Stefan Lange | empira Software GmbH

Von WinForms nach WPF

Tipps zu Hybrid-Anwendungen

Stefan.Lange@empira.de

25.02.2010

Agenda

Tipps zu den folgenden Punkten:

- Entscheidungen und Vorgehensweise
- WPF in WinForms Integration
- WinForms in WPF Integration
- Typische Probleme und Lösungsansätze
- Anwendungsarchitektur
- Fragen/Diskussion

Vorüberlegungen

Migrieren oder neu schreiben?

- Genaue Analyse durchführen
- Wo steht man?
- Wo will man hin?
- Ressourcen und Kenntnisstand der Entwickler
- Keine ad-hoc Entscheidungen treffen

Vorteile WPF (vs. WinForms)

- Bessere Trennung von UI und Code
- Datenbinding (MVVM) sehr produktiv
- Validierung
- Templates und Styles
- Gute externe Control Libraries verfügbar
- (Neue Features: Animation, 3D, Video, …)
- Hardwarebeschleunigt

Nachteile WPF (vs. WinForms)

- WPF vermeintlich komplizierter
- Erhöhte Hardwareanforderungen
- Leider immer noch: Probleme mit Terminalserver-Lösungen von Drittanbietern

Vorteile WinForms (vs. WPF)

- Breitere Entwicklergemeinde
- .NET 2.0 ist im Prinzip noch ausreichend

Nachteile WinForms (vs. WPF)

- „Technisches Auslaufmodell“
- GDI+ wird im Prinzip nicht weiterentwickelt
- Microsoft selbst verwendet es nicht mehr für neue Produkte

Umstieg: Wie fängt man an?

Fragen stellen:

- Analyse: Wo will man eigentlich hin?
- Entscheidung hängt stark von der konkreten Anwendung / den konkreten Rahmenbedingungen ab
- Pro und contra in Bezug auf die nächste Produktversion abwägen

Option: Bei WinForms bleiben

Vorteile

- Es ändert sich gar nichts (= bequem)
- Kein vorübergehender Produktivitätseinbruch
- Kein .NET Update beim Kunden

Option: Bei WinForms bleiben

Nachteile

- Ggf. Zeitdruck, wenn man doch umsteigen muss/will (z.B. wegen neuer Features)
- Verliert ggf. Anschluss an WPF & Silverlight

Option: Mit WPF neu schreiben

Vorteile

- Geringster Gesamtaufwand der WPF-Version
- Konsistentes UI und Applikations-Modell
- Alle WPF Features können unmittelbar genutzt werden

Wenn man die Möglichkeit dazu hat, dann mit WPF ein neues keines Projekt starten!

Option: Mit WPF neu schreiben

Nachteile

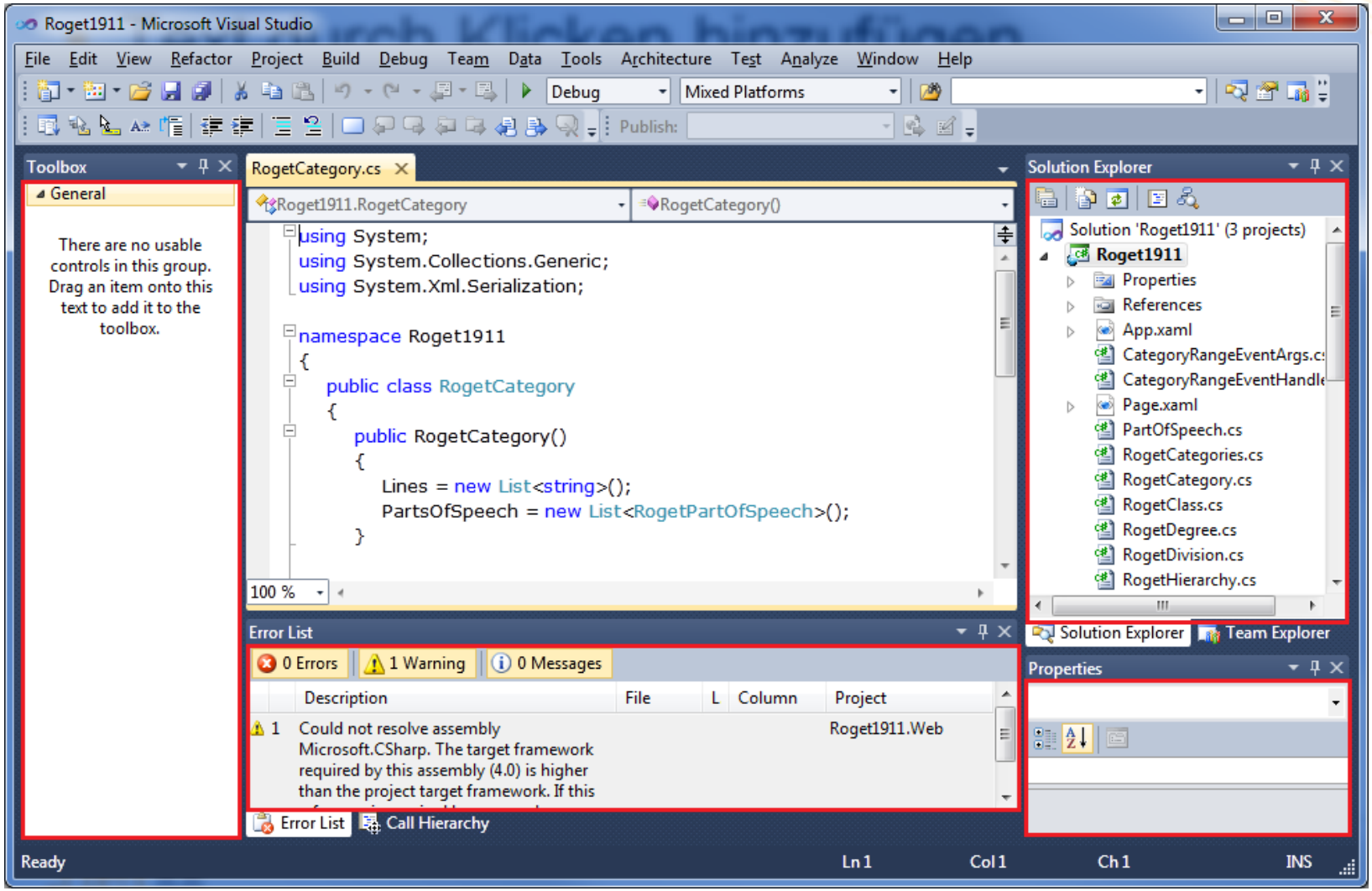
- Dauert am längsten bis zur fertigen Version
- Sehr steile Lernkurve
- Bei größeren Anwendungen oft nicht praktikabel

Option: Hybride Anwendung

Mischung von beiden Welten

- Manche Teile werden mit WPF erweitert
- Andere Teile bleiben WinForms
- „Sanfter“ und ggf. unsichtbarer Übergang
- Vermeidung des „Second System Effects“

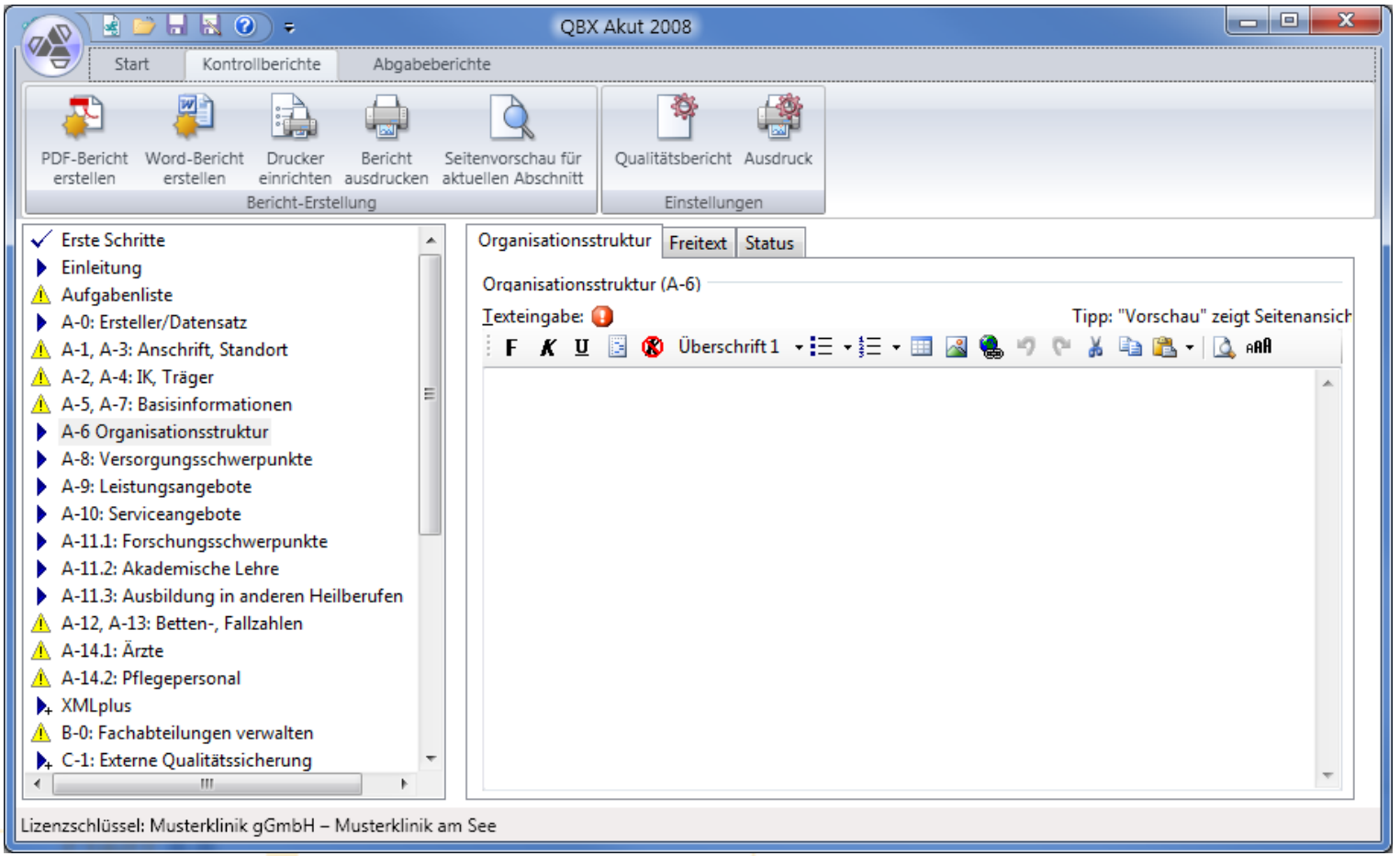
Beispiel: Visual Studio 2010



rot = WinForms bzw. natives Win32

Beispiel: RibbonWindow

WPF Frame mit WinForms Content



Hybride Anwendung

Vorteile

- Entwickler lernen WPF Schritt für Schritt
- Kurzer Zeitraum bis zur ersten Version
- Compiler-technisch unproblematisch: Einfach WPF *und* WinForms Assemblies referenzieren

Hybride Anwendung

Nachteile

- Ggf. nicht ganz konsistentes UI
- Man kann auf sehr komplexe Probleme stoßen (Interop, Applikations-Modell, …)

1. WinForms mit WPF erweitern

Teilkomponenten auf WPF Basis

- Neue modale Dialoge
- Spezielle UserControls
- Grafischer Editor
- Dokumentgenerierung / Vorschau / Druck
- XPS Dokumente generieren

WPF in WinForms Control

ElementHost regelt Interop

DEMO

- WPF Dialog-Boxes in WinForms
- WPF UserControls in WinForms

Tipps zur WPF-Einbettung

- Modale Dialoge funktionieren „einfach so“
- UserControls benötigen ElementHost
- Modeless Dialoge vermeiden

Wichtig

- Wenn es mit ElementHost nicht sofort funktioniert, nicht weiter rumprobieren

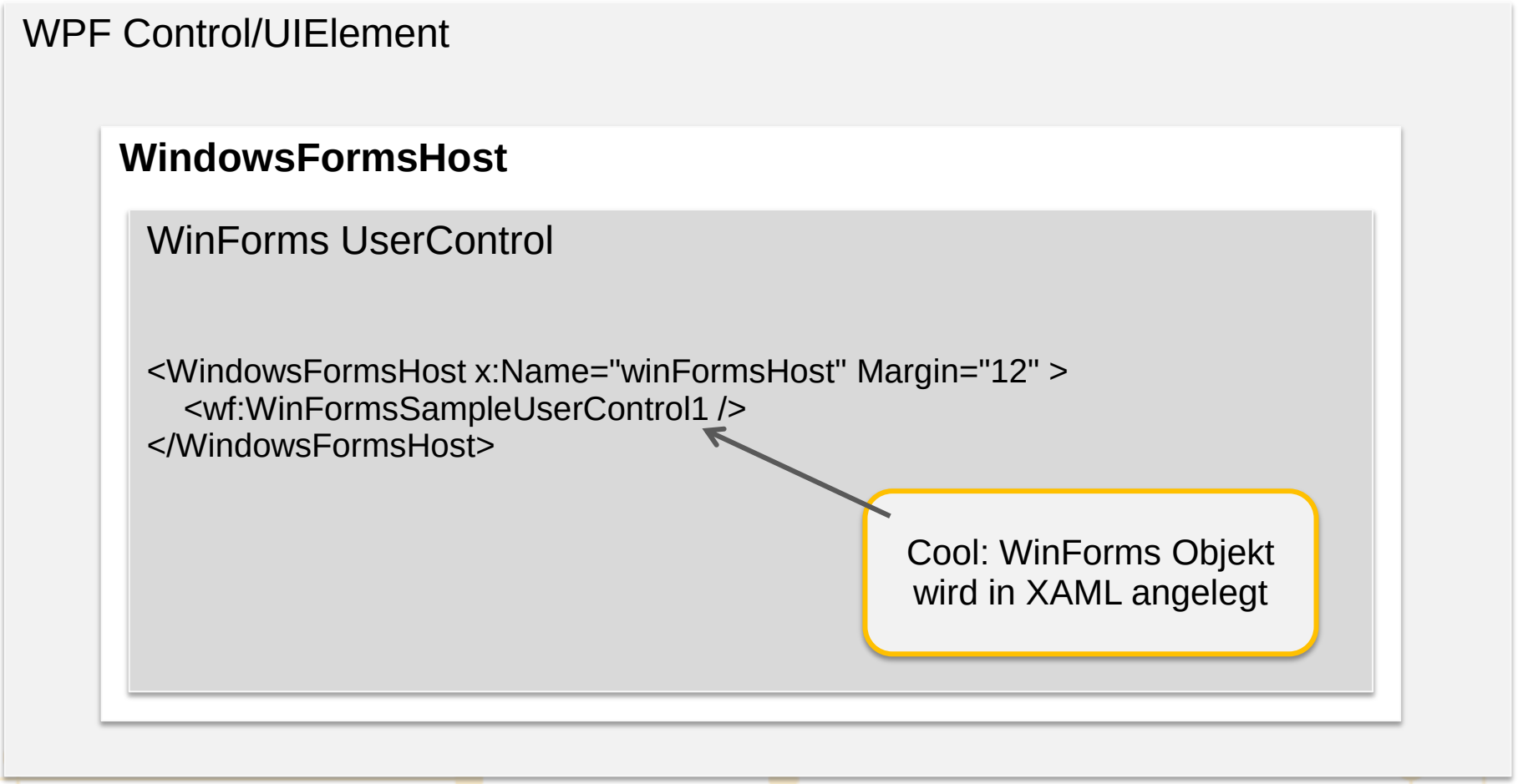
2. MainWindow wird WPF

Teilkomponenten bleiben WinForms

- Hauptfenster kann beispielsweise Ribbons verwenden
- Besseres „Workspace Layout“ möglich
- Einzelne Dialoge und UserControls bleiben in WinForms
- UI kann Stück für Stück erneuert werden

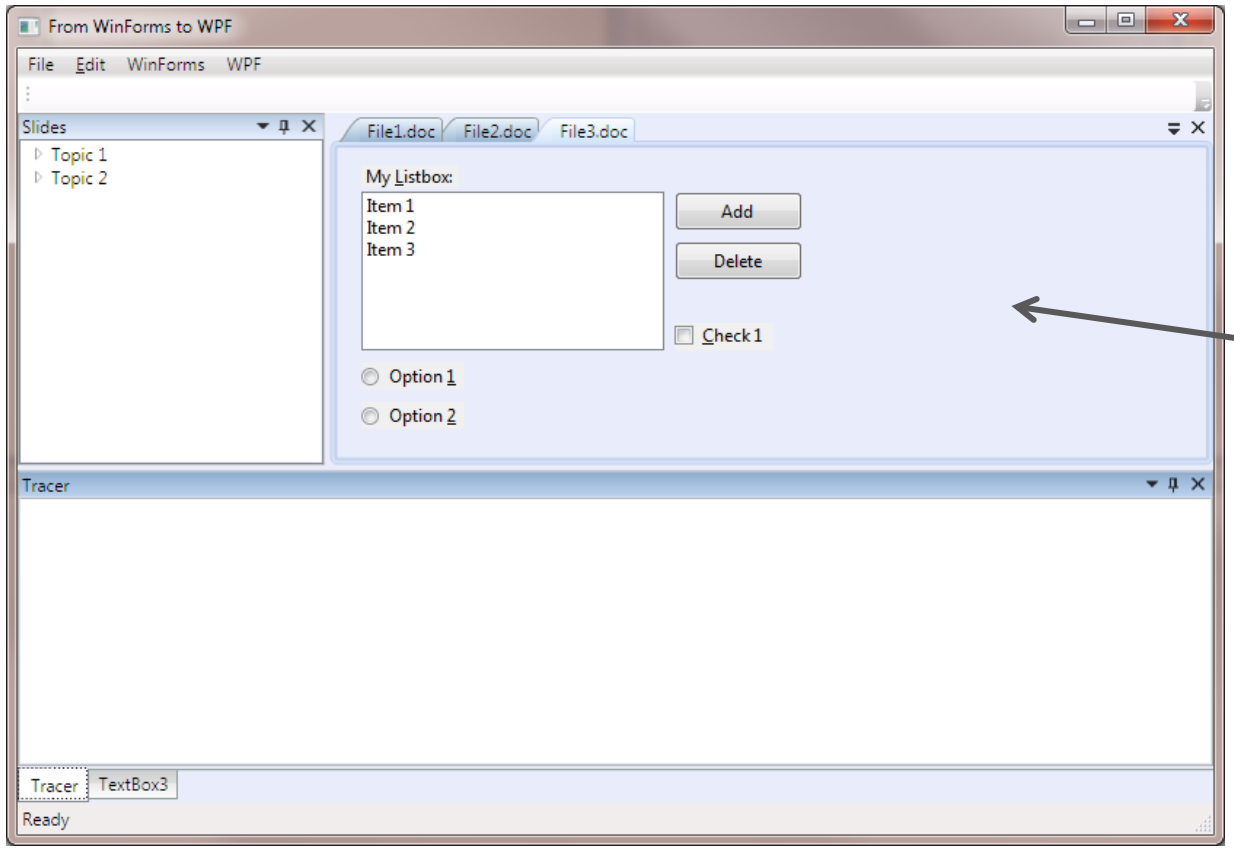
WinForms in WPF Control

WindowsFormsHost regelt Interop



Beispiel mit AvalonDock

WPF MainWindow ähnlich Visual Studio



WinForms UserControl

WinForms in WPF einbetten

DEMO

- Main Window mit Avalon Dock
- WinForms Dialoge aufrufen
- UserControls in WPF Controls einbetten

Tipps zur WinForms-Einbettung

- Modale Dialoge funktionieren „einfach so“
- WindowsFormsHost für UserControls
- Modeless Dialoge vermeiden
- WindowsFormsHost/ElementHost Verschachtelung vermeiden (wird sehr langsam)

WARNUNG

Es ist komplizierter als es aussieht!

- Tastaturinterface hakt manchmal
- Farben, Hintergrund und Transparenz teilweise schwierig einstellbar
- Probleme bei verschiedenen Fonts
- Probleme bei Themes (XP / Vista / 7)
- Layoutprobleme bei WinForms Docking
- Resizingprobleme bei DPI != 96
- u.v.m.

Generelle Tipps

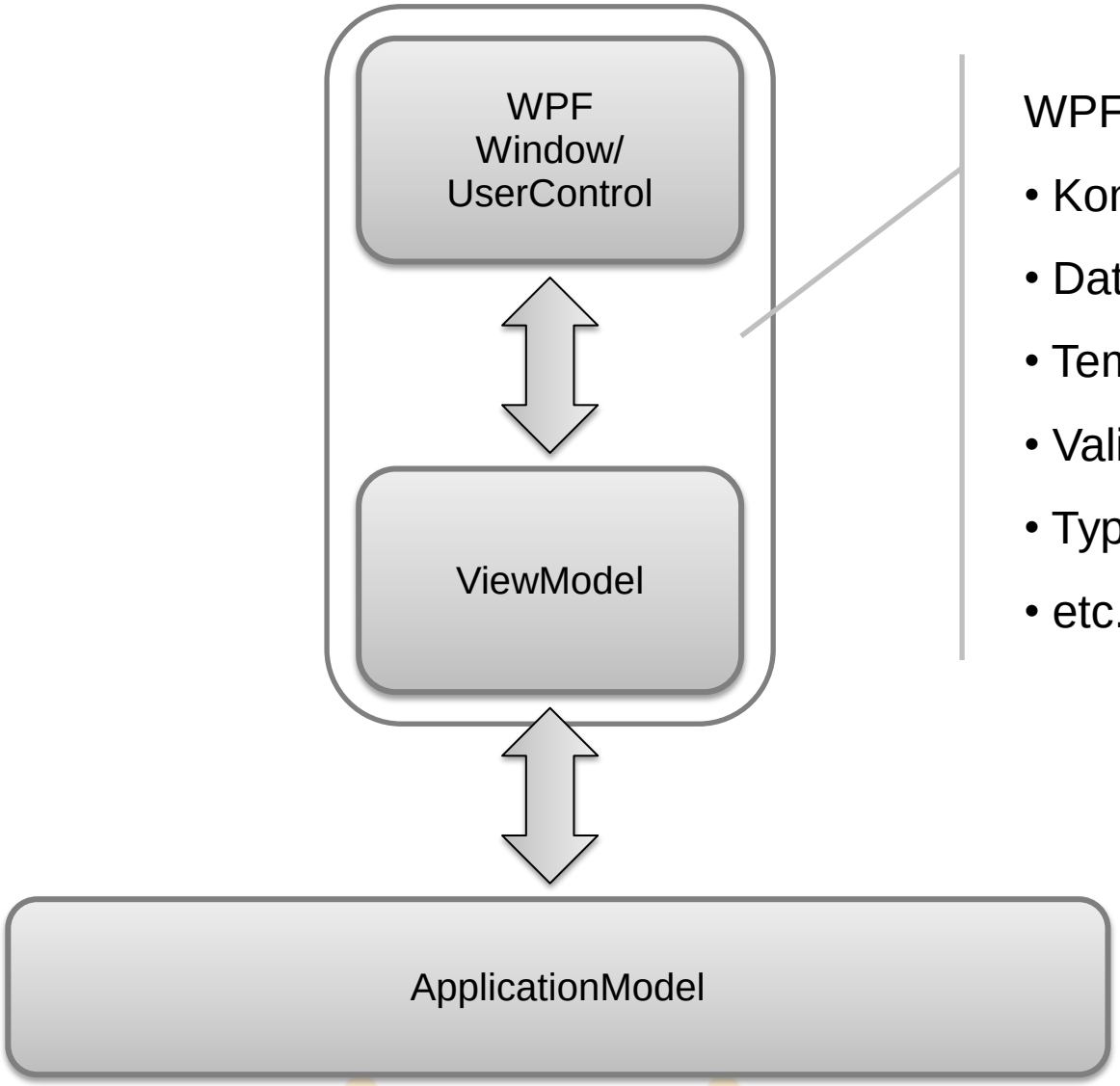
- Prototyp bauen und ausprobieren
- Auf allen Windows Versionen testen
- Wenn etwas nach einer bestimmten Zeit nicht klappt -> neu programmieren

Anwendungsarchitektur

Wie baut man Hybrid Anwendungen

- Anbindung des UI an Legacy Applikation
- Refactoring der Altanwendung

Aufbau einer WPF Komponente



WPF voll nutzen:

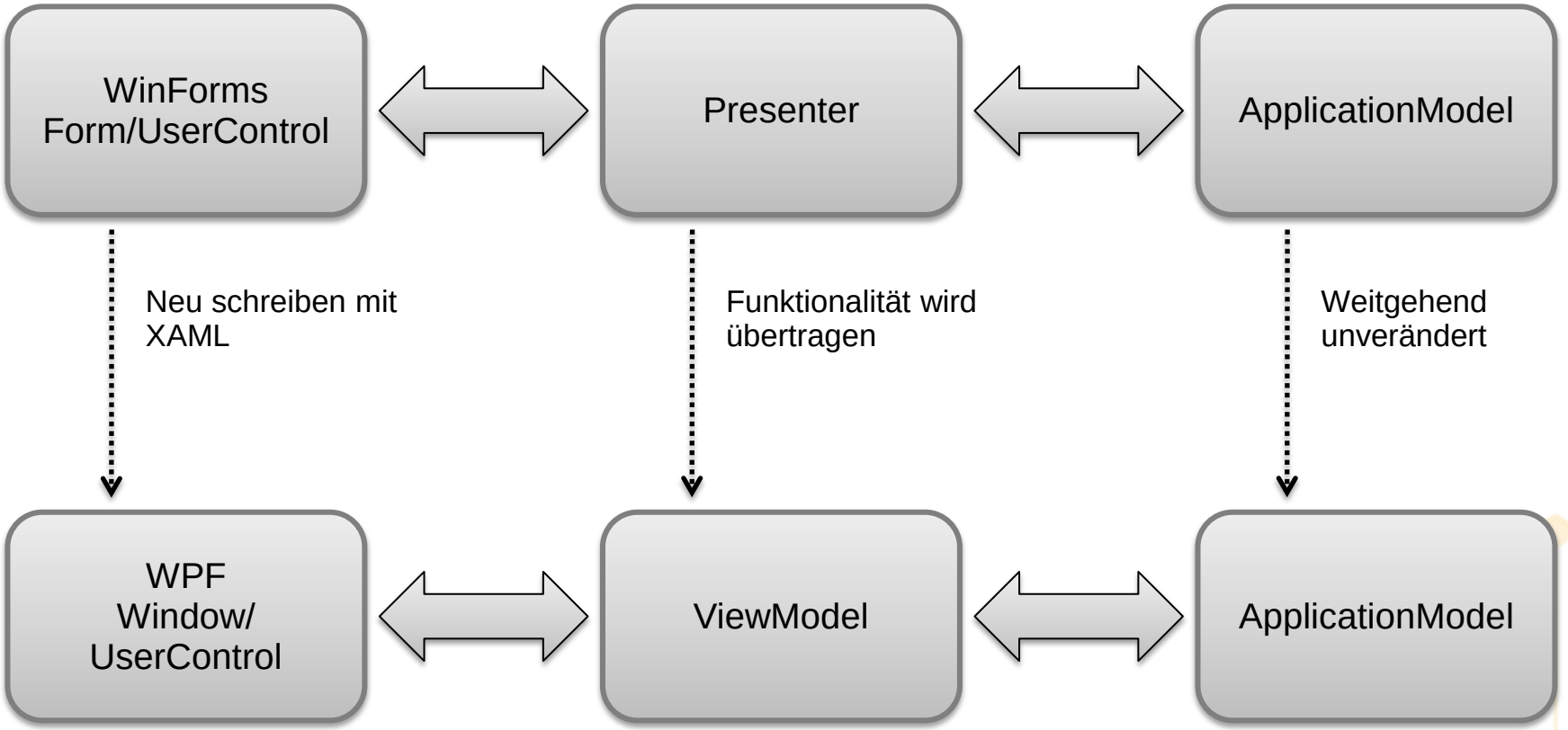
- Kommandos
- Datenbindung
- Templates
- Validierung
- Typ-Konverter
- etc.

Tipps zur Architektur

- Für jeden WPF View ein eigenes ViewModel schreiben
- DataContext ans ViewModel binden
- Zugriff auf den alten Applikations-Code nur aus dem ViewModel heraus

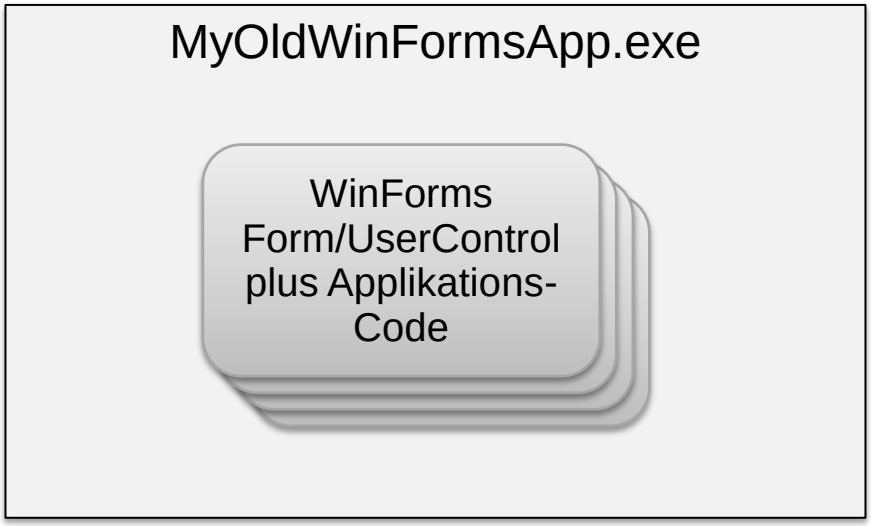
Gute WinForms Architektur

Wenn man sehr viel Glück hat…



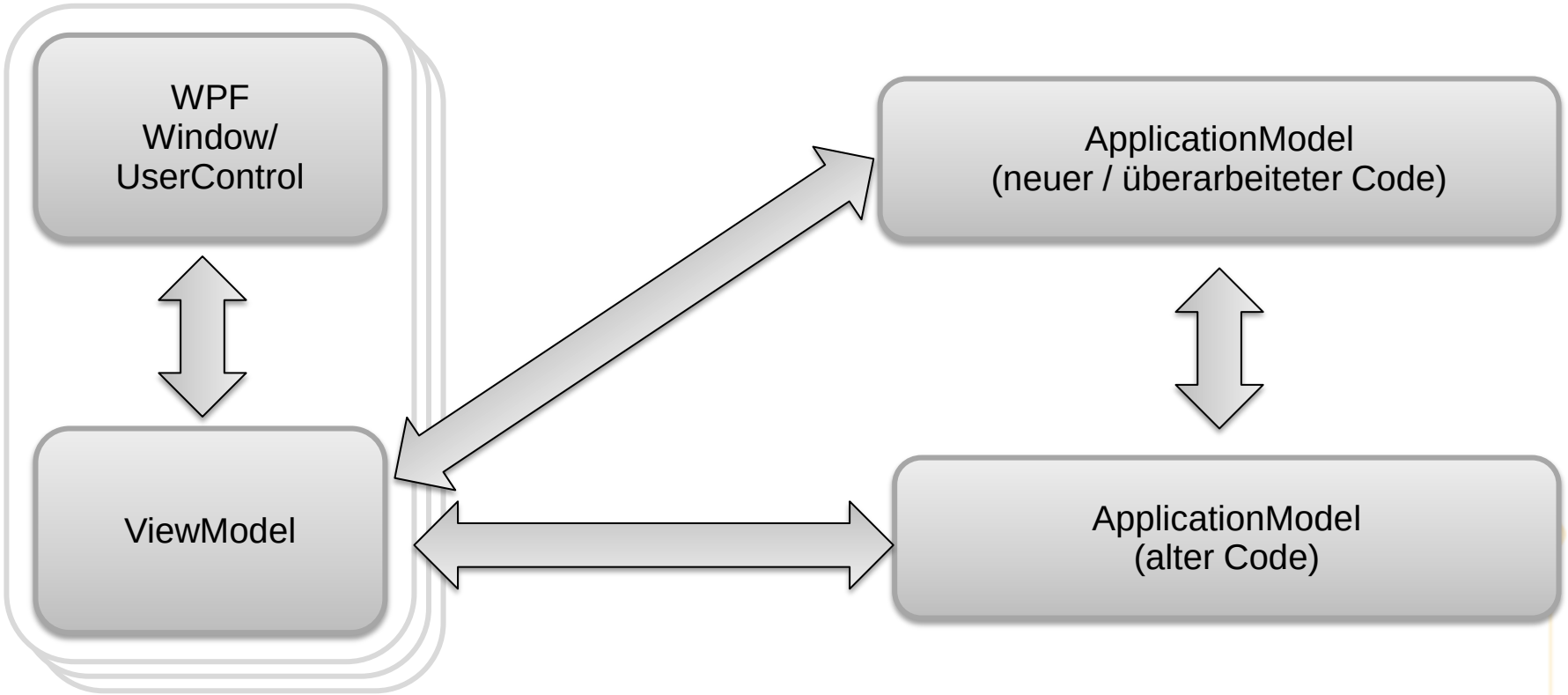
Typische WinForms Architektur

Meistens sieht es leider eher so aus:



Architektur-Umbau

Code Refaktorierung



Tipps zum Umbau

- MVVM Pattern verwenden
- Neue Komponenten sauber von altem Code trennen
- Ggf. schon vor dem Umstieg WinForms Anwendung „aufräumen“
- Etwas mehr Zeit für Fehlersuche und Tests einplanen

Fazit

- WPF ist die Zukunft des Desktops
- WPF macht mehr Spaß ☺
- Schrittweiser Umstieg hält Lernkurve flach
- Hybride Anwendungen sind Notlösungen
- Wenn möglich mit WPF Anwendung von Grund auf neu schreiben

Links

- Stefan.Lange@empira.de
- Unterlagen zu dieser Session: www.st-lange.net