

Dependency and its role in modern programming languages

Antony Harfield
8th January 2010

THE UNIVERSITY OF
WARWICK

Background

JOENSUUN
YLIOPISTO

- Warwick, UK: Empirical Modelling Research Group



- Joensuu, Finland: EdTech Research Group

Current work

- France: international project to develop fusion as a renewable energy source



Tessella
Technology & Consulting

A bit of history

- A long time ago, before Java and .NET existed...
- People have been using dependency in software
- Computer scientists at Warwick developed principles for using dependency and tools for building software that use dependency

What is dependency?

- Values (e.g. a total) dependent on other values

Item	Quantity	Amount
Mango	5	£ 60.00
Coconut	1	£ 20.00
Durian	3	£ 75.00
Orange	8	£ 24.00
<i>Total</i>		£ 179.00

What is dependency?

- Another example of values dependent on other values from relational databases

Customer
CustomerId,
Name

Order
OrderId,
CustomerId,
Amount

ValuableCustomers
Name, TotalSpent

```
CREATE VIEW ValuableCustomers AS
SELECT Name, Sum(Amount) TotalSpent
FROM Customer INNER JOIN Order ON
Customer.CustomerId = Order.CustomerId
WHERE Sum(Amount) > 1000
GROUP BY Name
```

What is dependency?

- Properties (e.g. cell colour) dependent on values

Item	Quantity	Amount
Mango	5	฿ 60.00
Coconut	1	฿ 20.00
Durian	3	฿ 75.00
Orange	8	฿ 24.00
<i>Total</i>		฿ 179.00

What is dependency?

- Properties dependent on other properties?
 - A bit more difficult
 - Requires a notion of dependency at a low level in the application/programming language
- But, it gives the freedom to create dependencies between any objects, properties, and variables

Key ingredients of dependency

1. Observable (Variable, property or object)
2. Definition (Formula or function)

For example, in a spreadsheet:

- The *observable* is the cell or the *value* displayed in a cell
- The *definition* is the *formula* of the cell (e.g. the sum of a column of cells)

Empirical Modelling Tools

- EDEN – a general purpose modelling environment in which any variable/property can depend on other variables/properties
- Web EDEN – a web-based version of EDEN currently in development
- DOSTE – another general purpose dependency environment
- ADM – a tool for defining agents with dependency
- JAM – a tool for adding dependency to Java

Dependency is the key principle behind all these tools!

EDEN example

```

1. myexample = window {
2.   title = "Listbox Example: " // mylistbox_selecteditems[1];
3.   content = [mylistbox]
4. };
5.
6. mylistbox = listbox {
7.   selectmode = "browse";
8.   items = [ "blue", "red", "green", "yellow" ];
9.   selecteditems = [ "red" ];
10.  background = mylistbox_selecteditems[1];
11.  font = "Verdana 32";
12.  width = myexample_width;
13.  height = mylistbox_items#
14. };

```

Spot the dependencies

The rise of WPF

- Windows Presentation Foundation is Microsoft's latest API for creating Windows applications
- Much richer interfaces than existing Windows Forms UIs
- Because it uses DirectX
- WPF 3.5 (in .NET Framework 3.5) is considered mature – reasonable VisualStudio integration

What can you do with WPF?

- Groovy user interfaces!
 - The usual GUI components
 - Rich drawing model for 2D and 3D
 - Animation, audio, video
 - Styles, templating, layouts
- In a variety of formats:
 - Traditional windows application
 - Packaged web app
 - Silverlight RIAs (Rich Internet Applications)

How do you write WPF applications?

- User interfaces can be written in XML, using a language called XAML
- Code behind in any of the CLR languages (C#, VB.NET, etc)
- Or you could write it all in code – but XAML is much cleaner and allows you to separate your presentation logic from your business logic

WPF Example

```
<Window x:Class="CoolShapedWindow.Mickey"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Mickey" AllowsTransparency="True" WindowStyle="None"
  Background="Transparent">
  <Grid>
    <Image Source="famousmouse.png"
      MouseLeftButtonDown="Image_MouseLeftButtonDown"
      MouseRightButtonDown="Image_MouseRightButtonDown"/>
  </Grid>
</Window>
```



EM technologies and WPF

- What is the connection between WPF and Empirical Modelling?
- Dependency!
- Or more precisely, Microsoft's implementation of .NET dependency properties

Normal properties

- In OOP, classes usually have fields and methods
- But in .NET classes also have 'properties' that wrap getters and setters:

```
private String name;
public String Name {
    get { return name; }
    set { name = Value; }
}
```

Dependency properties

- Look like normal properties, but...
- Support change notification -> dependency
 - Bind one property to another
 - Triggered actions
- Default value inheritance
- Efficient storage

Dependency properties

- Most properties in WPF are dependency properties
- Therefore you can create dependencies between almost every aspect of your GUI
- You can create dependency properties in your custom classes so that you can make your GUI 'depend' upon your business objects

Binding

- A 'binding' is what creates the actual dependency
- For example:

```
<Slider Name="SourceSlider" Value="20" />

<TextBlock Name="TargetTextBlock"
  Text="Sawasdee Naresuan!"
  FontSize="{Binding ElementName=SourceSlider, Path=Value}"/>
```

 GettingStartedWithDataBinding.exe

Binding

- Equivalent binding in code:

```
Binding binding = new Binding();
binding.Source = SourceSlider;
binding.Path = new PropertyPath("Value");
binding.Mode = BindingMode.OneWay;
TargetTextBlock.SetBinding(FontSize, binding);
```

- Binding is nothing new: it has been used to bind domain objects to user interfaces for some time
- But (I think) WPF has brought out (or will bring out) the power of binding...

Examples

- Simple dependency
- Two way dependency
- Triggers
- Animation

Examples

```
<Window ...
  Title="{Binding ElementName=MyTextBox, Path=Text}">

  <StackPanel>
    <TextBox Name="MyTextBox" />

    <TextBlock Name="MyTextBlock" Text="{Binding
      ElementName=MyTextBox, Path=Text}" />
  ...
```

 GettingStartedWithDataBinding.exe

Examples (two-way binding)

```
<Slider Name="FontSizeSlider" Minimum="10" Maximum="50"
  Value="20" Margin="3" />
<TextBlock Name="MyTextBlock" Text="Hello World!"
  FontSize="{Binding ElementName=FontSizeSlider, Path=Value,
  Mode=TwoWay}" Margin="3" />
<StackPanel Orientation="Horizontal">
  <Button Click="Click_SetSliderValue" Margin="5">Set Slider
  Value</Button>
  <Button Click="Click_SetTextBlockFontSize" Margin="5">Set
  TextBlock FontSize</Button>
</StackPanel>
```

 GettingStartedWithDataBinding.exe

Examples (triggers)

```
<Style.Triggers>
  <Trigger Property="Control.IsMouseOver" Value="True">
    <Setter Property="Control.Foreground" Value="White" />
    <Setter Property="Control.Background" Value="Red" />
  </Trigger>
</Style.Triggers>
```



Examples (animation)

```
<Button Name="MyButton" HorizontalAlignment="Center" Width="100" Height="30">
  <Button.Triggers>
    <EventTrigger RoutedEvent="Mouse.MouseEnter">
      <BeginStoryboard>
        <Storyboard>
          <DoubleAnimation Storyboard.TargetProperty="Width" To="120" Duration="0:0:1" />
          <DoubleAnimation Storyboard.TargetProperty="Height" To="50" Duration="0:0:1" />
        </Storyboard>
      </BeginStoryboard>
    </EventTrigger>
    <EventTrigger RoutedEvent="Mouse.MouseLeave">
      <BeginStoryboard>
        <Storyboard>
          <DoubleAnimation Storyboard.TargetProperty="Width" To="100" Duration="0:0:1" />
          <DoubleAnimation Storyboard.TargetProperty="Height" To="30" Duration="0:0:1" />
        </Storyboard>
      </BeginStoryboard>
    </EventTrigger>
  </Button.Triggers>
  Button 1
</Button>
<ProgressBar Minimum="100" Maximum="120" Value="{Binding ElementName=MyButton, Path=Width}" Height="20"/>
```



EM / WPF comparisons

1. Types of dependency

- WPF has 4 types of binding:
 - One time
 - One way
 - Two way
 - One way to source – nasty
- EM has one type of dependency
 - E.g. $a = b + c$

EM / WPF comparisons

2. Complexity of definitions

- WPF makes it easier to do one-to-one bindings, but ‘multi-bindings’ require a bit code
 - If you want to do $a = f(x,y,z)$ then you need to write an `IMultiValueConverter` class for your function f
- EM languages allow functional definitions for dependencies
 - Simply create a definition $a = f(x,y,z)$

EM / WPF comparisons

3. Triggered actions

- Enable you to write (ADM-like) definitions such as ‘when this condition occurs, make this state change’
- WPF has good support (see button hover example)
- Triggers are fundamental concepts in EM

EM / WPF comparisons

4. User interface layout

- WPF is really the first technology that encourages laying out your user interface with dependency
 - Make the size and position of your components dependent on each other
- EM has been doing this for a while, but the graphics were quite primitive
 - Visual effects in WPF are impressive (full power of DirectX)

EM / WPF comparisons

5. Transformations

- WPF has some support
 - E.g. 'VisualBrush' that uses dependency/binding to paint components that are transformed
- In DoNaLD (Definitive Notation for Line Drawing), there are transformations that fully use the power of dependency

EM / WPF comparisons

6. Animations

- Very similar ways of doing animation
 - Create an iterator
 - Make positions, sizes, colours, styles dependent on the iterator (or some other component that is dependent on the iterator)

EM / WPF comparisons

7. Interactivity

- The biggest area of difference!
- WPF is compiled from XAML/C#
 - The dependencies are fixed
- EM technologies are interactive environments
 - Dependencies can be changed on-the-fly

EM / WPF summary

- WPF has excellent graphical capabilities
- WPF's dependency properties allow developers to build software artefacts that are more concise
- BUT...
- The complexity of the definitions and types of dependency could be much better
- It is never going to be an interactive environment

Flex has dependency too

But they are not called 'dependency properties'...

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
  xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="vertical">
  <mx:TextInput id="input" />
  <mx:Label text="{input.text}" />
</mx:Application>
```



Animation through dependency (Flex)

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  creationComplete="init()" layout="absolute">
  <mx:Script>
    [Bindable]
    public var counter:int = 0;

    public function init() {
      setInterval(function(){ counter++; }, 1000);
    }
  </mx:Script>
  <mx:Text text="Hello" x="{counter}" scaleY="{counter/10}"
    color="{counter*1024}" />
</mx:Application>
```



Running the examples

- To run the WPF examples you will need Visual Studio 2008
 - Create new project -> WPF Application
- To run the Flex examples you can download a trial version of Flex Builder from Adobe

More information

- Empirical Modelling:
www.warwick.ac.uk/go/em
- WPF: pick up a book, or Google for “wpf dependency properties”
- Flex: go to the Adobe Developer Connection (www.adobe.com/devnet/flex) or Flex After Dark (www.flexafterdark.com)

Thank you for your patience

Questions?

Antony Harfield
ant@dcs.warwick.ac.uk