

Interface Migration Patterns

Stephen Vance
Boston Software Craftsmanship
December 2, 2013

Interface

- ◆ Any defined boundary between two components
- ◆ Examples
 - ◆ Database schema
 - ◆ API name, parameter, parameter name, return type
 - ◆ REST or SOAP end points

Versioning

- ◆ Pros

- ◆ Conceptually simple
- ◆ Can be maintained indefinitely
- ◆ Simple to manage, ...

- ◆ Cons

- ◆ Until it isn't
- ◆ Redundancy
- ◆ Compatibility

The Alternative: Migrations

- ◆ Pros

- ◆ Single source base
- ◆ Can reduce down time

- ◆ Cons

- ◆ Conceptual complexity
- ◆ Limited tool support
- ◆ Requires sunset horizon

The Meta Pattern

- ◆ N-1 Compatibility
- ◆ Every incompatible interface change can be reduced to two compatible interface changes, the first constructive or permissive and the second destructive or restrictive

Katas

- ◆ Java
 - ◆ Add param
 - ◆ Remove param
- ◆ JavaScript
 - ◆ Add param
 - ◆ Remove param
- ◆ MySQL
 - ◆ Rename column

References

- ◆ Ambler & Sadalage, "Refactoring Databases"
- ◆ Sadalage, "Recipes for Continuous Database Integration"
- ◆ <http://dataserefactoring.com/>

Tools

- ◆ Flyway, <http://flywaydb.org/>
- ◆ dbmigrate, <http://code.google.com/p/dbmigrate/>
- ◆ Liquibase, <http://www.liquibase.org/>
- ◆ Rails ActiveRecord::Migration