- Still have to think carefully.
  - *Physical Design Example Step 5*
  - *Upside Down Example*

© Copyright Reliable Software, Inc.

When to Version?

- Do release, then bump version number, clearer when QA, users report mistakes.
- Change only when assembly changes
- Deploy changed assembly and revised version file.
  - use side by side deployment (Codebase, GAC)
  - revert to previous versions through policy
- XCopy deployment to track different customer deployments

© Copyright Reliable Software, Inc.

Refactoring Physical Design

- *Moving Types example*
- Version, rebuild affected assemblies
- Dynamic load produces complications

© Copyright Reliable Software, Inc.

Refactoring and Saved Data

- *Serialization Example*
- SerializationBinder to redirect type
  - Changed Assembly version
  - Type moved to different assembly
  - Type no longer exists
- IDeserializationCallback
  - data moved to different classes

© Copyright Reliable Software, Inc.

Class Versioning

- No direct relationship between assembly versioning and class changes.
  - Multiple versions can correspond to a single class version (algorithmic changes)
  - Multiple classes can exist in an assembly
  - Person who versions assemblies may not be the person who modifies a class

© Copyright Reliable Software, Inc.

Summary

- Goal : Reproducible install / uninstall
- Components must be uniquely identifiable
- Policy drives version compatibility
- Physical as well as Logical Design matters

© Copyright Reliable Software, Inc.