Michael Stiefel

Reliable Software, Inc.

www.reliablesoftware.com

# How to Partition and Layer a Software Application

Session Code: AR. 10

SDN CONFERENCE

SOFTWARE DEVELOPMENT NETWORK

## The Problem

Software today cannot be rewritten as hardware, requirements, and technologies change.

## The Consequences of Failure

Applications will no longer be used.

Businesses will not be competitive.

Customers will not be happy.

You will not find new work.

## The Solution

Minimize consequences of software change.

Architect and design to evolve parts of an application independently.

*Partitioning and Layering* decouple parts of an application.

## Permits

Modifying part of an application without impacting other parts

Unit tests to validate changes to an application independently

## How do you partition and layer an application?

# Interfaces vs. Inheritance

Classes are not Types

Class inheritance is not Interface inheritance

"Program to an interface, not an implementation."

## Composition vs. Inheritance

Inheritance is white box reuse

Composition (association) is black box reuse

"Inheritance breaks encapsulation."

"Favor object composition over class inheritance."

## Coupling

Distinguish Essential Coupling

Remove Inessential Coupling

## Partitioning and Layering

Remove Inessential Coupling due to programming artifacts to minimize the impact of change

Cannot remove the Essential Coupling based on required behavior or semantics

# Electrical Analogy

Wall socket *interface* minimizes the inessential coupling due to the physical shape of plugs and appliances

An interface cannot remove the essential behavioral coupling of voltage and amperage of standard current

A transformer is a *pattern* to modify the behavior to minimize effect of the essential coupling.

# Object Oriented Design Principles

- Interface Based Design
- Use of Composition
- Single responsibility
- Patterns such as Facades
- Dependency Inversion
- Factories

# Demos

Illustrate through a series of demos

## For More Information

*Design Patterns*, Erich Gamma, et. al.

Section 1.6, Chapter 3, Chapter 4 Façade Pattern

*Domain-Driven Design*, Eric Evans

Chapters 4, 6

*Patterns of Enterprise Application Architecture*, Martin Fowler

Chapters 1-3, 13, 18

*Working Effectively with Legacy Code*, Michael Feathers

## Summary

Partitioning and layering are achieved through a mixture of design and programming techniques.

Partitioning and layering minimize the impact of change on an application.

Partitioning and layering allow you to build of unit tests to validate the software as changes are made.

## Evaluation form

**Vul je evaluatieformulier in en maak kans op een van de prachtige prijzen!!**

**Fill out your evaluation form and win one of the great prizes!!**

**Session Code: AR 10**