Trusted Components

Reuse, Contracts and Patterns

Prof. Dr. Bertrand Meyer
Dr. Karine Arnout
Lecture 26: Component model: The .NET example
Agenda for today

- What is .NET?
- C# basics
- Eiffel for .NET
What .NET is not?

- **NOT** a programming language:
  - “C# is not .NET and .NET is not C#”

- **NOT** an operating system (yet).
What .NET is...

- An open-language platform for enterprise and Web development.
The scope of .NET

- .NET addresses almost everyone:
  - The general public
    - New user-friendly services
  - Businesses
  - Developers
    - Security
    - Programming-language interoperability
    - Component-based development
    - Versioning
    - Memory management
## The .NET architecture

<table>
<thead>
<tr>
<th>Component</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Web services</strong></td>
<td>UDDI, WSDL, Passport</td>
</tr>
<tr>
<td><strong>Open interchange formats</strong></td>
<td>XML &amp; SOAP</td>
</tr>
<tr>
<td><strong>Frameworks &amp; libraries</strong></td>
<td>ASP.NET, Windows Forms, Remoting…</td>
</tr>
<tr>
<td><strong>Specific language compilers</strong></td>
<td>C#, Visual Basic.Net, Managed C++, Cobol, Eiffel for .NET…</td>
</tr>
<tr>
<td><strong>Language interoperability</strong></td>
<td>Common Language Specification (CLS)</td>
</tr>
<tr>
<td><strong>Development environment</strong></td>
<td>Visual Studio.Net</td>
</tr>
<tr>
<td><strong>Compilation, execution…</strong></td>
<td>Common Language Runtime (CLR)</td>
</tr>
<tr>
<td><strong>Underlying platform</strong></td>
<td>Hardware, Operating system, database system</td>
</tr>
</tbody>
</table>
The Common Language Runtime

- Basic set of mechanisms to execute .NET programs.
- Virtual machine based on internal code: MSIL.
- MSIL: not interpreted but “jitted” to native platform.
- Built-in security mechanisms.
The .NET execution model

C#  VB  C++  Eiffel  ...

Language compilers

MSIL + Metadata

Loader

JIT + Verification

Managed code

Execution

.UNET languages

Chair of Software Engineering

Trusted Components: Reuse, Contracts and Patterns - Lecture 26
.NET specific vocabulary

- **Managed code**: Native code for the target platform, intended to run under the control and with the help of the CLR.
  - Ex: C#, Eiffel for .NET, Managed C++

- **Unmanaged code**: Native code that doesn’t rely on the CLR.
  - Ex: “Classic” Eiffel, Unmanaged C++
The assembly

- Compiled form of a set of classes
- The .NET unit of reuse (".exe" or ".dll")
- May be shared or private.
  - Notion of *Global Assembly Cache*
- Contain:
  - *MSIL code* (not binary code)
  - *Metadata* (interface information)
- Can be viewed with *ILDasm*. 
The Global Assembly Cache
.NET core assembly (mscorlib.dll)
## .NET libraries (1/2)

<table>
<thead>
<tr>
<th>Namespace</th>
<th>What for</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>Basic types: <em>Int32, String</em> ...</td>
</tr>
<tr>
<td>System.Collections</td>
<td>Collection types: <em>ArrayList</em>...</td>
</tr>
<tr>
<td>System.Data</td>
<td>Relational database management</td>
</tr>
<tr>
<td>System.Drawing</td>
<td>GDI graphics</td>
</tr>
<tr>
<td>System.Net</td>
<td>Network communications</td>
</tr>
<tr>
<td>System.Runtime.Remoting</td>
<td>Distributed computing</td>
</tr>
<tr>
<td>System.Security</td>
<td>Security features</td>
</tr>
</tbody>
</table>
### .NET libraries (2/2)

<table>
<thead>
<tr>
<th>Namespace</th>
<th>What for?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System.Text</td>
<td>ASCII, Unicode, etc. encoding</td>
</tr>
<tr>
<td>System.Text.RegularExpressions</td>
<td>Regular expressions handling</td>
</tr>
<tr>
<td>System.Threading</td>
<td>Thread support</td>
</tr>
<tr>
<td>System.Timers</td>
<td>Raising time-controlled events</td>
</tr>
<tr>
<td>System.Web.UI.WebControls</td>
<td>Graphical Web controls</td>
</tr>
<tr>
<td>System.Windows.Forms</td>
<td>Graphical controls: <em>Button</em>...</td>
</tr>
</tbody>
</table>

→ See the [Reference documentation](#) for more information.
Towards new services…

A (demo) conference registration page

(Demo) Registration Form for TOOLS USA 2001

*Title: Mr.
*First Name: John
*Last Name: Smith
Company Name: Computer Inc.
*Address: 1 Main Street
*City: HisCity
*State: CA
*Zip Code: 93117
*Country: USA
*Number of participants: 1
*Registering as: Regular
*I am registering for: Pre-Conference, July 28

 Register
Successful registration

![Image of a web page displaying registrant and registration tables]

### Registrants Table

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Address Form</th>
<th>First Name</th>
<th>Last Name</th>
<th>Company Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1050</td>
<td>Mr.</td>
<td>John</td>
<td>Smith</td>
<td>Computer Inc</td>
<td>1 Main Street, HisCity CA 93117, USA</td>
</tr>
</tbody>
</table>

### Registrations Table

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Registrant Identifier</th>
<th>Quantity</th>
<th>Discount Plan</th>
<th>Pre-Conference</th>
<th>WET</th>
<th>Conference</th>
<th>Eiffel Summit</th>
<th>Post-Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1048</td>
<td>1050</td>
<td>1</td>
<td>Regular</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>
Passing erroneous data

Server Error in '/demo' Application.

Precondition violation: valid_first_name

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Exception: Precondition violation valid_first_name

Source Error:

Line 29: // Add registrant
Line 30: registrant.add_registrant(address_form.SelectedItems.Value,  
Line 31: first_name.Value,  
Line 32: last_name.Value,  

Source File: C:\WWW\demo\Registration.aspx  Line: 31

Stack Trace:

[Exception: Precondition violation: valid_first_name]  
RegistrationService.Registrant.add_registrant(address_form.SelectedItems.Value,  
System.Web.UI.WebControls.Button.OnClick(Object source, EventArgs e) in C:\WWW\demo\Registration.aspx  
System.Web.UI.Page.RaisePostBackEvent(String e) in C:\WWW\demo\Registration.aspx  
System.Web.UI.Page.RaisePostBackEvent(Object source, EventArgs e) in C:\WWW\demo\Registration.aspx  
System.Web.UI.Page.ProcessRequestMain() +293

Version Information: Microsoft.NET Framework Version 1.0.3705.0; ASP.NET Version 1.0.3705.0
About this demo...

- Some of what this demo illustrates:
  - **ASP.NET** (Active Server Pages .NET)
  - Server-side mechanisms, server controls
  - Debugging Web applications like traditional ones
  - Our first C# example...
  - **Multi-language development** (C#, Eiffel for .NET)
```csharp
<% Assembly Name="registrationservice" %>
<% Import Namespace="RegistrationService" %>
<% Page Language="C#" %>

<HTML>
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-1252">
    <TITLE>Eiffel# Demo</TITLE>
    <LINK REL="stylesheet" TYPE="text/css" HREF="usa.css">
    <LINK HREF="http://www.eiffel.com/images/interface/eiffel_purple.ico" REL="SHORTCUT ICON">
    <SCRIPT RUNAT="SERVER">
        bool registered, register_click;
        String error_message;
        Registrar registrar;

        void Page_Init( Object Source, EventArgs E )
        {
            registrar = new Registrar();
            registrar.start();
            Application ["Registrar"] = registrar;
            registered = False;
            register_click = false;
        }

        void Register_click( object Source, EventArgs E )
        {
            register_click = true;

            // Add registrant.
            registrar.add_registrant( address_form.SelectedItem.value, 
                                      first_name.value, 
                                      last_name.value, 
                                      company_name.value, 
                                      address.value,
                                      email.value, 
                                      phone.value, 
                                      memo.value,
                                      dob.value, 
                                      ssn.value, 
                                      taxid.value, 
                                      role.value, 
                                      notes.value,
                                      name.value,
                                      password.value,
                                      confirm_password.value,
                                      expire.value,
                                      status.value );
        }
    </SCRIPT>
</HEAD>
```
Multi-language support

- C++
- C#
- Eiffel
- ...
“Microsoft languages”
- Visual Basic.Net
- C#
- Managed C++
- Jscript

Commercial languages (see next)
Research languages (see next)
Commercial offerings from 3rd parties:
- Cobol (Fujitsu)
- Eiffel for .NET (Eiffel Software)
- Smalltalk (QKS)
- Java (JUMP program)
- Perl, Python (ActiveState)
- APL (Dyadic)
- Fortran (Fujitsu)
- Fortran (Salford Software)
Languages on .NET (3/3)

- **Research languages:**
  - CAML (INRIA)
  - Mercury (Melbourne Uni.)
  - Scheme
  - Oberon (ETH, Zürich)
  - Component Pascal (QUT)
  - Haskell (Universities, Microsoft Research UK)
Full interoperability

- Cross-language...
  - Inheritance
    - Classes can inherit from each other, regardless of language of origin.
    - No need for wrappers
    - No IDL
  - Debugging sessions
    - In Visual Studio.Net
  - Exceptions
Full interoperability: the price to pay...

- Must **conform to the .NET object model** (VOS, *Virtual Object System*):
  - The type system
  - Object-oriented principles (type, inheritance)
- Too much for some (non-O-O languages)
- Too little for some: multiple inheritance (Eiffel)
- Difficult features: overloading

- Must **observe the Common Language Specification** (CLS)
Common Language Specification

- Set of **rules** – more restrictive than the .NET object model – to determine **compatibility**.
- Part of the **ECMA** standard
- Three levels:
  - Producer ("framework")
  - Consumer
  - Extender
CLS rules example

(Overloading)

“Methods and events that have the same name must be distinguished by at least one argument type.”

- **Producer**: Must mark any offending methods/events as non-CLS-compliant.
- **Consumer**: Need not accept types that violate these rules, except if marked non-compliant.
- **Extender**: Need not provide syntax for defining types that violate this rule.
The .NET type system

**Value**
- Built-in
  - Integer type
  - Floating type
- User-defined
  - Enums

**Reference**
- Class
- Interface
- Pointer
  - Managed
  - Unmanaged
  - Array
  - Delegate
  - Function

**Type**
- Value
- Reference
Value vs. reference types

- Value types denote values (simple value or objects).
- Reference types denote location of values.
Simple value vs. object

- A **simple value** cannot by itself **identify its type**.
  - Ex: int32, float 32

- An **object** carries its own **type description**.

A simple value

An object

Type info

Fields

"A string"
Value vs. reference types

- **Every value type** has a corresponding reference type: its “boxed” version.

- **Some reference types** have a corresponding value type: an “unboxed” version.

- **Limitation**: Inheritance and interfaces apply only to reference types.
Built-in types

- bool
- char
- int8, int16, int32, int64
- unsigned int8, unsigned int16, unsigned int32, unsigned int64
- native int, native unsigned int
- float32, float64
- System.object, System.string
- typedref

*not in CLS*
Agenda for today

- What is .NET?
- C# basics
- Eiffel for .NET
The C# language

- “Simplified version of C++”
- Benefits from the Java experience
- The native language of the .NET framework
C# and Java

- **Object-oriented**: classes, objects, dynamic allocation, inheritance, polymorphism, dynamic binding...
- **Type-safe** by default (no C++ casts)
- Support for **overloading**
- Support for garbage collection
- **No multiple inheritance** from classes
- **No genericity**
C# vs. Java

- Coherent type system
- Close integration with other languages
- Compiler directives
- Every piece of software is in a class
- Variables initialized automatically
- Event-delegate model
- Properties
- Exception handling
```
using System;
public class Hello {
    static void Main() {
        Console.WriteLine( "Hello World!" );
    }
}
```

Compiler command line: `csc Hello.cs`
Classes

- A class consists of members:
  - Fields
  - Methods
  - Properties
  - Events
Methods

- Algorithms associated with the class
- Work on an instance (unless static).
- Instance calls or virtual calls
- A **virtual** method may be overridden in a descendant class.
- A virtual method may be marked **final**.
Information hiding (1/2)

- Access every object through the official operations of its interface.
- Possible member status:
  - Public
  - Private
  - Family: descendant types only
  - Assembly
  - Family and assembly
  - Family or assembly
Information hiding (2/2)

- Direct field modification (e.g. C++, Java):
  
  ```
  heater.temperature := 19
  ```

- Proper O-O technique:

  ```
  set_temperature (t: INTEGER) is
  do
    if not in_range (t) then
      -- Error ...
    else
      temperature := t
      notify_observers
    end
  end
  ```
public class Heater {

    private int TemperatureInternal;

    public int Temperature {
        get { return TemperatureInternal; }
        set {
            if (!InRange(value)) {
                throw new ArgumentException("Temperature out of range");
            }
            TemperatureInternal = value;
            NotifyObservers;
        }
    }
}
Events and Delegates

- Purpose: Event-driven programming

- A delegate is an object representing a method of a particular class (like function pointer, but typed).

- Basic methods on delegates:
  - Invoke
  - BeginInvoke
  - EndInvoke

- Associate one or more delegates with each event.

- No way to specify “closed” arguments
Setting an event handler

Button ClickMe = new Button();
ClickMe.Text = "Click me!";

ClickMe.Location = new System.Drawing.Point (200, 30);
ClickMe.Size = new System.Drawing.Size (30, 10);

EventHandler MyHandler = new EventHandler (this.ButtonClicked);
ClickMe.Click += MyHandler;

void ButtonClicked (Object sender, EventArgs args) {...}
The event definition

```csharp
public class Button {

    public event EventHandler Click;

    protected void OnClick (EventArgs e) {
        if (Click != null)
            Click (this, e);
    }
}
```
Exception handling

- Similar to the Java model:

  ```java
  try {
      // code that may cause an exception
  }
  catch (ExceptionType exc) {
      if (CanFix()) {
          FixIt();
      } else throw new Exception();
  }
  catch (OtherExceptionType exc) {...}
  ```
Agenda for today

- What is .NET?
- C# basics
- Eiffel for .NET
Eiffel for .NET

- **Full implementation of Eiffel on .NET:**
  - Design by Contract
  - Seamlessness of software development
  - Multiple inheritance
  - Genericity...

- **Large set of libraries:**
  - Eiffel libraries + .NET libraries

- Directly **available in EiffelStudio**

- Integrated into **Visual Studio.NET: Eiffel ENViSioN!™**
  - Smart editor
  - Project clusters browsing...
Like Eiffel, Eiffel for .NET directly enforces Design by Contract, which addresses:

- Correctness
- Documentation
- Debugging and testing
- Inheritance control
- Management

Eiffel for .NET is the sole language on .NET to support contracts.
Seamlessness of software development

- Like Eiffel, Eiffel for .NET covers the entire software lifecycle.

- EiffelStudio provides complete reversibility between class text and diagrams.
Multiple inheritance and genericity...

- ... Or how to map non-CLS compliant mechanisms on .NET while preserving language interoperability?
Multiple inheritance on .NET (1/2)

- How Eiffel for .NET does it:
  - Taking full advantage of interfaces and namespaces.

Shadowing classes with interfaces.
What about class instantiation?
Eiffel for .NET:
- Supports genericity.
- Implements a safe variant of covariance (with type checking) detecting catcalls at run-time.
class

    HELLO_WORLD

create

    make

feature {NONE} -- Initialization

    make is

        -- Print Hello_world_message.

        do

            io.put_string (Hello_world_message)

        end

feature -- Constant

    Hello_world_message: STRING is "Hello World!"

        -- Hello World! message

end
With C#:

```csharp
public class HelloWorld1
{
    public static void Main()
    {
        HelloWorld sample;
        sample = Create.HelloWorld.Make();
    }
}
```
With VB.NET:

**Public Class** HelloWorld1

**Public Sub** Main ()

**Dim** sample as HelloWorld

sample = Create.HelloWorld.Make()

**End Sub**

**End Class**
With C#:

```csharp
using System;

public class HelloWorld2: Impl.HelloWorld {
    public static void Main() {
        HelloWorld sample;
        sample = new Impl.HelloWorld();
        Console.WriteLine (sample.HelloWorldMessage());
    }
}
```
With VB.NET:

```vbnet
Imports System

Public Class HelloWorld2
    Inherits Impl.HelloWorld

    Public Shared Sub Main()
        Dim h2 As HelloWorld2
        h2 = New HelloWorld2()
        Console.WriteLine(h2.HelloWorldMessage())
    End Sub

End Class
```
Eiffel for .NET in Visual Studio.NET

- EiffelStudio and Visual Studio.NET, two complementary development environments:
  - EiffelStudio for possibly multi-platform development in Eiffel only.
  - Visual Studio.NET for multi-language development.
... Or how to take advantage from .NET libraries from an Eiffel for .NET project.

Demo
It is possible to write ASP.NET pages directly in Eiffel for .NET.

```aspnet
<%@ Page Language = "Eiffel" %>
```

The registration page demo with Eiffel for .NET
References


End of lecture 26